pyrtma

Release 2.2.4

RNEL

May 08, 2024

CONTENTS

Functions

client_context	Context manager function to simplify initializing a pyrtma Client
get_header_cls	Get the correct header class depending on whether time-code is used
get_msg_cls	get msg class for a given message type ID
message_def	Decorator to add user message definitions.
msg_def	Decorator to add user message definitions.

CHAPTER

ONE

PYRTMA.CLIENT_CONTEXT

client_context(*module_id=0*, *server_name='localhost:7111'*, *msg_list=None*, *host_id=0*, *timecode=False*, *logger_status=False*, *daemon_status=False*)

Context manager function to simplify initializing a pyrtma Client

Context manager will yield a Client object after connecting to message manager, optionally subscribing to msg_list, and after calling send_module_ready(). Client will disconnect when exiting context.

Parameters

- **module_id** (*optional*) – Static module ID, which must be unique. Defaults to 0, which generates a dynamic module ID.
- **server_name** (*optional*) – IP_addr:port_num string associated with message manager. Defaults to "localhost:7111".
- **msg_list** (*optional*) – A list of numeric message IDs to subscribe to
- **host_id** (*optional*) – Host ID. Defaults to 0.
- **timecode** (*optional*) – Add additional timecode fields to message header, used by some projects at RNEL. Defaults to False.
- **logger_status** (*optional*) – Flag to declare client as a logger module. Logger modules are automatically subscribed to all message types. Defaults to False.
- **daemon_status** (*optional*) – Flag to declare client as a daemon. Defaults to False.

Yields

Client – initialized pyrtma Client object

CHAPTER

TWO

PYRTMA.GET_HEADER_CLS

get_header_cls(*timecode=False*)

Get the correct header class depending on whether timecode is used

Parameters

timecode (*bool, optional*) – Flag indicating if timecode fields are needed. Defaults to False.

Returns

MessageHeader class

Return type

Type[*MessageHeader*]

CHAPTER

THREE

PYRTMA.GET_MSG_CLS

get_msg_cls(*id*)

get msg class for a given message type ID

Parameters
id (*int*) – Message Type ID

Raises
UnknownMessageType – Message type is undefined

Returns
Message class

Return type
Type[*MessageData*]

CHAPTER

FOUR

PYRTMA.MESSAGE_DEF

message_def(*msg_cls*, **args*, ***kwargs*)

Decorator to add user message definitions.

Return type

Type[*TypeVar*(_MD, bound= *MessageData*)]

Parameters

msg_cls (*Type[_MD]*) –

CHAPTER

FIVE

PYRTMA.MSG_DEF

msg_def(*msg_cls*, **args*, ***kwargs*)

Decorator to add user message definitions.

Return type
Type[*TypeVar*(_MD, bound= *MessageData*)]

Parameters
msg_cls (*Type[_MD]*) –

Classes

Client	RTMA Client interface
Message	Message class
MessageData	MessageData base class
MessageHeader	RTMA Message Header class

CHAPTER

SIX

PYRTMA.CLIENT

class Client(*module_id=0*, *host_id=0*, *timecode=False*)

Bases: object

RTMA Client interface

Parameters

- **module_id** (*optional*) – Static module ID, which must be unique. Defaults to 0, which generates a dynamic module ID.
- **host_id** (*optional*) – Host ID. Defaults to 0.
- **timecode** (*optional*) – Add additional timecode fields to message header, used by some projects at RNEL. Defaults to False.

Methods

connect	Connect to message manager server
discard_messages	Read and discard messages in socket buffer up to timeout
disconnect	Disconnect from message manager server
forward_message	Forward a message
pause_all_subscriptions	Pause all subscribed types
pause_subscription	Pause subscription to message types
paused_subscription_context	Context manager to pause subscriptions to a list of message types
read_message	Read a message
resume_all_subscriptions	Resume all paused subscriptions
resume_subscription	Resume subscription to message types
send_message	Send a message
send_module_ready	Send a signal to message manager that client is ready
send_signal	Send a signal
subscribe	Subscribe to message types
subscription_context	Context manager to subscribe to a list of message types
unsubscribe	Unsubscribe from message types
unsubscribe_from_all	Unsubscribe from all subscribed types

Attributes

connected	Status of connection to message manager server
header_cls	Class defining the RTMA message header
ip_addr	Message manager IP address string
module_id	Numeric module ID of client
msg_count	Count of messages that have been sent
paused_subscribed_types	Subscriptions on pause
port	Message manager port number
server	Message manager server address as a (IP_addr, port_num) tuple
sock	Underlying socket connection with MessageManager
subscribed_types	List of subscribed message types

connect(*server_name='localhost:7111'*, *logger_status=False*, *daemon_status=False*)

Connect to message manager server

Parameters

- **server_name** (*optional*) – IP_addr:port_num string associated with message manager. Defaults to "localhost:7111".
- **logger_status** (*optional*) – Flag to declare client as a logger module. Logger modules are automatically subscribed to all message types. Defaults to False.
- **daemon_status** (*optional*) – Flag to declare client as a daemon. Defaults to False.

Raises

MessageManagerNotFound – Unable to connect to message manager

property connected: bool

Status of connection to message manager server

discard_messages(*timeout=1*)

Read and discard messages in socket buffer up to timeout

Parameters

timeout (*optional*) – Maximum time in seconds to loop through message buffer. Defaults to 1.

Return type

bool

Returns

True if all messages have been read, False if messages remain in buffer

disconnect()

Disconnect from message manager server

forward_message(*msg_hdr*, *msg_data=None*, *timeout=-1*)

Forward a message

A message is a packet that contains a defined data payload. To send a message without associated data, see *send_signal()*.

Parameters

- **msg_hdr** (*MessageHeader*) – Object containing RTMA header to send

- **msg_data** (Optional[*MessageData*]) – Object containing the message to send
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

property header_cls: Type[*MessageHeader*]

Class defining the RTMA message header

property ip_addr: str

Message manager IP address string

property module_id: int

Numeric module ID of client

property msg_count: int

Count of messages that have been sent

pause_all_subscriptions()

Pause all subscribed types

pause_subscription(*msg_list*)

Pause subscription to message types

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to temporarily unsubscribe to

property paused_subscribed_types: Set[int]

Subscriptions on pause

paused_subscription_context(*msg_list*)

Context manager to pause subscriptions to a list of message types

Message types will automatically resume subscriptions after exiting context.

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to temporarily unsubscribe to

property port: int

Message manager port number

read_message(*timeout=-1*, *ack=False*, *sync_check=False*)

Read a message

Parameters

- **timeout** (*optional*) – Timeout to wait for a message to be available for reading. Defaults to -1 (blocking).
- **ack** (*optional*) – Primarily for internal use. When True, will not discard ACK messages. Defaults to False.
- **sync_check** (*optional*) – Validate message definition matches header version. Defaults to False.

Raises

ConnectionLost – Connection error to message manager server

Return type

Optional[*Message*]

Returns

Message object. If no message is read before timeout, returns None.

resume_all_subscriptions()

Resume all paused subscriptions

resume_subscription(*msg_list*)

Resume subscription to message types

Parameters

msg_list (*Iterable[int]*) – A list of paused message IDs to resubscribe to

send_message(*msg_data*, *dest_mod_id=0*, *dest_host_id=0*, *timeout=-1*)

Send a message

A message is a packet that contains a defined data payload. To send a message without associated data, see *send_signal()*.

Parameters

- **msg_data** (*MessageData*) – Object containing the message to send
- **dest_mod_id** (*optional*) – Specific module ID to send to. Defaults to 0 (broadcast).
- **dest_host_id** (*optional*) – Specific host ID to send to. Defaults to 0 (broadcast).
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

Raises

- ***InvalidDestinationModule*** – Specified destination module is invalid
- ***InvalidDestinationHost*** – Specified destination host is invalid

send_module_ready()

Send a signal to message manager that client is ready

This method also sends the client's process ID to message manager.

send_signal(*signal_type*, *dest_mod_id=0*, *dest_host_id=0*, *timeout=-1*)

Send a signal

A signal is a message type without an associated data payload. Only a unique message type ID is required to send a signal. To send a message with data, see *send_message()*.

Parameters

- **signal_type** (*int*) – Numeric message type ID of signal
- **dest_mod_id** (*optional*) – Specific module ID to send to. Defaults to 0 (broadcast).
- **dest_host_id** (*optional*) – Specific host ID to send to. Defaults to 0 (broadcast).
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

Raises

- ***InvalidDestinationModule*** – Specified destination module is invalid
- ***InvalidDestinationHost*** – Specified destination host is invalid

property server: Tuple[str, int]

Message manager server address as a (IP_addr, port_num) tuple

property sock: socket

Underlying socket connection with MessageManager

subscribe(*msg_list*)

Subscribe to message types

Calling this method multiple times will add to, and not replace, the list of subscribed messages.

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to subscribe to

property subscribed_types: Set[int]

List of subscribed message types

subscription_context(*msg_list*)

Context manager to subscribe to a list of message types

Message types will automatically unsubscribe after exiting context.

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to subscribe to

unsubscribe(*msg_list*)

Unsubscribe from message types

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to unsubscribe to

unsubscribe_from_all()

Unsubscribe from all subscribed types

CHAPTER

SEVEN

PYRTMA.MESSAGE

class Message(*header*, *data*)

Bases: object

Message class

Contains message header and data

Methods

copy	Generate a copy of a message structure
from_json	Create message object from JSON string
pretty_print	Generate formatted string for pretty printing of message
to_dict	Convert message to dictionary
to_json	Convert message to JSON string

Attributes

name
type_id

Parameters

- **header** (MessageHeader) –
- **data** (MessageData) –

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*Message*) – Message structure to copy

Return type
Message

classmethod from_json(*s*)

Create message object from JSON string

Parameters
s (*str*) – JSON message string

Raises
InvalidMessageDefinition – JSON data does not match expected message defintion

Returns
Message object

Return type
Message

pretty_print(*add_tabs=0*)

Generate formatted string for pretty printing of message

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to JSON string

Parameters
minify (*bool, optional*) – Flag to minify (compact format). Defaults to False.

Returns
JSON message string

Return type
str

CHAPTER

EIGHT

PYRTMA.MESSAGEDATA

class MessageData

Bases: *MessageBase*

MessageData base class

This is intended to be treated as an abstract class and and should not be directly instantiated.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_id
type_name
type_size
type_source
type_hash

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

CHAPTER

NINE

PYRTMA.MESSAGEHEADER

class MessageHeader

Bases: *MessageBase*

RTMA Message Header class

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	
src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

Exceptions

AcknowledgementTimeout	Raised when client does not receive ack from message manager.
ClientError	Base exception for all Client Errors.
ConnectionLost	Raised when there is a connection error with the server.
InvalidDestinationHost	Raised when client tries to send to an invalid host.
InvalidDestinationModule	Raised when client tries to send to an invalid module.
MessageManagerNotFound	Raised when unable to connect to message manager.
NotConnectedError	Raised when the client tries to read/write while not connected.

CHAPTER

TEN

PYRTMA.ACKNOWLEDGEMENTTIMEOUT

exception AcknowledgementTimeout

Raised when client does not receive ack from message manager.

CHAPTER

ELEVEN

PYRTMA.CLIENTERROR

exception ClientError

Base exception for all Client Errors.

CHAPTER

TWELVE

PYRTMA.CONNECTIONLOST

exception ConnectionLost

Raised when there is a connection error with the server.

CHAPTER

THIRTEEN

PYRTMA.INVALIDDESTINATIONHOST

exception InvalidDestinationHost

Raised when client tries to send to an invalid host.

CHAPTER

FOURTEEN

PYRTMA.INVALIDDESTINATIONMODULE

exception InvalidDestinationModule

Raised when client tries to send to an invalid module.

CHAPTER

FIFTEEN

PYRTMA.MESSAGEMANAGERNOTFOUND

exception MessageManagerNotFound

Raised when unable to connect to message manager.

CHAPTER

SIXTEEN

PYRTMA.NOTCONNECTEDERROR

exception NotConnectedError

Raised when the client tries to read/write while not connected.

pyrtma.client	pyrtma.client module
pyrtma.compile	pyrtma.compile Message Type Compiler
pyrtma.constants	pyrtma.constants module
pyrtma.core_defs	This message def file was auto-generated by pyrtma.compile version 2.2.4
pyrtma.exceptions	
pyrtma.header	
pyrtma.manager	pyrtma.manager module
pyrtma.message	pyrtma.messaage: RTMA message classes
pyrtma.message_base	
pyrtma.message_data	
pyrtma.parser	Message definition YAML parser
pyrtma.validators	
pyrtma.web_manager	

CHAPTER

SEVENTEEN

PYRTMA.CLIENT

pyrtma.client module

Includes *Client* class and associated exception classes

Functions

cast	Cast a value to a type.
client_context	Context manager function to simplify initializing a pyrtma Client
contextmanager	@contextmanager decorator.
get_header_cls	Get the correct header class depending on whether timecode is used
get_msg_cls	get msg class for a given message type ID
requires_connection	Decorator wrapper for Client methods that require a connection
warn	Issue a warning, or maybe ignore it or raise an exception.
wraps	Decorator factory to apply update_wrapper() to a wrapper function

17.1 pyrtma.client.cast

cast(*typ*, *val*)

Cast a value to a type.

This returns the value unchanged. To the type checker this signals that the return value has the designated type, but at runtime we intentionally don't check anything (we want this to be as fast as possible).

17.2 pyrtma.client.client_context

client_context(*module_id=0*, *server_name='localhost:7111'*, *msg_list=None*, *host_id=0*, *timecode=False*, *logger_status=False*, *daemon_status=False*)

Context manager function to simplify initializing a pyrtma Client

Context manager will yield a Client object after connecting to message manager, optionally subscribing to msg_list, and after calling send_module_ready(). Client will disconnect when exiting context.

Parameters

- **module_id** (*optional*) – Static module ID, which must be unique. Defaults to 0, which generates a dynamic module ID.
- **server_name** (*optional*) – IP_addr:port_num string associated with message manager. Defaults to "localhost:7111".
- **msg_list** (*optional*) – A list of numeric message IDs to subscribe to
- **host_id** (*optional*) – Host ID. Defaults to 0.
- **timecode** (*optional*) – Add additional timecode fields to message header, used by some projects at RNEL. Defaults to False.
- **logger_status** (*optional*) – Flag to declare client as a logger module. Logger modules are automatically subscribed to all message types. Defaults to False.
- **daemon_status** (*optional*) – Flag to declare client as a daemon. Defaults to False.

Yields
Client – initialized pyrtma Client object

17.3 pyrtma.client.contextmanager

contextmanager(*func*)

@contextmanager decorator.

Typical usage:

@contextmanager def some_generator(<arguments>):

<setup> try:

yield <value>

finally:
<cleanup>

This makes this:

with some_generator(<arguments>) as <variable>:
<body>

equivalent to this:

<setup> try:

<variable> = <value> <body>

finally:
<cleanup>

17.4 pyrtma.client.get_header_cls

get_header_cls(*timecode=False*)

Get the correct header class depending on whether timecode is used

Parameters
timecode (*bool, optional*) – Flag indicating if timecode fields are needed. Defaults to False.

Returns
MessageHeader class

Return type
Type[*MessageHeader*]

17.5 pyrtma.client.get_msg_cls

get_msg_cls(*id*)

get msg class for a given message type ID

Parameters
id (*int*) – Message Type ID

Raises
UnknownMessageType – Message type is undefined

Returns
Message class

Return type
Type[*MessageData*]

17.6 pyrtma.client.requires_connection

requires_connection(*func*)

Decorator wrapper for Client methods that require a connection

Return type
TypeVar(F, bound= Callable[..., Any])

Parameters
func (*F*) –

17.7 pyrtma.client.warn

warn(*message*, *category=None*, *stacklevel=1*, *source=None*)

Issue a warning, or maybe ignore it or raise an exception.

17.8 pyrtma.client.wraps

wraps(*wrapped*, *assigned=('__module__', '__name__', '__qualname__', '__doc__', '__annotations__')*, *updated=('__dict__',)*)

Decorator factory to apply update_wrapper() to a wrapper function

Returns a decorator that invokes update_wrapper() with the decorated function as the wrapper argument and the arguments to wraps() as the remaining arguments. Default arguments are as for update_wrapper(). This is a convenience function to simplify applying partial() to update_wrapper().

Classes

Client	RTMA Client interface
Message	Message class
MessageData	MessageData base class
MessageHeader	RTMA Message Header class
TypeVar	Type variable.

17.9 pyrtma.client.Client

class Client(*module_id=0*, *host_id=0*, *timecode=False*)

Bases: object

RTMA Client interface

Parameters

- **module_id** (*optional*) – Static module ID, which must be unique. Defaults to 0, which generates a dynamic module ID.
- **host_id** (*optional*) – Host ID. Defaults to 0.
- **timecode** (*optional*) – Add additional timecode fields to message header, used by some projects at RNEL. Defaults to False.

Methods

connect	Connect to message manager server
discard_messages	Read and discard messages in socket buffer up to timeout
disconnect	Disconnect from message manager server
forward_message	Forward a message
pause_all_subscriptions	Pause all subscribed types
pause_subscription	Pause subscription to message types
paused_subscription_context	Context manager to pause subscriptions to a list of message types
read_message	Read a message
resume_all_subscriptions	Resume all paused subscriptions
resume_subscription	Resume subscription to message types
send_message	Send a message
send_module_ready	Send a signal to message manager that client is ready
send_signal	Send a signal
subscribe	Subscribe to message types
subscription_context	Context manager to subscribe to a list of message types
unsubscribe	Unsubscribe from message types
unsubscribe_from_all	Unsubscribe from all subscribed types

Attributes

connected	Status of connection to message manager server
header_cls	Class defining the RTMA message header
ip_addr	Message manager IP address string
module_id	Numeric module ID of client
msg_count	Count of messages that have been sent
paused_subscribed_types	Subscriptions on pause
port	Message manager port number
server	Message manager server address as a (IP_addr, port_num) tuple
sock	Underlying socket connection with MessageManager
subscribed_types	List of subscribed message types

connect(*server_name='localhost:7111'*, *logger_status=False*, *daemon_status=False*)

Connect to message manager server

Parameters

- **server_name** (*optional*) – IP_addr:port_num string associated with message manager. Defaults to "localhost:7111".
- **logger_status** (*optional*) – Flag to declare client as a logger module. Logger modules are automatically subscribed to all message types. Defaults to False.
- **daemon_status** (*optional*) – Flag to declare client as a daemon. Defaults to False.

Raises

MessageManagerNotFound – Unable to connect to message manager

property connected: bool

Status of connection to message manager server

discard_messages(*timeout=1*)

Read and discard messages in socket buffer up to timeout

Parameters
timeout (*optional*) – Maximum time in seconds to loop through message buffer. Defaults to 1.

Return type
bool

Returns
True if all messages have been read, False if messages remain in buffer

disconnect()

Disconnect from message manager server

forward_message(*msg_hdr*, *msg_data=None*, *timeout=-1*)

Forward a message

A message is a packet that contains a defined data payload. To send a message without associated data, see *send_signal()*.

Parameters

- **msg_hdr** (*MessageHeader*) – Object containing RTMA header to send
- **msg_data** (Optional[*MessageData*]) – Object containing the message to send
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

property header_cls: Type[*MessageHeader*]

Class defining the RTMA message header

property ip_addr: str

Message manager IP address string

property module_id: int

Numeric module ID of client

property msg_count: int

Count of messages that have been sent

pause_all_subscriptions()

Pause all subscribed types

pause_subscription(*msg_list*)

Pause subscription to message types

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to temporarily unsubscribe to

property paused_subscribed_types: Set[int]

Subscriptions on pause

paused_subscription_context(*msg_list*)

Context manager to pause subscriptions to a list of message types

Message types will automatically resume subscriptions after exiting context.

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to temporarily unsubscribe to

property port: int

Message manager port number

read_message(*timeout=-1*, *ack=False*, *sync_check=False*)

Read a message

Parameters

- **timeout** (*optional*) – Timeout to wait for a message to be available for reading. Defaults to -1 (blocking).
- **ack** (*optional*) – Primarily for internal use. When True, will not discard ACK messages. Defaults to False.
- **sync_check** (*optional*) – Validate message definition matches header version. Defaults to False.

Raises
ConnectionLost – Connection error to message manager server

Return type
Optional[*Message*]

Returns
Message object. If no message is read before timeout, returns None.

resume_all_subscriptions()

Resume all paused subscriptions

resume_subscription(*msg_list*)

Resume subscription to message types

Parameters
msg_list (*Iterable[int]*) – A list of paused message IDs to resubscribe to

send_message(*msg_data*, *dest_mod_id=0*, *dest_host_id=0*, *timeout=-1*)

Send a message

A message is a packet that contains a defined data payload. To send a message without associated data, see *send_signal()*.

Parameters

- **msg_data** (*MessageData*) – Object containing the message to send
- **dest_mod_id** (*optional*) – Specific module ID to send to. Defaults to 0 (broadcast).
- **dest_host_id** (*optional*) – Specific host ID to send to. Defaults to 0 (broadcast).
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

Raises

- *InvalidDestinationModule* – Specified destination module is invalid
- *InvalidDestinationHost* – Specified destination host is invalid

send_module_ready()

Send a signal to message manager that client is ready

This method also sends the client's process ID to message manager.

send_signal(*signal_type*, *dest_mod_id=0*, *dest_host_id=0*, *timeout=-1*)

Send a signal

A signal is a message type without an associated data payload. Only a unique message type ID is required to send a signal. To send a message with data, see *send_message()*.

Parameters

- **signal_type** (*int*) – Numeric message type ID of signal
- **dest_mod_id** (*optional*) – Specific module ID to send to. Defaults to 0 (broadcast).
- **dest_host_id** (*optional*) – Specific host ID to send to. Defaults to 0 (broadcast).
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

Raises

- *InvalidDestinationModule* – Specified destination module is invalid
- *InvalidDestinationHost* – Specified destination host is invalid

property server: Tuple[str, int]

Message manager server address as a (IP_addr, port_num) tuple

property sock: socket

Underlying socket connection with MessageManager

subscribe(*msg_list*)

Subscribe to message types

Calling this method multiple times will add to, and not replace, the list of subscribed messages.

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to subscribe to

property subscribed_types: Set[int]

List of subscribed message types

subscription_context(*msg_list*)

Context manager to subscribe to a list of message types

Message types will automatically unsubscribe after exiting context.

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to subscribe to

unsubscribe(*msg_list*)

Unsubscribe from message types

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to unsubscribe to

unsubscribe_from_all()

Unsubscribe from all subscribed types

17.10 pyrtma.client.Message

class Message(*header*, *data*)

Bases: object

Message class

Contains message header and data

Methods

copy	Generate a copy of a message structure
from_json	Create message object from JSON string
pretty_print	Generate formatted string for pretty printing of message
to_dict	Convert message to dictionary
to_json	Convert message to JSON string

Attributes

name
type_id

Parameters

- **header** (MessageHeader) –
- **data** (MessageData) –

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*Message*) – Message structure to copy

Return type
Message

classmethod from_json(*s*)

Create message object from JSON string

Parameters
s (*str*) – JSON message string

Raises
InvalidMessageDefinition – JSON data does not match expected message defintion

Returns
Message object

Return type
Message

pretty_print(*add_tabs=0*)

Generate formatted string for pretty printing of message

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to JSON string

Parameters
minify (*bool, optional*) – Flag to minify (compact format). Defaults to False.

Returns
JSON message string

Return type
str

17.11 pyrtma.client.MessageData

class MessageData

Bases: *MessageBase*

MessageData base class

This is intended to be treated as an abstract class and and should not be directly instantiated.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_id
type_name
type_size
type_source
type_hash

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

17.12 pyrtma.client.MessageHeader

class MessageHeader

Bases: *MessageBase*

RTMA Message Header class

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	
src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

17.13 pyrtma.client.TypeVar

class TypeVar(*name*, **constraints*, *bound=None*, *covariant=False*, *contravariant=False*)

Bases: _Final, _Immutable

Type variable.

Usage:

```
T = TypeVar('T')  # Can be anything
A = TypeVar('A', str, bytes)  # Must be str or bytes
```

Type variables exist primarily for the benefit of static type checkers. They serve as the parameters for generic types as well as for generic function definitions. See class Generic for more information on generic types. Generic functions work as follows:

def repeat(x: T, n: int) -> List[T]:
'''Return a list containing n references to x.''' return [x]*n

def longest(x: A, y: A) -> A:
'''Return the longest of two strings.''' return x if len(x) >= len(y) else y

The latter example's signature is essentially the overloading of (str, str) -> str and (bytes, bytes) -> bytes. Also note that if the arguments are instances of some subclass of str, the return type is still plain str.

At runtime, isinstance(x, T) and issubclass(C, T) will raise TypeError.

Type variables defined with covariant=True or contravariant=True can be used to declare covariant or contravariant generic types. See PEP 484 for more details. By default generic types are invariant in all type variables.

Type variables can be introspected. e.g.:

T.__name__ == 'T' T.__constraints__ == () T.__covariant__ == False T.__contravariant__ = False A.__constraints__ == (str, bytes)

Note that only type variables defined in global scope can be pickled.

Methods

Exceptions

AcknowledgementTimeout	Raised when client does not receive ack from message manager.
ClientError	Base exception for all Client Errors.
ConnectionLost	Raised when there is a connection error with the server.
InvalidDestinationHost	Raised when client tries to send to an invalid host.
InvalidDestinationModule	Raised when client tries to send to an invalid module.
InvalidMessageDefinition	Raised when there is message definition is out of sync with sent data.
MessageManagerNotFound	Raised when unable to connect to message manager.
NotConnectedError	Raised when the client tries to read/write while not connected.
SocketOptionError	Raised when unable to set socket options.
UnknownMessageType	Raised when there is no message definition.

17.14 pyrtma.client.AcknowledgementTimeout

exception AcknowledgementTimeout

Raised when client does not receive ack from message manager.

17.15 pyrtma.client.ClientError

exception ClientError

Base exception for all Client Errors.

17.16 pyrtma.client.ConnectionLost

exception ConnectionLost

Raised when there is a connection error with the server.

17.17 pyrtma.client.InvalidDestinationHost

exception InvalidDestinationHost

Raised when client tries to send to an invalid host.

17.18 pyrtma.client.InvalidDestinationModule

exception InvalidDestinationModule

Raised when client tries to send to an invalid module.

17.19 pyrtma.client.InvalidMessageDefinition

exception InvalidMessageDefinition

Raised when there is message definition is out of sync with sent data.

17.20 pyrtma.client.MessageManagerNotFound

exception MessageManagerNotFound

Raised when unable to connect to message manager.

17.21 pyrtma.client.NotConnectedError

exception NotConnectedError

Raised when the client tries to read/write while not connected.

17.22 pyrtma.client.SocketOptionError

exception SocketOptionError

Raised when unable to set socket options.

17.23 pyrtma.client.UnknownMessageType

exception UnknownMessageType

Raised when there is no message definition.

CHAPTER

EIGHTEEN

PYRTMA.COMPILE

pyrtma.compile Message Type Compiler

Functions

compile	compile message defs
install	Install a rich traceback handler.
main	**rtype** None

18.1 pyrtma.compile.compile

compile(*defs_files*, *out_dir*, *out_name*, *python=False*, *javascript=False*, *matlab=False*, *c_lang=False*, *info=False*, *combined=False*, *debug=False*, *validate_alignment=True*, *auto_pad=True*, *import_coredefs=True*)

compile message defs

Parameters

- **defs_files** (*List[str]*) – Root YAML message defintion file to parse. List of C header file(s) will use v1 python compiler (deprecated)
- **out_name** (*str*) – Output directory for compiled files. For v1 compiler (deprecated), full output filename.
- **python** (*bool, optional*) – Output python .py file. Defaults to False.
- **javascript** (*bool, optional*) – Output javascript .js file. Defaults to False.
- **matlab** (*bool, optional*) – Output matlab .m file. Defaults to False.
- **c_lang** (*bool, optional*) – Output C .h file. Defaults to False.
- **info** (*bool, optional*) – Output info .txt file. Defaults to False.
- **combined** (*bool, optional*) – Output combined YAML file. Defaults to False.
- **debug** (*bool, optional*) – Debug mode. Defaults to False.
- **validate_alignment** (*bool, optional*) – Validate message 64-bit alignment. Defaults to True.

- **auto_pad** (*bool, optional*) – Automatically pad messages failing 64-bit alignment validation. Defaults to True. Has no effect if validate_alignment is False.
- **import_coredefs** (*bool, optional*) – Automatically import pyrtma.core_defs. Defaults to True.
- **out_dir** (*str*) –

Raises

- ***FileFormatError*** – Issue with input file format
- **FileExistsError** – Output file is not a directory
- **RuntimeError** – Invalid output filename

18.2 pyrtma.compile.install

install(**, console=None, width=100, extra_lines=3, theme=None, word_wrap=False, show_locals=False, locals_max_length=10, locals_max_string=80, locals_hide_dunder=True, locals_hide_sunder=None, indent_guides=True, suppress=(), max_frames=100*)

Install a rich traceback handler.

Once installed, any tracebacks will be printed with syntax highlighting and rich formatting.

Parameters

- **console** (*Optional[Console], optional*) – Console to write exception to. Default uses internal Console instance.
- **width** (*Optional[int], optional*) – Width (in characters) of traceback. Defaults to 100.
- **extra_lines** (*int, optional*) – Extra lines of code. Defaults to 3.
- **theme** (*Optional[str], optional*) – Pygments theme to use in traceback. Defaults to None which will pick a theme appropriate for the platform.
- **word_wrap** (*bool, optional*) – Enable word wrapping of long lines. Defaults to False.
- **show_locals** (*bool, optional*) – Enable display of local variables. Defaults to False.
- **locals_max_length** (*int, optional*) – Maximum length of containers before abbreviating, or None for no abbreviation. Defaults to 10.
- **locals_max_string** (*int, optional*) – Maximum length of string before truncating, or None to disable. Defaults to 80.
- **locals_hide_dunder** (*bool, optional*) – Hide locals prefixed with double underscore. Defaults to True.
- **locals_hide_sunder** (*bool, optional*) – Hide locals prefixed with single underscore. Defaults to False.
- **indent_guides** (*bool, optional*) – Enable indent guides in code and locals. Defaults to True.
- **suppress** (*Sequence[Union[str, ModuleType]]*) – Optional sequence of modules or paths to exclude from traceback.
- **max_frames** (*int*) –

Returns
The previous exception handler that was replaced.

Return type
Callable

18.3 pyrtma.compile.main

`main()`

Return type
None

Classes

CDefCompiler	
InfoCompiler	
JSDefCompiler	
MatlabDefCompiler	
Parser	Parser class
PyDefCompiler	
YAMLCompiler	

18.4 pyrtma.compile.CDefCompiler

`class CDefCompiler(parser, filename, debug=False)`

Bases: object

Methods

generate	
generate_close_guard	**rtype** str
generate_constant	
generate_hash_id	**rtype** str
generate_host_id	**rtype** str
generate_includes	**rtype** str
generate_module_id	**rtype** str
generate_msg_type_id	**rtype** str
generate_rtma_info_getter	**rtype** str
generate_string_constant	
generate_struct	**rtype** str
generate_type_alias	**rtype** str
generate_type_info	**rtype** str
generate_version_comment	**rtype** str

Parameters

- **parser** (Parser) –
- **filename** (*str*) –
- **debug** (*bool*) –

18.5 pyrtma.compile.InfoCompiler

class InfoCompiler(*parser*, *filename*, *debug=False*)

Bases: object

Methods

generate

Parameters

- **parser** (Parser) –
- **filename** (*str*) –
- **debug** (*bool*) –

18.6 pyrtma.compile.JSDefCompiler

class JSDefCompiler(*parser*, *debug=False*)

Bases: object

Methods

generate	
generate_constant	
generate_hash_id	**rtype** str
generate_host_id	**rtype** str
generate_module_id	**rtype** str
generate_msg_type_id	**rtype** str
generate_obj	**rtype** str
generate_prop	
generate_string_constant	
generate_type_alias	**rtype** str

Parameters

- **parser** (Parser) –
- **debug** (*bool*) –

18.7 pyrtma.compile.MatlabDefCompiler

class MatlabDefCompiler(*parser*, *debug=False*)

Bases: object

Methods

generate	
generate_by_MT	**rtype** str
generate_constant	**rtype** str
generate_constant_string	
generate_fcn_close	**rtype** str
generate_fcn_header	**rtype** str
generate_field	**rtype** str
generate_hash_id	**rtype** str
generate_host_id	**rtype** str
generate_message_header	**rtype** str
generate_mex_opcodes	**rtype** str
generate_module_id	**rtype** str
generate_msg_def	**rtype** str
generate_msg_type_id	**rtype** str
generate_struct	**rtype** str
generate_type_alias	

Parameters

- **parser** (Parser) –
- **debug** (*bool*) –

18.8 pyrtma.compile.Parser

class Parser(*debug=False*, *validate_alignment=True*, *auto_pad=True*, *import_coredefs=True*)

Bases: object

Parser class

Parser class

Parameters

- **debug** (*bool, optional*) – Flag for debug mode. Defaults to False.
- **validate_alignment** (*bool*) –
- **auto_pad** (*bool*) –
- **import_coredefs** (*bool*) –

Methods

add_fields	
check_alignment	Confirm 64 bit alignment of structures
check_duplicate_name	Check namespaces for conflicting names.
check_key_value_separation	
check_name	Check that names start with a letter.
clear	
expand_expression	**rtype** Tuple[str, int]
get_ctype_cls	**rtype** Type[Structure]
get_ctype_size	**rtype** int
handle_alias	Find the base type ultimately represented by the typedef alias
handle_compiler_options	
handle_expression	

continues on next page

Table 1 – continued from previous page

handle_host_id	
handle_import	
handle_message_def	
handle_metadata	
handle_module_id	
handle_reserve	
handle_signal	
handle_string	
handle_struct	
parse	
parse_compiler_options	**rtype** Dict[str, *CompilerOptions*]
parse_file	
parse_options	
parse_options_text	
parse_text	
to_json	
trim_root	**rtype** Path
validate_msg_def	
validate_msg_id	
warning	

check_alignment(*s*)

Confirm 64 bit alignment of structures

Parameters
s (*Union[SDF, MDF]*) –

check_duplicate_name(*section*, *name*, *namespaces*)

Check namespaces for conflicting names.

Parameters

- **section** (*str*) –
- **name** (*str*) –
- **namespaces** (*Tuple[str, ...]*) –

check_name(*name*)

Check that names start with a letter.

Parameters

name (*str*) –

handle_alias(*alias*, *ftype*)

Find the base type ultimately represented by the typedef alias

Parameters

- **alias** (*str*) –
- **ftype** (*str*) –

18.9 pyrtma.compile.PyDefCompiler

class PyDefCompiler(*parser*, *debug=False*)

Bases: object

Methods

generate	
generate_constant	**rtype** str
generate_context	
generate_docstr	**rtype** str
generate_host_id	**rtype** str
generate_imports	**rtype** str
generate_module_id	**rtype** str
generate_msg_def	**rtype** str
generate_msg_type_id	**rtype** str
generate_string_constant	**rtype** str
generate_struct	**rtype** str
generate_type_alias	**rtype** str
get_descriptor	**rtype** str

Parameters

- **parser** (Parser) –
- **debug** (*bool*) –

18.10 pyrtma.compile.YAMLCompiler

class YAMLCompiler(*parser*, *filename*, *debug=False*)

Bases: object

Methods

generate

Parameters

- **parser** (Parser) –
- **filename** (*str*) –
- **debug** (*bool*) –

Exceptions

FileFormatError	Raised when the wrong file extension is referenced.
ParserError	Base class for all parser exceptions

18.11 pyrtma.compile.FileFormatError

exception FileFormatError

Raised when the wrong file extension is referenced.

18.12 pyrtma.compile.ParserError

exception ParserError

Base class for all parser exceptions

CHAPTER

NINETEEN

PYRTMA.CONSTANTS

pyrtma.constants module

This module is deprecated and included for backwards compatilibity with pyrtma 1.2.3 Constants have been moved to pyrtma.core_defs

Functions

warn	Issue a warning, or maybe ignore it or raise an exception.

19.1 pyrtma.constants.warn

warn(*message*, *category=None*, *stacklevel=1*, *source=None*)

Issue a warning, or maybe ignore it or raise an exception.

Classes

HOST_ID	alias of c_short
MODULE_ID	alias of c_short
MSG_COUNT	alias of c_int
MSG_TYPE	alias of c_int

19.2 pyrtma.constants.HOST_ID

HOST_ID

alias of c_short

19.3 pyrtma.constants.MODULE_ID

MODULE_ID

alias of c_short

19.4 pyrtma.constants.MSG_COUNT

MSG_COUNT

alias of c_int

19.5 pyrtma.constants.MSG_TYPE

MSG_TYPE

alias of c_int

CHAPTER

TWENTY

PYRTMA.CORE_DEFS

This message def file was auto-generated by pyrtma.compile version 2.2.4

Functions

check_compiled_version	
get_context	**rtype** Dict[str, Dict[str, Any]]

20.1 pyrtma.core_defs.check_compiled_version

check_compiled_version(*compiled_version*)

Parameters
compiled_version (*str*) –

20.2 pyrtma.core_defs.get_context

get_context()

Return type
Dict[str, Dict[str, Any]]

Classes

Byte	Validator for single byte values
ByteArray	Validator class for Bytes arrays
Char	Validator for scalar char values
Double	Double (64-bit float) validator class
Float	32-bit Float validator class
FloatArray	Validator class for float arrays

continues on next page

Table 1 – continued from previous page

HOST_ID	alias of c_short
Int16	Validator for 16-bit integers
Int32	Validator for 32-bit integers
Int64	Validator for 64-bit integers
Int8	Validator for 8-bit integers
IntArray	IntArray validator class
MDF_ACKNOWLEDGE	
MDF_CONNECT	
MDF_DISCONNECT	
MDF_DUMP_MESSAGE_LOG	
MDF_EXIT	
MDF_FAILED_MESSAGE	
MDF_FAIL_SUBSCRIBE	
MDF_FORCE_DISCONNECT	
MDF_KILL	
MDF_LM_EXIT	
MDF_LM_READY	
MDF_MESSAGE_LOG_SAVED	
MDF_MODULE_READY	
MDF_PAUSE_MESSAGE_LOGGING	
MDF_PAUSE_SUBSCRIPTION	
MDF_RESET_MESSAGE_LOG	
MDF_RESUME_MESSAGE_LOGGING	
MDF_RESUME_SUBSCRIPTION	
MDF_SAVE_MESSAGE_LOG	
MDF_SUBSCRIBE	
MDF_TIMING_MESSAGE	
MDF_UNSUBSCRIBE	
MODULE_ID	alias of c_short

continues on next page

Table 1 – continued from previous page

MSG_COUNT	alias of c_int
MSG_TYPE	alias of c_int
MessageBase	MessageBase base class
MessageData	MessageData base class
MessageMeta	MessageMeta metaclass
RTMA_MSG_HEADER	
String	Validator for strings (char arrays)
Struct	Validator class for Structures
StructArray	Validator for structure arrays
Uint16	Validator for unsigned 16-bit integers
Uint32	Validator for unsigned 32-bit integers
Uint64	Validator for unsigned 64-bit integers
Uint8	Validator for unsigned 8-bit integers

20.3 pyrtma.core_defs.Byte

class Byte(**args*)

Bases: *FieldValidator*[_P, int], *Generic*[_P]

Validator for single byte values

Methods

validate_many	Validate multiple byte values
validate_one	validate a single byte value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple byte values

Parameters

value (*Union[Iterable[int], bytes, bytearray]*) – Byte values to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Value out of range

validate_one(*value*)

validate a single byte value

Parameters
value (*Union[int, bytes, bytearray]*) – Byte value to validate

Raises

- **ValueError** – Value out of range
- **TypeError** – Wrong type

20.4 pyrtma.core_defs.ByteArray

class ByteArray(*len*)

Bases: *ArrayField*[*Byte*]

Validator class for Bytes arrays

Validator class for Bytes arrays

Parameters
len (*int*) – Byte array length

Methods

count

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters
value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

20.5 pyrtma.core_defs.Char

class Char

Bases: *String*

Validator for scalar char values

Methods

validate_many	Validate multiple strings
validate_one	Validate a string value

validate_many(*value*)

Validate multiple strings

Not implemented

Raises
NotImplementedError –

validate_one(*value*)

Validate a string value

Parameters
value (*str*) – String value

Raises

- **TypeError** – Wrong type
- **ValueError** – String exceeds max length

20.6 pyrtma.core_defs.Double

class Double(**args*)

Bases: *FloatValidatorBase*

Double (64-bit float) validator class

Methods

validate_many	Validate multiple float values
validate_one	Validate a float value

validate_many(*value*)

Validate multiple float values

Parameters
value (*Iterable[float]*) – Iterable of floats to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

validate_one(*value*)

Validate a float value

Parameters
value (*float*) – Float value

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

20.7 pyrtma.core_defs.Float

class Float(**args*)

Bases: *FloatValidatorBase*

32-bit Float validator class

Methods

validate_many	Validate multiple float values
validate_one	Validate a float value

validate_many(*value*)

Validate multiple float values

Parameters
value (*Iterable[float]*) – Iterable of floats to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

validate_one(*value*)

Validate a float value

Parameters
value (*float*) – Float value

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

20.8 pyrtma.core_defs.FloatArray

class FloatArray(*validator*, *len*)

Bases: *ArrayField*[_FPV], *Generic*[_FPV]

Validator class for float arrays

Validator class for float arrays

Parameters

- **validator** (*Type[_FPV]*) – Float type validator
- **len** (*int*) – Array length

Methods

count	

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters
value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

20.9 pyrtma.core_defs.HOST_ID

HOST_ID

alias of c_short

20.10 pyrtma.core_defs.Int16

class Int16(**args*)

Bases: *IntValidatorBase*

Validator for 16-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.11 pyrtma.core_defs.Int32

class Int32(**args*)

Bases: *IntValidatorBase*

Validator for 32-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.12 pyrtma.core_defs.Int64

class Int64(**args*)

Bases: *IntValidatorBase*

Validator for 64-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.13 pyrtma.core_defs.Int8

class Int8(**args*)

Bases: *IntValidatorBase*

Validator for 8-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.14 pyrtma.core_defs.IntArray

class IntArray(*validator*, *len*)

Bases: *ArrayField*[_IV], *Generic*[_IV]

IntArray validator class

IntArray validator class

Parameters

- **validator** (*Type[IV]*) – Field validator class for Int type
- **len** (*int*) – Field length

Methods

count	

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters
value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

20.15 pyrtma.core_defs.MDF_ACKNOWLEDGE

class MDF_ACKNOWLEDGE

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.16 pyrtma.core_defs.MDF_CONNECT

class MDF_CONNECT

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

daemon_status	Validator for 16-bit integers
logger_status	Validator for 16-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.17 pyrtma.core_defs.MDF_DISCONNECT

class MDF_DISCONNECT

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json`(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.18 pyrtma.core_defs.MDF_DUMP_MESSAGE_LOG

`class MDF_DUMP_MESSAGE_LOG`

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.19 pyrtma.core_defs.MDF_EXIT

class MDF_EXIT

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.20 pyrtma.core_defs.MDF_FAILED_MESSAGE

class MDF_FAILED_MESSAGE

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_mod_id	Validator for 16-bit integers
msg_header	Validator class for Structures
reserved	IntArray validator class
size	
time_of_failure	Double (64-bit float) validator class
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.21 pyrtma.core_defs.MDF_FAIL_SUBSCRIBE

class MDF_FAIL_SUBSCRIBE

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

mod_id	Validator for 16-bit integers
msg_type	Validator for 32-bit integers
reserved	Validator for 16-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.22 pyrtma.core_defs.MDF_FORCE_DISCONNECT

class MDF_FORCE_DISCONNECT

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

mod_id	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.23 pyrtma.core_defs.MDF_KILL

class MDF_KILL

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json`(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.24 pyrtma.core_defs.MDF_LM_EXIT

`class MDF_LM_EXIT`

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.25 pyrtma.core_defs.MDF_LM_READY

class MDF_LM_READY

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json`(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.26 pyrtma.core_defs.MDF_MESSAGE_LOG_SAVED

`class MDF_MESSAGE_LOG_SAVED`

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.27 pyrtma.core_defs.MDF_MODULE_READY

class MDF_MODULE_READY

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

pid	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.28 pyrtma.core_defs.MDF_PAUSE_MESSAGE_LOGGING

class MDF_PAUSE_MESSAGE_LOGGING

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.29 pyrtma.core_defs.MDF_PAUSE_SUBSCRIPTION

class MDF_PAUSE_SUBSCRIPTION

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

msg_type	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json`(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.30 pyrtma.core_defs.MDF_RESET_MESSAGE_LOG

`class MDF_RESET_MESSAGE_LOG`

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.31 pyrtma.core_defs.MDF_RESUME_MESSAGE_LOGGING

class MDF_RESUME_MESSAGE_LOGGING

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_hash
type_id
type_name
type_size
type_source

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json`(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.32 pyrtma.core_defs.MDF_RESUME_SUBSCRIPTION

`class MDF_RESUME_SUBSCRIPTION`

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

msg_type	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.33 pyrtma.core_defs.MDF_SAVE_MESSAGE_LOG

class MDF_SAVE_MESSAGE_LOG

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

pathname	Validator for strings (char arrays)
pathname_length	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.34 pyrtma.core_defs.MDF_SUBSCRIBE

class MDF_SUBSCRIBE

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

msg_type	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.35 pyrtma.core_defs.MDF_TIMING_MESSAGE

class MDF_TIMING_MESSAGE

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

ModulePID	IntArray validator class
send_time	Double (64-bit float) validator class
size	
timing	IntArray validator class
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.36 pyrtma.core_defs.MDF_UNSUBSCRIBE

class MDF_UNSUBSCRIBE

Bases: *MessageData*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

msg_type	Validator for 32-bit integers
size	
type_def	
type_hash	
type_id	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.37 pyrtma.core_defs.MODULE_ID

MODULE_ID

alias of c_short

20.38 pyrtma.core_defs.MSG_COUNT

MSG_COUNT

alias of c_int

20.39 pyrtma.core_defs.MSG_TYPE

MSG_TYPE

alias of c_int

20.40 pyrtma.core_defs.MessageBase

class MessageBase

Bases: Structure

MessageBase base class

This class should be treated as if abstract and not instantiated directly.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.41 pyrtma.core_defs.MessageData

class MessageData

Bases: *MessageBase*

MessageData base class

This is intended to be treated as an abstract class and and should not be directly instantiated.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_id
type_name
type_size
type_source
type_hash

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.

- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.42 pyrtma.core_defs.MessageMeta

class MessageMeta(*name*, *bases*, *namespace*)

Bases: PyCStructType

MessageMeta metaclass

Responsible for generating ctypes fields from descriptor attributes prior to class creation

Methods

from_address	access a C instance at the specified address
from_buffer	create a C instance from a writeable buffer
from_buffer_copy	create a C instance from a readable buffer
from_param	Convert a Python object into a function call parameter.
in_dll	access a C instance in a dll
mro	Return a type's method resolution order.

__call__(**args*, ***kwargs*)
Call self as a function.

__mul__(*value*, */*)
Return self*value.

from_address(*integer*) → C instance
access a C instance at the specified address

from_buffer(*object*, *offset=0*) → C instance
create a C instance from a writeable buffer

from_buffer_copy(*object*, *offset=0*) → C instance
create a C instance from a readable buffer

from_param()
Convert a Python object into a function call parameter.

in_dll(*dll*, *name*) → C instance
access a C instance in a dll

mro()
Return a type's method resolution order.

20.43 pyrtma.core_defs.RTMA_MSG_HEADER

class RTMA_MSG_HEADER

Bases: *MessageBase*

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	
src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
type_def	
type_hash	
type_name	
type_size	
type_source	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

`to_dict()`

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json(minify=False, **kwargs)`

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

20.44 pyrtma.core_defs.String

`class String(len)`

Bases: *FieldValidator*[_P, str], *Generic*[_P]

Validator for strings (char arrays)

Methods

validate_many	Validate multiple strings
validate_one	Validate a string value

`validate_many(value)`

Validate multiple strings

Not implemented

Raises
NotImplementedError –

`validate_one(value)`

Validate a string value

Parameters
value (*str*) – String value

Raises

- **TypeError** – Wrong type
- **ValueError** – String exceeds max length

20.45 pyrtma.core_defs.Struct

class Struct(*_ctype*)

Bases: *FieldValidator*, *Generic*[_S]

Validator class for Structures

Methods

validate_many	Validate multiple structures
validate_one	Validate a structure

validate_many(*value*)

Validate multiple structures

Parameters
value (*Iterable[_S]*) – Iterable of structures to validate

Raises
TypeError – Wrong type

validate_one(*value*)

Validate a structure

Parameters
value (*_S*) – Structure value to validate

Raises
TypeError – Wrong type

20.46 pyrtma.core_defs.StructArray

class StructArray(*msg_struct*, *len*)

Bases: *FieldValidator*, Sequence, *Generic*[_S]

Validator for structure arrays

Validator for structure arrays

Parameters

- **msg_struct** (*Type[_S]*) – Structure class
- **len** (*int*) – Array length

Methods

count	

index	Raises ValueError if the value is not present.
pretty_print	Generate formatted string for structure array
validate_array	Validate structure array
validate_many	Validate multiple structures
validate_one	Validate a structure

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

pretty_print(*add_tabs=0*)

Generate formatted string for structure array

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

validate_array(*value*)

Validate structure array

Parameters
value (*StructArray[_S]*) – StructArray to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple structures

Parameters
value (*Iterable[_S]*) – Structure values to validate

validate_one(*value*)

Validate a structure

Parameters
value (*_S*) – Structure value to validate

20.47 pyrtma.core_defs.Uint16

class Uint16(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 16-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.48 pyrtma.core_defs.Uint32

class Uint32(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 32-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.49 pyrtma.core_defs.Uint64

class Uint64(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 64-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

20.50 pyrtma.core_defs.Uint8

class Uint8(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 8-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

CHAPTER

TWENTYONE

PYRTMA.EXCEPTIONS

Exceptions

AcknowledgementTimeout	Raised when client does not receive ack from message manager.
ClientError	Base exception for all Client Errors.
ConnectionLost	Raised when there is a connection error with the server.
InvalidDestinationHost	Raised when client tries to send to an invalid host.
InvalidDestinationModule	Raised when client tries to send to an invalid module.
InvalidMessageDefinition	Raised when there is message definition is out of sync with sent data.
JSONDecodingError	Raised when there is an error decoding a message from json.
MessageManagerNotFound	Raised when unable to connect to message manager.
NotConnectedError	Raised when the client tries to read/write while not connected.
RTMAMessageError	Base exception for message errors.
SocketOptionError	Raised when unable to set socket options.
UnknownMessageType	Raised when there is no message definition.
VersionMismatchWarning	Raised when import message defs were compiled with a different pyrtma version

21.1 pyrtma.exceptions.AcknowledgementTimeout

exception AcknowledgementTimeout

Raised when client does not receive ack from message manager.

21.2 pyrtma.exceptions.ClientError

exception ClientError

Base exception for all Client Errors.

21.3 pyrtma.exceptions.ConnectionLost

exception ConnectionLost

Raised when there is a connection error with the server.

21.4 pyrtma.exceptions.InvalidDestinationHost

exception InvalidDestinationHost

Raised when client tries to send to an invalid host.

21.5 pyrtma.exceptions.InvalidDestinationModule

exception InvalidDestinationModule

Raised when client tries to send to an invalid module.

21.6 pyrtma.exceptions.InvalidMessageDefinition

exception InvalidMessageDefinition

Raised when there is message definition is out of sync with sent data.

21.7 pyrtma.exceptions.JSONDecodingError

exception JSONDecodingError

Raised when there is an error decoding a message from json.

21.8 pyrtma.exceptions.MessageManagerNotFound

exception MessageManagerNotFound

Raised when unable to connect to message manager.

21.9 pyrtma.exceptions.NotConnectedError

exception NotConnectedError

Raised when the client tries to read/write while not connected.

21.10 pyrtma.exceptions.RTMAMessageError

exception RTMAMessageError

Base exception for message errors.

21.11 pyrtma.exceptions.SocketOptionError

exception SocketOptionError

Raised when unable to set socket options.

21.12 pyrtma.exceptions.UnknownMessageType

exception UnknownMessageType

Raised when there is no message definition.

21.13 pyrtma.exceptions.VersionMismatchWarning

exception VersionMismatchWarning

Raised when import message defs were compiled with a different pyrtma version

CHAPTER

TWENTYTWO

PYRTMA.HEADER

Functions

get_header_cls	Get the correct header class depending on whether timecode is used

22.1 pyrtma.header.get_header_cls

get_header_cls(*timecode=False*)

Get the correct header class depending on whether timecode is used

Parameters
timecode (*bool, optional*) – Flag indicating if timecode fields are needed. Defaults to False.

Returns
MessageHeader class

Return type
Type[*MessageHeader*]

Classes

Double	Double (64-bit float) validator class
HOST_ID	alias of c_short
Int16	Validator for 16-bit integers
Int32	Validator for 32-bit integers
MODULE_ID	alias of c_short
MSG_COUNT	alias of c_int
MSG_TYPE	alias of c_int
MessageBase	MessageBase base class
MessageHeader	RTMA Message Header class
MessageMeta	MessageMeta metaclass
TimeCodeMessageHeader	Variant of MessageHeader with additional Timecode fields
Uint32	Validator for unsigned 32-bit integers

22.2 pyrtma.header.Double

class Double(**args*)

Bases: *FloatValidatorBase*

Double (64-bit float) validator class

Methods

validate_many	Validate multiple float values
validate_one	Validate a float value

validate_many(*value*)

Validate multiple float values

Parameters
value (*Iterable[float]*) – Iterable of floats to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

validate_one(*value*)

Validate a float value

Parameters
value (*float*) – Float value

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

22.3 pyrtma.header.HOST_ID

HOST_ID

alias of c_short

22.4 pyrtma.header.Int16

class Int16(**args*)

Bases: *IntValidatorBase*

Validator for 16-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

22.5 pyrtma.header.Int32

class Int32(**args*)

Bases: *IntValidatorBase*

Validator for 32-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

22.6 pyrtma.header.MODULE_ID

MODULE_ID

alias of c_short

22.7 pyrtma.header.MSG_COUNT

MSG_COUNT

alias of c_int

22.8 pyrtma.header.MSG_TYPE

MSG_TYPE

alias of c_int

22.9 pyrtma.header.MessageBase

class MessageBase

Bases: Structure

MessageBase base class

This class should be treated as if abstract and not instantiated directly.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size

classmethod copy(*m*)

Generate a copy of a message structure

Parameters

m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type

TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

`to_dict()`

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json(minify=False, **kwargs)`

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

22.10 pyrtma.header.MessageHeader

`class MessageHeader`

Bases: *MessageBase*

RTMA Message Header class

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	

src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

22.11 pyrtma.header.MessageMeta

class MessageMeta(*name*, *bases*, *namespace*)

Bases: PyCStructType

MessageMeta metaclass

Responsible for generating ctypes fields from descriptor attributes prior to class creation

Methods

from_address	access a C instance at the specified address
from_buffer	create a C instance from a writeable buffer
from_buffer_copy	create a C instance from a readable buffer
from_param	Convert a Python object into a function call parameter.
in_dll	access a C instance in a dll
mro	Return a type's method resolution order.

__call__(**args*, ***kwargs*)

Call self as a function.

__mul__(*value*, */*)

Return self*value.

from_address(*integer*) → C instance

access a C instance at the specified address

from_buffer(*object*, *offset=0*) → C instance

create a C instance from a writeable buffer

from_buffer_copy(*object*, *offset=0*) → C instance

create a C instance from a readable buffer

from_param()

Convert a Python object into a function call parameter.

in_dll(*dll*, *name*) → C instance

access a C instance in a dll

mro()

Return a type's method resolution order.

22.12 pyrtma.header.TimeCodeMessageHeader

class TimeCodeMessageHeader

Bases: *MessageHeader*

Variant of MessageHeader with additional Timecode fields

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size

utc_fraction	Validator for unsigned 32-bit integers
utc_seconds	Validator for unsigned 32-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

22.13 pyrtma.header.Uint32

class Uint32(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 32-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

CHAPTER

TWENTYTHREE

PYRTMA.MANAGER

pyrtma.manager module

Contains *MessageManager* class

Functions

dataclass	Returns the same class as was passed in, with dunder methods added based on the fields defined in the class.
get_header_cls	Get the correct header class depending on whether timecode is used
get_msg_cls	get msg class for a given message type ID
main	

23.1 pyrtma.manager.dataclass

dataclass(*cls=None*, */*, ***, *init=True*, *repr=True*, *eq=True*, *order=False*, *unsafe_hash=False*, *frozen=False*)

Returns the same class as was passed in, with dunder methods added based on the fields defined in the class.

Examines PEP 526 __annotations__ to determine fields.

If init is true, an __init__() method is added to the class. If repr is true, a __repr__() method is added. If order is true, rich comparison dunder methods are added. If unsafe_hash is true, a __hash__() method function is added. If frozen is true, fields may not be assigned to after instance creation.

23.2 pyrtma.manager.get_header_cls

get_header_cls(*timecode=False*)

Get the correct header class depending on whether timecode is used

Parameters
timecode (*bool, optional*) – Flag indicating if timecode fields are needed. Defaults to False.

Returns
MessageHeader class

Return type
Type[*MessageHeader*]

23.3 pyrtma.manager.get_msg_cls

get_msg_cls(*id*)

get msg class for a given message type ID

Parameters
id (*int*) – Message Type ID

Raises
UnknownMessageType – Message type is undefined

Returns
Message class

Return type
Type[*MessageData*]

23.4 pyrtma.manager.main

main()

Classes

Counter	Dict subclass for counting hashable items.
Message	Message class
MessageData	MessageData base class
MessageHeader	RTMA Message Header class
MessageManager	MessageManager class
Module	Module dataclass
defaultdict	defaultdict(default_factory[, ...]) --> dict with default factory

23.5 pyrtma.manager.Counter

class Counter(*iterable=None*, */*, ***kwds*)

Bases: dict

Dict subclass for counting hashable items. Sometimes called a bag or multiset. Elements are stored as dictionary keys and their counts are stored as dictionary values.

```
>>> c = Counter('abcdeabcdabcaba')  # count elements from a string
```

```
>>> c.most_common(3)                # three most common elements
[('a', 5), ('b', 4), ('c', 3)]
>>> sorted(c)                       # list all unique elements
['a', 'b', 'c', 'd', 'e']
>>> ''.join(sorted(c.elements()))   # list elements with repetitions
'aaaaabbbbcccdde'
```

(continues on next page)

(continued from previous page)

```
>>> sum(c.values())                 # total of all counts
15
```

```
>>> c['a']                          # count of letter 'a'
5
>>> for elem in 'shazam':           # update counts from an iterable
...     c[elem] += 1                # by adding 1 to each element's count
>>> c['a']                          # now there are seven 'a'
7
>>> del c['b']                      # remove all 'b'
>>> c['b']                          # now there are zero 'b'
0
```

```
>>> d = Counter('simsalabim')       # make another counter
>>> c.update(d)                     # add in the second counter
>>> c['a']                          # now there are nine 'a'
9
```

```
>>> c.clear()                       # empty the counter
>>> c
Counter()
```

Note: If a count is set to zero or reduced to zero, it will remain in the counter until the entry is deleted or the counter is cleared:

```
>>> c = Counter('aaabbc')
>>> c['b'] -= 2                     # reduce the count of 'b' by two
>>> c.most_common()                 # 'b' is still in, but its count is zero
[('a', 3), ('c', 1), ('b', 0)]
```

Create a new, empty Counter object. And if given, count elements from an input iterable. Or, initialize the count from another mapping of elements to their counts.

```
>>> c = Counter()                           # a new, empty counter
>>> c = Counter('gallahad')                 # a new counter from an iterable
>>> c = Counter({'a': 4, 'b': 2})           # a new counter from a mapping
>>> c = Counter(a=4, b=2)                   # a new counter from keyword args
```

Methods

clear	
copy	Return a shallow copy.
elements	Iterator over elements repeating each as many times as its count.
fromkeys	Create a new dictionary with keys from iterable and values set to value.
get	Return the value for key if key is in the dictionary, else default.
items	
keys	
most_common	List the n most common elements and their counts from the most common to the least.
pop	If key is not found, d is returned if given, otherwise KeyError is raised
popitem	Remove and return a (key, value) pair as a 2-tuple.
setdefault	Insert key with a value of default if key is not in the dictionary.
subtract	Like dict.update() but subtracts counts instead of replacing them.
update	Like dict.update() but add counts instead of replacing them.
values	

__add__(*other*)

Add counts from two counters.

```
>>> Counter('abbb') + Counter('bcc')
Counter({'b': 4, 'c': 2, 'a': 1})
```

clear() → None. Remove all items from D.

copy()

Return a shallow copy.

elements()

Iterator over elements repeating each as many times as its count.

```
>>> c = Counter('ABCABC')
>>> sorted(c.elements())
['A', 'A', 'B', 'B', 'C', 'C']
```

Knuth's example for prime factors of 1836: 2**2 * 3**3 * 17**1 >>> prime_factors = Counter({2: 2, 3: 3, 17: 1}) >>> product = 1 >>> for factor in prime_factors.elements(): # loop over factors ... product *= factor # and multiply them >>> product 1836

Note, if an element's count has been set to zero or is a negative number, elements() will ignore it.

classmethod fromkeys(*iterable*, *v=None*)

Create a new dictionary with keys from iterable and values set to value.

get(*key*, *default=None*, */*)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

most_common(*n=None*)

List the n most common elements and their counts from the most common to the least. If n is None, then list all element counts.

```
>>> Counter('abracadabra').most_common(3)
[('a', 5), ('b', 2), ('r', 2)]
```

pop(*k*[, *d*]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

setdefault(*key*, *default=None*, */*)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

subtract(*iterable=None*, */*, ***kwds*)

Like dict.update() but subtracts counts instead of replacing them. Counts can be reduced below zero. Both the inputs and outputs are allowed to contain zero and negative counts.

Source can be an iterable, a dictionary, or another Counter instance.

```
>>> c = Counter('which')
>>> c.subtract('witch')             # subtract elements from another iterable
>>> c.subtract(Counter('watch'))    # subtract elements from another counter
>>> c['h']                          # 2 in which, minus 1 in witch, minus 1 in␣
↪watch
0
>>> c['w']                          # 1 in which, minus 1 in witch, minus 1 in␣
↪watch
-1
```

update(*iterable=None*, */*, ***kwds*)

Like dict.update() but add counts instead of replacing them.

Source can be an iterable, a dictionary, or another Counter instance.

```
>>> c = Counter('which')
>>> c.update('witch')           # add elements from another iterable
>>> d = Counter('watch')
>>> c.update(d)                 # add elements from another counter
>>> c['h']                      # four 'h' in which, witch, and watch
4
```

values() → an object providing a view on D's values

23.6 pyrtma.manager.Message

class Message(*header*, *data*)

Bases: object

Message class

Contains message header and data

Methods

copy	Generate a copy of a message structure
from_json	Create message object from JSON string
pretty_print	Generate formatted string for pretty printing of message
to_dict	Convert message to dictionary
to_json	Convert message to JSON string

Attributes

name
type_id

Parameters

- **header** (MessageHeader) –
- **data** (MessageData) –

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*Message*) – Message structure to copy

Return type
Message

classmethod from_json(*s*)

Create message object from JSON string

Parameters
s (*str*) – JSON message string

Raises
InvalidMessageDefinition – JSON data does not match expected message defintion

Returns
Message object

Return type
Message

`pretty_print`(*add_tabs=0*)

Generate formatted string for pretty printing of message

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

`to_dict`()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json`(*minify=False*, ***kwargs*)

Convert message to JSON string

Parameters
minify (*bool, optional*) – Flag to minify (compact format). Defaults to False.

Returns
JSON message string

Return type
str

23.7 pyrtma.manager.MessageData

`class MessageData`

Bases: *MessageBase*

MessageData base class

This is intended to be treated as an abstract class and and should not be directly instantiated.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_id
type_name
type_size
type_source
type_hash

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.

- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

23.8 pyrtma.manager.MessageHeader

class MessageHeader

Bases: *MessageBase*

RTMA Message Header class

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	
src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

`to_dict()`
Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json(minify=False, **kwargs)`
Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

23.9 pyrtma.manager.MessageManager

`class MessageManager(ip_address='', port=7111, timecode=False, debug=False, send_msg_timing=True)`
Bases: object

MessageManager class

RTMA Message Manager server implemented in python.

MessageManager class

RTMA Message Manager server implemented in python.

Parameters

- **ip_address** (*str, optional*) – server IP address. Defaults to "".
- **timecode** (*bool, optional*) – Flag to use message header with timecode values. Defaults to False.
- **debug** (*bool, optional*) – Flag for debug mode. Defaults to False.
- **send_msg_timing** (*bool, optional*) – Flag to send TIMING_MSG. Defaults to True.
- **port** (*int*) –

Methods

add_subscription	Add message subscription
assign_module_id	Assign module ID dynamically to connecting module
close	Close manager server
connect_module	Connect module
disconnect_module	Disconnect module
forward_message	Forward a message from other modules
pause_subscription	Pause message subscription
process_message	Process incoming message
read_message	Read an incoming message
register_module_ready	Handle MODULE_READY message and register PID
remove_module	Remove connected module
remove_subscription	Remove message subscription
resume_subscription	Resume message subscription
run	Start the message manager server
send_ack	Send ACKNOWLEDGE signal header
send_failed_message	Send FAILED_MESSAGE
send_timing_message	Send TIMING_MESSAGE
send_to_loggers	Forward message to registered logger modules

Attributes

header
message

add_subscription(*src_module*, *msg*)

Add message subscription

Parameters

- **src_module** (Module) – Subscribing module
- **msg** (Message) – incoming SUBSCRIBE message

assign_module_id()

Assign module ID dynamically to connecting module

Raises
RuntimeError – Exceeded maximum number of allowed dynamic modules

Returns
module ID

Return type
int

close()

Close manager server

connect_module(*src_module*, *msg*)

Connect module

Parameters

- **src_module** (Module) – Connecting module
- **msg** (Message) – Incoming connect message

Returns
success code

Return type
bool

disconnect_module(*src_module*)

Disconnect module

Parameters
src_module (Module) – Module object to disconnect

forward_message(*header*, *data*, *wlist*)

Forward a message from other modules

The given message will be forwarded to:

- all subscribed logger modules (ALWAYS)
- if the message has a destination address, and it is subscribed to by that destination it will be forwarded only there
- if the message has no destination address, it will be forwarded to all subscribed modules

Parameters

- **header** (MessageHeader) – Message Header
- **data** (*Union[bytes, MessageData]*) – Message Data
- **wlist** (*List[socket.socket]*) – sockets ready for writing

pause_subscription(*src_module*, *msg*)

Pause message subscription

Parameters

- **src_module** (Module) – Subscribing module
- **msg** (Message) – incoming PAUSE_SUBSCRIPTION message

process_message(*src_module*, *wlist*)

Process incoming message

Parameters

- **src_module** (Module) – Message source module
- **wlist** (*List[socket.socket]*) – Sockets ready for writing

read_message(*sock*)

Read an incoming message

Parameters
sock (*socket.socket*) – socket to read from

Returns
Success code

Return type
bool

register_module_ready(*src_module*, *msg*)
Handle MODULE_READY message and register PID

Parameters

- **src_module** (Module) – Module that is ready
- **msg** (Message) – Incoming MODULE_READY message

remove_module(*module*)
Remove connected module

Parameters
module (Module) – Module object to remove

remove_subscription(*src_module*, *msg*)
Remove message subscription

Parameters

- **src_module** (Module) – Unsubscribing module
- **msg** (Message) – incoming UNSUBSCRIBE message

resume_subscription(*src_module*, *msg*)
Resume message subscription

Parameters

- **src_module** (Module) – Subscribing module
- **msg** (Message) – incoming RESUME_SUBSCRIPTION message

run()
Start the message manager server

send_ack(*src_module*, *wlist*)
Send ACKNOWLEDGE signal header

Parameters

- **src_module** (Module) – Module to send ACK to
- **wlist** (*List[socket.socket]*) – Sockets ready for writing

send_failed_message(*dest_module*, *header*, *time_of_failure*, *wlist*)
Send FAILED_MESSAGE

Parameters

- **dest_module** (Module) – Intended destination
- **header** (MessageHeader) – Header of failed message
- **time_of_failure** (*float*) – Time of send failure
- **wlist** (*List[socket.socket]*) – Sockets ready for writing

send_timing_message(*wlist*)

Send TIMING_MESSAGE

Parameters

wlist (*List[socket.socket]*) – Sockets ready for writing

send_to_loggers(*header*, *payload*, *wlist*)

Forward message to registered logger modules

Parameters

- **header** (MessageHeader) – Message header to send
- **payload** (*Union[bytes, MessageData]*) – Message data to send
- **wlist** (*List[socket.socket]*) – Sockets ready for writing

23.10 pyrtma.manager.Module

class Module(*conn*, *address*, *header_cls*, *id=0*, *pid=0*, *connected=False*, *is_logger=False*)

Bases: object

Module dataclass

Used internally by MessageManager to manage connections to each client module.

Methods

close	Close connection
send_ack	Send ACKNOWLEDGE signal header
send_message	Send a message

Attributes

connected
id
is_logger
pid
conn
address
header_cls

Parameters

- **conn** (*socket*) –

- **address** (*Tuple[str, int]*) –
- **header_cls** (*Type[MessageHeader]*) –
- **id** (*int*) –
- **pid** (*int*) –
- **connected** (*bool*) –
- **is_logger** (*bool*) –

close()

Close connection

send_ack()

Send ACKNOWLEDGE signal header

send_message(*header*, *payload*)

Send a message

Parameters

- **header** (MessageHeader) – Message header
- **payload** (*Union[bytes, MessageData]*) – Message data

23.11 pyrtma.manager.defaultdict

class defaultdict

Bases: dict

defaultdict(default_factory[, ...]) –> dict with default factory

The default factory is called without arguments to produce a new value when a key is not present, in __getitem__ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

Methods

clear	
copy	
fromkeys	Create a new dictionary with keys from iterable and values set to value.
get	Return the value for key if key is in the dictionary, else default.
items	
keys	
pop	If key is not found, d is returned if given, otherwise KeyError is raised
popitem	Remove and return a (key, value) pair as a 2-tuple.
setdefault	Insert key with a value of default if key is not in the dictionary.
update	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
values	

Attributes

default_factory	Factory for default value called by __missing__().

clear() → None. Remove all items from D.

copy() → a shallow copy of D.

default_factory

Factory for default value called by __missing__().

fromkeys(*value=None*, */*)

Create a new dictionary with keys from iterable and values set to value.

get(*key*, *default=None*, */*)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

pop(*k*[, *d*]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

setdefault(*key*, *default=None*, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update(*[E]*, ***F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values() → an object providing a view on D's values

CHAPTER

TWENTYFOUR

PYRTMA.MESSAGE

pyrtma.messaage: RTMA message classes

Functions

clear_msg_defs	
get_header_cls	Get the correct header class depending on whether timecode is used
get_msg_cls	get msg class for a given message type ID
get_msg_defs	**rtype** Dict[int, Type[*MessageData*]]
message_def	Decorator to add user message definitions.
msg_def	Decorator to add user message definitions.
set_msg_defs	
update_msg_defs	

24.1 pyrtma.message.clear_msg_defs

clear_msg_defs()

24.2 pyrtma.message.get_header_cls

get_header_cls(*timecode=False*)

Get the correct header class depending on whether timecode is used

Parameters
timecode (*bool, optional*) – Flag indicating if timecode fields are needed. Defaults to False.

Returns
MessageHeader class

Return type
Type[*MessageHeader*]

24.3 pyrtma.message.get_msg_cls

get_msg_cls(*id*)

get msg class for a given message type ID

Parameters
id (*int*) – Message Type ID

Raises
UnknownMessageType – Message type is undefined

Returns
Message class

Return type
Type[*MessageData*]

24.4 pyrtma.message.get_msg_defs

get_msg_defs()

Return type
Dict[int, Type[*MessageData*]]

24.5 pyrtma.message.message_def

message_def(*msg_cls*, **args*, ***kwargs*)

Decorator to add user message definitions.

Return type
Type[*TypeVar*(_MD, bound= *MessageData*)]

Parameters
msg_cls (*Type[_MD]*) –

24.6 pyrtma.message.msg_def

msg_def(*msg_cls*, **args*, ***kwargs*)

Decorator to add user message definitions.

Return type
Type[*TypeVar*(_MD, bound= *MessageData*)]

Parameters
msg_cls (*Type[_MD]*) –

24.7 pyrtma.message.set_msg_defs

set_msg_defs(*defs*)

Parameters
defs (*Dict[int, Type[MessageData]]*) –

24.8 pyrtma.message.update_msg_defs

update_msg_defs(*defs*)

Parameters
defs (*Dict[int, Type[MessageData]]*) –

Classes

Message	Message class
MessageData	MessageData base class
MessageHeader	RTMA Message Header class
RTMAJSONEncoder	JSONEncoder object used to convert MessageData to json
TypeVar	Type variable.

24.9 pyrtma.message.Message

class Message(*header*, *data*)

Bases: object

Message class

Contains message header and data

Methods

copy	Generate a copy of a message structure
from_json	Create message object from JSON string
pretty_print	Generate formatted string for pretty printing of message
to_dict	Convert message to dictionary
to_json	Convert message to JSON string

Attributes

name
type_id

Parameters

- **header** (MessageHeader) –
- **data** (MessageData) –

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*Message*) – Message structure to copy

Return type
Message

classmethod from_json(*s*)

Create message object from JSON string

Parameters
s (*str*) – JSON message string

Raises
InvalidMessageDefinition – JSON data does not match expected message defintion

Returns
Message object

Return type
Message

pretty_print(*add_tabs=0*)

Generate formatted string for pretty printing of message

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to JSON string

Parameters

minify (*bool, optional*) – Flag to minify (compact format). Defaults to False.

Returns

JSON message string

Return type

str

24.10 pyrtma.message.MessageData

class MessageData

Bases: *MessageBase*

MessageData base class

This is intended to be treated as an abstract class and and should not be directly instantiated.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_id
type_name
type_size
type_source
type_hash

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

24.11 pyrtma.message.MessageHeader

class MessageHeader

Bases: *MessageBase*

RTMA Message Header class

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	

src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

24.12 pyrtma.message.RTMAJSONEncoder

class RTMAJSONEncoder(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: `JSONEncoder`

JSONEncoder object used to convert MessageData to json

Example

data = json.dumps(msg, cls=pyrtma.encoding.RTMAJSONEncoder)

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an OverflowError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is `None` and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

Methods

default	Default method to return serializable objects
encode	Return a JSON string representation of a Python data structure.
iterencode	Encode the given object and yield each string representation as available.

Attributes

item_separator
key_separator

default(*o*)

Default method to return serializable objects

Parameters
o (*Any*) – data to serialize

Returns
Serializable data

Return type
Any

encode(*o*)

Return a JSON string representation of a Python data structure.

```
>>> from json.encoder import JSONEncoder
>>> JSONEncoder().encode({"foo": ["bar", "baz"]})
'{"foo": ["bar", "baz"]}'
```

iterencode(*o*, *_one_shot=False*)

Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

24.13 pyrtma.message.TypeVar

class TypeVar(*name*, **constraints*, *bound=None*, *covariant=False*, *contravariant=False*)

Bases: _Final, _Immutable

Type variable.

Usage:

```
T = TypeVar('T')  # Can be anything
A = TypeVar('A', str, bytes)  # Must be str or bytes
```

Type variables exist primarily for the benefit of static type checkers. They serve as the parameters for generic types as well as for generic function definitions. See class Generic for more information on generic types. Generic functions work as follows:

def repeat(x: T, n: int) -> List[T]:
'''Return a list containing n references to x.''' return [x]*n

def longest(x: A, y: A) -> A:
'''Return the longest of two strings.''' return x if len(x) >= len(y) else y

The latter example's signature is essentially the overloading of (str, str) -> str and (bytes, bytes) -> bytes. Also note that if the arguments are instances of some subclass of str, the return type is still plain str.

At runtime, isinstance(x, T) and issubclass(C, T) will raise TypeError.

Type variables defined with covariant=True or contravariant=True can be used to declare covariant or contravariant generic types. See PEP 484 for more details. By default generic types are invariant in all type variables.

Type variables can be introspected. e.g.:

T.__name__ == 'T' T.__constraints__ == () T.__covariant__ == False T.__contravariant__ = False A.__constraints__ == (str, bytes)

Note that only type variables defined in global scope can be pickled.

Methods

Exceptions

InvalidMessageDefinition	Raised when there is message definition is out of sync with sent data.
UnknownMessageType	Raised when there is no message definition.

24.14 pyrtma.message.InvalidMessageDefinition

exception InvalidMessageDefinition

Raised when there is message definition is out of sync with sent data.

24.15 pyrtma.message.UnknownMessageType

exception UnknownMessageType

Raised when there is no message definition.

CHAPTER

TWENTYFIVE

PYRTMA.MESSAGE_BASE

Functions

hexdump	Dump bytes as hex
is_dataclass	Returns True if obj is a dataclass or an instance of a dataclass.
print_ctype_array	Expand and print ctype arrays to string

25.1 pyrtma.message_base.hexdump

hexdump(*src*, *length=16*, *sep=' '*)

Dump bytes as hex

Parameters

- **src** (*bytes*) – bytes to hexdump
- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable characters. Defaults to ” “.

25.2 pyrtma.message_base.is_dataclass

is_dataclass(*obj*)

Returns True if obj is a dataclass or an instance of a dataclass.

25.3 pyrtma.message_base.print_ctype_array

print_ctype_array(*arr*)

Expand and print ctype arrays to string

Parameters

arr (*ctypes.Array*) – ctype array

Returns

string representation of ctype array

Return type

str

Classes

CStructType	alias of PyCStructType
MessageBase	MessageBase base class
MessageMeta	MessageMeta metaclass
RTMAJSONEncoder	JSONEncoder object used to convert MessageData to json
TypeVar	Type variable.

25.4 pyrtma.message_base.CStructType

CStructType

alias of PyCStructType

25.5 pyrtma.message_base.MessageBase

class MessageBase

Bases: Structure

MessageBase base class

This class should be treated as if abstract and not instantiated directly.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size

classmethod copy(*m*)

Generate a copy of a message structure

Parameters

m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

`to_dict()`
Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

`to_json(minify=False, **kwargs)`
Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

25.6 pyrtma.message_base.MessageMeta

`class MessageMeta(name, bases, namespace)`
Bases: PyCStructType

MessageMeta metaclass

Responsible for generating ctypes fields from descriptor attributes prior to class creation

Methods

from_address	access a C instance at the specified address
from_buffer	create a C instance from a writeable buffer
from_buffer_copy	create a C instance from a readable buffer
from_param	Convert a Python object into a function call parameter.
in_dll	access a C instance in a dll
mro	Return a type's method resolution order.

`__call__(*args, **kwargs)`
Call self as a function.

`__mul__(value, /)`
Return self*value.

`from_address(integer)` → C instance
access a C instance at the specified address

from_buffer(*object*, *offset=0*) → C instance

create a C instance from a writeable buffer

from_buffer_copy(*object*, *offset=0*) → C instance

create a C instance from a readable buffer

from_param()

Convert a Python object into a function call parameter.

in_dll(*dll*, *name*) → C instance

access a C instance in a dll

mro()

Return a type's method resolution order.

25.7 pyrtma.message_base.RTMAJSONEncoder

class RTMAJSONEncoder(**, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Bases: `JSONEncoder`

JSONEncoder object used to convert MessageData to json

Example

data = json.dumps(msg, cls=pyrtma.encoding.RTMAJSONEncoder)

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an OverflowError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is `None` and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

Methods

default	Default method to return serializable objects
encode	Return a JSON string representation of a Python data structure.
iterencode	Encode the given object and yield each string representation as available.

Attributes

item_separator
key_separator

default(*o*)

Default method to return serializable objects

Parameters
o (*Any*) – data to serialize

Returns
Serializable data

Return type
Any

encode(*o*)

Return a JSON string representation of a Python data structure.

```
>>> from json.encoder import JSONEncoder
>>> JSONEncoder().encode({"foo": ["bar", "baz"]})
'{"foo": ["bar", "baz"]}'
```

iterencode(*o*, *_one_shot=False*)

Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

25.8 pyrtma.message_base.TypeVar

class TypeVar(*name*, **constraints*, *bound=None*, *covariant=False*, *contravariant=False*)

Bases: _Final, _Immutable

Type variable.

Usage:

```
T = TypeVar('T')  # Can be anything
A = TypeVar('A', str, bytes)  # Must be str or bytes
```

Type variables exist primarily for the benefit of static type checkers. They serve as the parameters for generic types as well as for generic function definitions. See class Generic for more information on generic types. Generic functions work as follows:

def repeat(x: T, n: int) -> List[T]:
'''Return a list containing n references to x.''' return [x]*n

def longest(x: A, y: A) -> A:
'''Return the longest of two strings.''' return x if len(x) >= len(y) else y

The latter example's signature is essentially the overloading of (str, str) -> str and (bytes, bytes) -> bytes. Also note that if the arguments are instances of some subclass of str, the return type is still plain str.

At runtime, isinstance(x, T) and issubclass(C, T) will raise TypeError.

Type variables defined with covariant=True or contravariant=True can be used to declare covariant or contravariant generic types. See PEP 484 for more details. By default generic types are invariant in all type variables.

Type variables can be introspected. e.g.:

T.__name__ == 'T' T.__constraints__ == () T.__covariant__ == False T.__contravariant__ = False A.__constraints__ == (str, bytes)

Note that only type variables defined in global scope can be pickled.

Methods

Exceptions

JSONDecodingError	Raised when there is an error decoding a message from json.

25.9 pyrtma.message_base.JSONDecodingError

exception JSONDecodingError

Raised when there is an error decoding a message from json.

CHAPTER

TWENTYSIX

PYRTMA.MESSAGE_DATA

Classes

MessageBase	MessageBase base class
MessageData	MessageData base class
MessageMeta	MessageMeta metaclass

26.1 pyrtma.message_data.MessageBase

class MessageBase

Bases: Structure

MessageBase base class

This class should be treated as if abstract and not instantiated directly.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

26.2 pyrtma.message_data.MessageData

class MessageData

Bases: *MessageBase*

MessageData base class

This is intended to be treated as an abstract class and and should not be directly instantiated.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size
type_def
type_id
type_name
type_size
type_source
type_hash

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.

- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

26.3 pyrtma.message_data.MessageMeta

class MessageMeta(*name*, *bases*, *namespace*)

Bases: PyCStructType

MessageMeta metaclass

Responsible for generating ctypes fields from descriptor attributes prior to class creation

Methods

from_address	access a C instance at the specified address
from_buffer	create a C instance from a writeable buffer
from_buffer_copy	create a C instance from a readable buffer
from_param	Convert a Python object into a function call parameter.
in_dll	access a C instance in a dll
mro	Return a type's method resolution order.

__call__(**args*, ***kwargs*)
Call self as a function.

__mul__(*value*, */*)
Return self*value.

from_address(*integer*) → C instance
access a C instance at the specified address

from_buffer(*object*, *offset=0*) → C instance
create a C instance from a writeable buffer

from_buffer_copy(*object*, *offset=0*) → C instance
create a C instance from a readable buffer

from_param()
Convert a Python object into a function call parameter.

in_dll(*dll*, *name*) → C instance
access a C instance in a dll

mro()
Return a type's method resolution order.

CHAPTER

TWENTYSEVEN

PYRTMA.PARSER

Message definition YAML parser

Functions

asdict	Return the fields of a dataclass instance as a new dictionary mapping field names to field values.
copy	Shallow copy operation on arbitrary Python objects.
dataclass	Returns the same class as was passed in, with dunder methods added based on the fields defined in the class.
field	Return an object to identify dataclass fields.
is_dataclass	Returns True if obj is a dataclass or an instance of a dataclass.
sha256	Returns a sha256 hash object; optionally initialized with a string

27.1 pyrtma.parser.asdict

asdict(*obj*, *, *dict_factory*=<*class 'dict'*>)

Return the fields of a dataclass instance as a new dictionary mapping field names to field values.

Example usage:

@dataclass class C:

x: int y: int

c = C(1, 2) assert asdict(c) == {'x': 1, 'y': 2}

If given, 'dict_factory' will be used instead of built-in dict. The function applies recursively to field values that are dataclass instances. This will also look into built-in containers: tuples, lists, and dicts.

27.2 pyrtma.parser.copy

copy(*x*)

Shallow copy operation on arbitrary Python objects.

See the module's __doc__ string for more info.

27.3 pyrtma.parser.dataclass

dataclass(*cls=None*, */*, ***, *init=True*, *repr=True*, *eq=True*, *order=False*, *unsafe_hash=False*, *frozen=False*)

Returns the same class as was passed in, with dunder methods added based on the fields defined in the class.

Examines PEP 526 __annotations__ to determine fields.

If init is true, an __init__() method is added to the class. If repr is true, a __repr__() method is added. If order is true, rich comparison dunder methods are added. If unsafe_hash is true, a __hash__() method function is added. If frozen is true, fields may not be assigned to after instance creation.

27.4 pyrtma.parser.field

field(***, *default=<dataclasses._MISSING_TYPE object>*, *default_factory=<dataclasses._MISSING_TYPE object>*, *init=True*, *repr=True*, *hash=None*, *compare=True*, *metadata=None*)

Return an object to identify dataclass fields.

default is the default value of the field. default_factory is a 0-argument function called to initialize a field's value. If init is True, the field will be a parameter to the class's __init__() function. If repr is True, the field will be included in the object's repr(). If hash is True, the field will be included in the object's hash(). If compare is True, the field will be used in comparison functions. metadata, if specified, must be a mapping which is stored but not otherwise examined by dataclass.

It is an error to specify both default and default_factory.

27.5 pyrtma.parser.is_dataclass

is_dataclass(*obj*)

Returns True if obj is a dataclass or an instance of a dataclass.

27.6 pyrtma.parser.sha256

sha256(*string=b''*)

Returns a sha256 hash object; optionally initialized with a string

Classes

CompilerOptions	
ConstantExpr	
ConstantString	
CustomEncoder	Constructor for JSONEncoder, with sensible defaults.
Field	
HID	
Import	
MDF	
MID	
MT	
Metadata	
NativeType	
Parser	Parser class
SDF	
TypeAlias	
YAML	typ: 'rt'/None -> RoundTripLoader/RoundTripDumper, (default)

27.7 pyrtma.parser.CompilerOptions

class CompilerOptions(*name*, *value*, *src*)

Bases: object

Methods

Attributes

name
value
src

Parameters

- **name** (*str*) –
- **value** (*Union[int, float, bool, str]*) –
- **src** (*Path*) –

27.8 pyrtma.parser.ConstantExpr

class ConstantExpr(*name*, *expression*, *expanded*, *value*, *src*)

Bases: object

Methods

Attributes

name
expression
expanded
value
src

Parameters

- **name** (*str*) –
- **expression** (*str*) –
- **expanded** (*Optional[str]*) –

- **value** (*Union[str, int, float]*) –
- **src** (*Path*) –

27.9 pyrtma.parser.ConstantString

class ConstantString(*name*, *value*, *src*)

Bases: `object`

Methods

Attributes

name
value
src

Parameters

- **name** (*str*) –
- **value** (*str*) –
- **src** (*Path*) –

27.10 pyrtma.parser.CustomEncoder

class CustomEncoder(**, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Bases: `JSONEncoder`

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an OverflowError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is `None` and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

Methods

default	Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).
encode	Return a JSON string representation of a Python data structure.
iterencode	Encode the given object and yield each string representation as available.

Attributes

item_separator
key_separator

default(*o*)

Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

Return type
Any

Parameters
o (*Any*) –

encode(*o*)

Return a JSON string representation of a Python data structure.

```
>>> from json.encoder import JSONEncoder
>>> JSONEncoder().encode({"foo": ["bar", "baz"]})
'{"foo": ["bar", "baz"]}'
```

iterencode(*o*, *_one_shot=False*)

Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

27.11 pyrtma.parser.Field

class Field(*name*, *type_name*, *type_obj*, *length_expression=None*, *length_expanded=None*, *length=None*, *offset=-1*)

Bases: `object`

Methods

Attributes

alignment
base_size
format
length
length_expanded
length_expression
offset
size
name
type_name
type_obj

Parameters

- **name** (*str*) –
- **type_name** (*str*) –
- **type_obj** (*Union[NativeType, TypeAlias, MDF, SDF]*) –
- **length_expression** (*Optional[str]*) –
- **length_expanded** (*Optional[str]*) –
- **length** (*Optional[int]*) –
- **offset** (*int*) –

27.12 pyrtma.parser.HID

class HID(*name*, *value*, *src*)

Bases: object

Methods

Attributes

name
value
src

Parameters

- **name** (*str*) –
- **value** (*int*) –
- **src** (*Path*) –

27.13 pyrtma.parser.Import

class Import(*file*, *src*)

Bases: object

Methods

Attributes

file
src

Parameters

- **file** (*Path*) –
- **src** (*Path*) –

27.14 pyrtma.parser.MDF

class MDF(*raw*, *hash*, *name*, *type_id*, *src*, *fields=<factory>*, *alignment=8*)

Bases: object

Methods

Attributes

alignment
format
signal
size
raw
hash
name
type_id
src
fields

Parameters

- **raw** (*str*) –
- **hash** (*str*) –
- **name** (*str*) –
- **type_id** (*int*) –
- **src** (*Path*) –
- **fields** (*List[Field]*) –
- **alignment** (*int*) –

27.15 pyrtma.parser.MID

class MID(*name*, *value*, *src*)

Bases: object

Methods

Attributes

name
value
src

Parameters

- **name** (*str*) –
- **value** (*int*) –
- **src** (*Path*) –

27.16 pyrtma.parser.MT

class MT(*name*, *value*, *src*)

Bases: object

Methods

Attributes

name
value
src

Parameters

- **name** (*str*) –
- **value** (*int*) –
- **src** (*Path*) –

27.17 pyrtma.parser.Metadata

class Metadata(*name*, *value*, *src*)

Bases: object

Methods

Attributes

name
value
src

Parameters

- **name** (*str*) –
- **value** (*Union[int, float, bool, str]*) –
- **src** (*Path*) –

27.18 pyrtma.parser.NativeType

class NativeType(*name*, *size*, *format*)

Bases: object

Methods

Attributes

name
size
format

Parameters

- **name** (*str*) –
- **size** (*int*) –
- **format** (*str*) –

27.19 pyrtma.parser.Parser

class Parser(*debug=False*, *validate_alignment=True*, *auto_pad=True*, *import_coredefs=True*)

Bases: object

Parser class

Parser class

Parameters

- **debug** (*bool, optional*) – Flag for debug mode. Defaults to False.
- **validate_alignment** (*bool*) –
- **auto_pad** (*bool*) –
- **import_coredefs** (*bool*) –

Methods

add_fields	
check_alignment	Confirm 64 bit alignment of structures
check_duplicate_name	Check namespaces for conflicting names.
check_key_value_separation	
check_name	Check that names start with a letter.
clear	
expand_expression	**rtype** Tuple[str, int]
get_ctype_cls	**rtype** Type[Structure]
get_ctype_size	**rtype** int
handle_alias	Find the base type ultimately represented by the typedef alias
handle_compiler_options	
handle_expression	
handle_host_id	
handle_import	
handle_message_def	
handle_metadata	
handle_module_id	
handle_reserve	
handle_signal	
handle_string	
handle_struct	
parse	

continues on next page

Table 1 – continued from previous page

parse_compiler_options	**rtype** Dict[str, *CompilerOptions*]
parse_file	
parse_options	
parse_options_text	
parse_text	
to_json	
trim_root	**rtype** Path
validate_msg_def	
validate_msg_id	
warning	

check_alignment(*s*)

Confirm 64 bit alignment of structures

Parameters
s (*Union[SDF, MDF]*) –

check_duplicate_name(*section*, *name*, *namespaces*)

Check namespaces for conflicting names.

Parameters

- **section** (*str*) –
- **name** (*str*) –
- **namespaces** (*Tuple[str, ...]*) –

check_name(*name*)

Check that names start with a letter.

Parameters
name (*str*) –

handle_alias(*alias*, *ftype*)

Find the base type ultimately represented by the typedef alias

Parameters

- **alias** (*str*) –
- **ftype** (*str*) –

27.20 pyrtma.parser.SDF

class SDF(*raw*, *hash*, *name*, *src*, *fields=<factory>*, *alignment=8*)

Bases: object

Methods

Attributes

alignment
format
size
raw
hash
name
src
fields

Parameters

- **raw** (*str*) –
- **hash** (*str*) –
- **name** (*str*) –
- **src** (*Path*) –
- **fields** (*List[Field]*) –
- **alignment** (*int*) –

27.21 pyrtma.parser.TypeAlias

class TypeAlias(*name*, *type_name*, *type_obj*, *src*)

Bases: object

Methods

Attributes

alignment
format
size
name
type_name
type_obj
src

Parameters

- **name** (*str*) –
- **type_name** (*str*) –
- **type_obj** (*Union[NativeType, SDF]*) –
- **src** (*Path*) –

27.22 pyrtma.parser.YAML

class YAML(*, *typ=None*, *pure=False*, *output=None*, *plug_ins=None*)

Bases: object

typ: 'rt'/None -> RoundTripLoader/RoundTripDumper, (default)
'safe' -> SafeLoader/SafeDumper, 'unsafe' -> normal/unsafe Loader/Dumper (pending deprecation) 'full' -> full Dumper only, including python built-ins that are

potentially unsafe to load

'base' -> baseloader

pure: if True only use Python modules input/output: needed to work as context manager plug_ins: a list of plug-in files

Methods

Xdump_all	Serialize a sequence of Python objects into a YAML stream.
compact **rtype** None	
compose	Parse the first YAML document in a stream and produce the corresponding representation tree.
compose_all	Parse all YAML documents in a stream and produce corresponding representation trees.
dump **rtype** Any	
dump_all **rtype** Any	
emit	Emit YAML parsing events into a stream.
get_constructor_parser	the old cyaml needs special setup, and therefore the stream
get_serializer_representer_emitter **rtype** Any	
load	at this point you either have the non-pure Parser (which has its own reader and scanner) or you have the pure Parser.
load_all **rtype** Any	
map **rtype** Any	
official_plug_ins	search for list of subdirs that are plug-ins, if __file__ is not available, e.g.
parse	Parse a YAML stream and produce parsing events.
register_class	register a class for dumping/loading :rtype: Any
scan	Scan a YAML stream and produce scanning tokens.
seq **rtype** Any	
serialize	Serialize a representation tree into a YAML stream.
serialize_all	Serialize a sequence of representation trees into a YAML stream.

Attributes

block_seq_indent
composer
constructor
emitter
indent
parser
reader
representer
resolver
scanner
serializer
tags
version

Parameters

- **typ** (*Optional[Union[List[Text], Text]]*) –
- **pure** (*Any*) –
- **output** (*Any*) –
- **plug_ins** (*Any*) –

Xdump_all(*documents*, *stream*, *, *transform=None*)

Serialize a sequence of Python objects into a YAML stream.

Return type
Any

Parameters

- **documents** (*Any*) –
- **stream** (*Any*) –
- **transform** (*Any*) –

compose(*stream*)

Parse the first YAML document in a stream and produce the corresponding representation tree.

Return type
Any

Parameters
stream (*Union[Path, StreamTextType]*) –

compose_all(*stream*)

Parse all YAML documents in a stream and produce corresponding representation trees.

Return type
Any

Parameters
stream (*Union[Path, StreamTextType]*) –

emit(*events*, *stream*)

Emit YAML parsing events into a stream. If stream is None, return the produced string instead.

Return type
None

Parameters

- **events** (*Any*) –
- **stream** (*Any*) –

get_constructor_parser(*stream*)

the old cyaml needs special setup, and therefore the stream

Return type
Any

Parameters
stream (*StreamTextType*) –

load(*stream*)

at this point you either have the non-pure Parser (which has its own reader and scanner) or you have the pure Parser. :rtype: Any

If the pure Parser is set, then set the Reader and Scanner, if not already set. If either the Scanner or Reader are set, you cannot use the non-pure Parser,

so reset it to the pure parser and set the Reader resp. Scanner if necessary

Parameters
stream (*Union[Path, StreamTextType]*) –

Return type
Any

official_plug_ins()

search for list of subdirs that are plug-ins, if __file__ is not available, e.g. single file installers that are not properly emulating a file-system (issue 324) :rtype: Any

no plug-ins will be found. If any are packaged, you know which file that are and you can explicitly provide it during instantiation:

yaml = ruamel.yaml.YAML(plug_ins=[‘ruamel/yaml/jinja2/__plug_in__’])

Return type
Any

parse(*stream*)

Parse a YAML stream and produce parsing events.

Return type
Any

Parameters
stream (*StreamTextType*) –

register_class(*cls*)

register a class for dumping/loading :rtype: Any

- if it has attribute yaml_tag use that to register, else use class name
- if it has methods to_yaml/from_yaml use those to dump/load else dump attributes as mapping

Parameters
cls (*Any*) –

Return type
Any

scan(*stream*)

Scan a YAML stream and produce scanning tokens.

Return type
Any

Parameters
stream (*StreamTextType*) –

serialize(*node*, *stream*)

Serialize a representation tree into a YAML stream. If stream is None, return the produced string instead.

Return type
Any

Parameters

- **node** (*Any*) –
- **stream** (*Optional[StreamType]*) –

serialize_all(*nodes*, *stream*)

Serialize a sequence of representation trees into a YAML stream. If stream is None, return the produced string instead.

Return type
Any

Parameters

- **nodes** (*Any*) –
- **stream** (*Optional[StreamType]*) –

Exceptions

AlignmentError	Raised when a struct is not 64-bit aligned
CircularRefError	Raised when an expression contains a circular reference
DuplicateNameError	Raised when the a name is already in use
ExpressionExpansionError	Raised when an expression can not be expanded.
FileFormatError	Raised when the wrong file extension is referenced.
HostIDError	Raised when the a host id is already in use
InvalidMessageSize	Raised when a message is too large
InvalidTypeError	Raised when a field contains the wrong type of data
MessageIDError	Raised when the a message id is already in use
ModuleIDError	Raised when the a module id is already in use
ParserError	Base class for all parser exceptions
RTMASyntaxError	Raised when the parser encounters invalid RTMA syntax
RecurisionError	Raised when recursion limit is exceeded evaluating aliases and expressions.
YAMLSyntaxError	Raised when the parser encounters invalid YAML

27.23 pyrtma.parser.AlignmentError

exception AlignmentError

Raised when a struct is not 64-bit aligned

27.24 pyrtma.parser.CircularRefError

exception CircularRefError

Raised when an expression contains a circular reference

27.25 pyrtma.parser.DuplicateNameError

exception DuplicateNameError

Raised when the a name is already in use

27.26 pyrtma.parser.ExpressionExpansionError

exception ExpressionExpansionError

Raised when an expression can not be expanded.

27.27 pyrtma.parser.FileFormatError

exception FileFormatError

Raised when the wrong file extension is referenced.

27.28 pyrtma.parser.HostIDError

exception HostIDError

Raised when the a host id is already in use

27.29 pyrtma.parser.InvalidMessageSize

exception InvalidMessageSize

Raised when a message is too large

27.30 pyrtma.parser.InvalidTypeError

exception InvalidTypeError

Raised when a field contains the wrong type of data

27.31 pyrtma.parser.MessageIDError

exception MessageIDError

Raised when the a message id is already in use

27.32 pyrtma.parser.ModuleIDError

exception ModuleIDError

Raised when the a module id is already in use

27.33 pyrtma.parser.ParserError

exception ParserError

Base class for all parser exceptions

27.34 pyrtma.parser.RTMASyntaxError

exception RTMASyntaxError

Raised when the parser encounters invalid RTMA syntax

27.35 pyrtma.parser.RecurisionError

exception RecurisionError

Raised when recursion limit is exceeded evaluating aliases and expressions.

27.36 pyrtma.parser.YAMLSyntaxError

exception YAMLSyntaxError

Raised when the parser encounters invalid YAML

CHAPTER

TWENTYEIGHT

PYRTMA.VALIDATORS

Functions

abstractmethod	A decorator indicating abstract methods.
contextmanager	@contextmanager decorator.
disable_message_validation	Context manager function to temporarily disable message field validation Use with *with* keyword: *with disable_message_validation():*
overload	Decorator for overloaded functions/methods.

28.1 pyrtma.validators.abstractmethod

abstractmethod(*funcobj*)

A decorator indicating abstract methods.

Requires that the metaclass is ABCMeta or derived from it. A class that has a metaclass derived from ABCMeta cannot be instantiated unless all of its abstract methods are overridden. The abstract methods can be called using any of the normal 'super' call mechanisms. abstractmethod() may be used to declare abstract methods for properties and descriptors.

Usage:

class C(metaclass=ABCMeta):
@abstractmethod def my_abstract_method(self, ...):

...

28.2 pyrtma.validators.contextmanager

contextmanager(*func*)

@contextmanager decorator.

Typical usage:

@contextmanager def some_generator(<arguments>):

<setup> try:

yield <value>

finally:
<cleanup>

This makes this:

with some_generator(<arguments>) as <variable>:
<body>

equivalent to this:

<setup> try:

<variable> = <value> <body>

finally:
<cleanup>

28.3 pyrtma.validators.disable_message_validation

disable_message_validation(*ignore=False*)

Context manager function to temporarily disable message field validation Use with *with* keyword: *with disable_message_validation():*

Optionally pass in ignore=True to do nothing, e.g. for debugging:

``` DEBUG = True with disable_message_validation(ignore=DEBUG):

... # disable validation unless DEBUG is True

```

28.4 pyrtma.validators.overload

overload(*func*)

Decorator for overloaded functions/methods.

In a stub file, place two or more stub definitions for the same function in a row, each decorated with @overload. For example:

@overload def utf8(value: None) -> None: ... @overload def utf8(value: bytes) -> bytes: ... @overload def utf8(value: str) -> bytes: ...

In a non-stub file (i.e. a regular .py file), do the same but follow it with an implementation. The implementation should *not* be decorated with @overload. For example:

@overload def utf8(value: None) -> None: ... @overload def utf8(value: bytes) -> bytes: ... @overload def utf8(value: str) -> bytes: ... def utf8(value):

implementation goes here

Classes

ABCMeta	Metaclass for defining Abstract Base Classes (ABCs).
ArrayField	Array field validator base class
Byte	Validator for single byte values
ByteArray	Validator class for Bytes arrays
Char	Validator for scalar char values
ContextVar	
Double	Double (64-bit float) validator class
FieldValidator	Abstract base class for all message field validator descriptors
Float	32-bit Float validator class
FloatArray	Validator class for float arrays
FloatValidatorBase	Abstract base class for float type validators
Generic	Abstract base class for generic types.
Int16	Validator for 16-bit integers
Int32	Validator for 32-bit integers
Int64	Validator for 64-bit integers
Int8	Validator for 8-bit integers
IntArray	IntArray validator class
IntValidatorBase	Abstract base class for integer type validators
MessageBase	MessageBase base class
String	Validator for strings (char arrays)
Struct	Validator class for Structures
StructArray	Validator for structure arrays
TypeVar	Type variable.
Uint16	Validator for unsigned 16-bit integers
Uint32	Validator for unsigned 32-bit integers
Uint64	Validator for unsigned 64-bit integers
Uint8	Validator for unsigned 8-bit integers

28.5 pyrtma.validators.ABCMeta

class ABCMeta(*name*, *bases*, *namespace*, ***kwargs*)

Bases: type

Metaclass for defining Abstract Base Classes (ABCs).

Use this metaclass to create an ABC. An ABC can be subclassed directly, and then acts as a mix-in class. You can also register unrelated concrete classes (even built-in classes) and unrelated ABCs as 'virtual subclasses' – these and their descendants will be considered subclasses of the registering ABC by the built-in issubclass() function, but the registering ABC won't show up in their MRO (Method Resolution Order) nor will method implementations defined by the registering ABC be callable (not even via super()).

Methods

mro	Return a type's method resolution order.
register	Register a virtual subclass of an ABC.

__call__(**args*, ***kwargs*)

Call self as a function.

mro()

Return a type's method resolution order.

register(*subclass*)

Register a virtual subclass of an ABC.

Returns the subclass, to allow usage as a class decorator.

28.6 pyrtma.validators.ArrayField

class ArrayField(*validator*, *len*)

Bases: *FieldValidator*, Sequence, *Generic*[_FV]

Array field validator base class

Array field validator base class

Parameters

- **validator** (*Type[_FV]*) – Field validator class for datatype
- **len** (*int*) – Field length

Methods

count	

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters

value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

28.7 pyrtma.validators.Byte

class Byte(**args*)

Bases: *FieldValidator*[_P, int], *Generic*[_P]

Validator for single byte values

Methods

validate_many	Validate multiple byte values
validate_one	validate a single byte value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple byte values

Parameters
value (*Union[Iterable[int], bytes, bytearray]*) – Byte values to validate

Raises

- TypeError – Wrong type
- ValueError – Value out of range

validate_one(*value*)

validate a single byte value

Parameters
value (*Union[int, bytes, bytearray]*) – Byte value to validate

Raises

- **ValueError** – Value out of range
- **TypeError** – Wrong type

28.8 pyrtma.validators.ByteArray

class ByteArray(*len*)

Bases: *ArrayField*[*Byte*]

Validator class for Bytes arrays

Validator class for Bytes arrays

Parameters
len (*int*) – Byte array length

Methods

count	

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters
value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

28.9 pyrtma.validators.Char

class Char

Bases: *String*

Validator for scalar char values

Methods

validate_many	Validate multiple strings
validate_one	Validate a string value

validate_many(*value*)

Validate multiple strings

Not implemented

Raises
NotImplementedError –

validate_one(*value*)

Validate a string value

Parameters
value (*str*) – String value

Raises

- **TypeError** – Wrong type
- **ValueError** – String exceeds max length

28.10 pyrtma.validators.ContextVar

class ContextVar

Bases: object

Methods

get	Return a value for the context variable for the current context.
reset	Reset the context variable.
set	Call to set a new value for the context variable in the current context.

Attributes

name

get()

Return a value for the context variable for the current context.

If there is no value for the variable in the current context, the method will:

- return the value of the default argument of the method, if provided; or
- return the default value for the context variable, if it was created with one; or
- raise a LookupError.

reset(*token*, /)

Reset the context variable.

The variable is reset to the value it had before the *ContextVar.set()* that created the token was used.

set(*value*, /)

Call to set a new value for the context variable in the current context.

The required value argument is the new value for the context variable.

Returns a Token object that can be used to restore the variable to its previous value via the *ContextVar.reset()* method.

28.11 pyrtma.validators.Double

class Double(**args*)

Bases: *FloatValidatorBase*

Double (64-bit float) validator class

Methods

validate_many	Validate multiple float values
validate_one	Validate a float value

validate_many(*value*)

Validate multiple float values

Parameters
value (*Iterable[float]*) – Iterable of floats to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

validate_one(*value*)

Validate a float value

Parameters
value (*float*) – Float value

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

28.12 pyrtma.validators.FieldValidator

class FieldValidator

Bases: *Generic*[_P, _V]

Abstract base class for all message field validator descriptors

Methods

validate_many
validate_one

28.13 pyrtma.validators.Float

class Float(**args*)

Bases: *FloatValidatorBase*

32-bit Float validator class

Methods

validate_many	Validate multiple float values
validate_one	Validate a float value

validate_many(*value*)

Validate multiple float values

Parameters
value (*Iterable[float]*) – Iterable of floats to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

validate_one(*value*)

Validate a float value

Parameters
value (*float*) – Float value

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

28.14 pyrtma.validators.FloatArray

class FloatArray(*validator*, *len*)

Bases: *ArrayField*[_FPV], *Generic*[_FPV]

Validator class for float arrays

Validator class for float arrays

Parameters

- **validator** (*Type[_FPV]*) – Float type validator
- **len** (*int*) – Array length

Methods

count	

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters
value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

28.15 pyrtma.validators.FloatValidatorBase

class FloatValidatorBase(**args*)

Bases: *FieldValidator*[_P, float], *Generic*[_P]

Abstract base class for float type validators

Methods

validate_many	Validate multiple float values
validate_one	Validate a float value

validate_many(*value*)

Validate multiple float values

Parameters
value (*Iterable[float]*) – Iterable of floats to validate

Raises

- TypeError – Wrong type
- ValueError – Value cannot be precisely represented with this datatype

validate_one(*value*)

Validate a float value

Parameters
value (*float*) – Float value

Raises

- **TypeError** – Wrong type
- **ValueError** – Value cannot be precisely represented with this datatype

28.16 pyrtma.validators.Generic

class Generic(**args*, ***kwds*)

Bases: object

Abstract base class for generic types.

A generic type is typically declared by inheriting from this class parameterized with one or more type variables. For example, a generic mapping type might be defined as:

```
class Mapping(Generic[KT, VT]):
    def __getitem__(self, key: KT) -> VT:
        ...
    # Etc.
```

This class can then be used as follows:

```
def lookup_name(mapping: Mapping[KT, VT], key: KT, default: VT) -> VT:
    try:
        return mapping[key]
    except KeyError:
        return default
```

Methods

28.17 pyrtma.validators.Int16

class Int16(**args*)

Bases: *IntValidatorBase*

Validator for 16-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.18 pyrtma.validators.Int32

class Int32(**args*)

Bases: *IntValidatorBase*

Validator for 32-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.19 pyrtma.validators.Int64

class Int64(**args*)

Bases: *IntValidatorBase*

Validator for 64-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.20 pyrtma.validators.Int8

class Int8(**args*)

Bases: *IntValidatorBase*

Validator for 8-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.21 pyrtma.validators.IntArray

class IntArray(*validator*, *len*)

Bases: *ArrayField*[_IV], *Generic*[_IV]

IntArray validator class

IntArray validator class

Parameters

- **validator** (*Type[IV]*) – Field validator class for Int type
- **len** (*int*) – Field length

Methods

count	

index	Raises ValueError if the value is not present.
validate_array	Validate array
validate_many	Validate multiple values
validate_one	Validate one value

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

validate_array(*value*)

Validate array

Parameters
value (ArrayField) – Array value to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple values

Parameters
value – Values to validate

validate_one(*value*)

Validate one value

Parameters
value – Value to validate

28.22 pyrtma.validators.IntValidatorBase

class IntValidatorBase(**args*)

Bases: *FieldValidator*[_P, int], *Generic*[_P]

Abstract base class for integer type validators

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.23 pyrtma.validators.MessageBase

class MessageBase

Bases: Structure

MessageBase base class

This class should be treated as if abstract and not instantiated directly.

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

size

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

28.24 pyrtma.validators.String

class String(*len*)

Bases: *FieldValidator*[_P, str], *Generic*[_P]

Validator for strings (char arrays)

Methods

validate_many	Validate multiple strings
validate_one	Validate a string value

validate_many(*value*)

Validate multiple strings

Not implemented

Raises
NotImplementedError –

validate_one(*value*)

Validate a string value

Parameters
value (*str*) – String value

Raises

- TypeError – Wrong type
- ValueError – String exceeds max length

28.25 pyrtma.validators.Struct

class Struct(*_ctype*)

Bases: *FieldValidator*, *Generic*[_S]

Validator class for Structures

Methods

validate_many	Validate multiple structures
validate_one	Validate a structure

validate_many(*value*)

Validate multiple structures

Parameters
value (*Iterable[_S]*) – Iterable of structures to validate

Raises
TypeError – Wrong type

validate_one(*value*)

Validate a structure

Parameters
value (*_S*) – Structure value to validate

Raises
TypeError – Wrong type

28.26 pyrtma.validators.StructArray

class StructArray(*msg_struct*, *len*)

Bases: *FieldValidator*, Sequence, *Generic*[_S]

Validator for structure arrays

Validator for structure arrays

Parameters

- **msg_struct** (*Type[_S]*) – Structure class
- **len** (*int*) – Array length

Methods

count	

index	Raises ValueError if the value is not present.
pretty_print	Generate formatted string for structure array
validate_array	Validate structure array
validate_many	Validate multiple structures
validate_one	Validate a structure

count(*value*) → integer -- return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

pretty_print(*add_tabs=0*)

Generate formatted string for structure array

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

validate_array(*value*)

Validate structure array

Parameters
value (*StructArray[_S]*) – StructArray to validate

Raises
TypeError – Wrong type

validate_many(*value*)

Validate multiple structures

Parameters
value (*Iterable[_S]*) – Structure values to validate

validate_one(*value*)

Validate a structure

Parameters
value (*_S*) – Structure value to validate

28.27 pyrtma.validators.TypeVar

class TypeVar(*name*, **constraints*, *bound=None*, *covariant=False*, *contravariant=False*)

Bases: _Final, _Immutable

Type variable.

Usage:

```
T = TypeVar('T')  # Can be anything
A = TypeVar('A', str, bytes)  # Must be str or bytes
```

Type variables exist primarily for the benefit of static type checkers. They serve as the parameters for generic types as well as for generic function definitions. See class Generic for more information on generic types. Generic functions work as follows:

def repeat(x: T, n: int) -> List[T]:
'''Return a list containing n references to x.''' return [x]*n

def longest(x: A, y: A) -> A:
'''Return the longest of two strings.''' return x if len(x) >= len(y) else y

The latter example's signature is essentially the overloading of (str, str) -> str and (bytes, bytes) -> bytes. Also note that if the arguments are instances of some subclass of str, the return type is still plain str.

At runtime, isinstance(x, T) and issubclass(C, T) will raise TypeError.

Type variables defined with covariant=True or contravariant=True can be used to declare covariant or contravariant generic types. See PEP 484 for more details. By default generic types are invariant in all type variables.

Type variables can be introspected. e.g.:

T.__name__ == 'T' T.__constraints__ == () T.__covariant__ == False T.__contravariant__ = False A.__constraints__ == (str, bytes)

Note that only type variables defined in global scope can be pickled.

Methods

28.28 pyrtma.validators.Uint16

class Uint16(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 16-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.29 pyrtma.validators.Uint32

class Uint32(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 32-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.30 pyrtma.validators.Uint64

class Uint64(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 64-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

28.31 pyrtma.validators.Uint8

class Uint8(**args*)

Bases: *IntValidatorBase*

Validator for unsigned 8-bit integers

Methods

validate_many	Validate multiple integer values
validate_one	Validate an integer value

Attributes

max
min
size
unsigned

validate_many(*value*)

Validate multiple integer values

Parameters
value (*Iterable[int]*) – Iterable of integers to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

validate_one(*value*)

Validate an integer value

Parameters
value (*int*) – Integer to validate

Raises

- **TypeError** – Wrong type
- **ValueError** – Integer out of range for this datatype

CHAPTER

TWENTYNINE

PYRTMA.WEB_MANAGER

Functions

cast	Cast a value to a type.
get_msg_cls	get msg class for a given message type ID
main	Main function for starting web_manager
ws_client_connect	Websocket client connect
ws_client_disconnect	Websocket client disconnect

29.1 pyrtma.web_manager.cast

cast(*typ*, *val*)

Cast a value to a type.

This returns the value unchanged. To the type checker this signals that the return value has the designated type, but at runtime we intentionally don't check anything (we want this to be as fast as possible).

29.2 pyrtma.web_manager.get_msg_cls

get_msg_cls(*id*)

get msg class for a given message type ID

Parameters
id (*int*) – Message Type ID

Raises
UnknownMessageType – Message type is undefined

Returns
Message class

Return type
Type[*MessageData*]

29.3 pyrtma.web_manager.main

main()

Main function for starting web_manager

29.4 pyrtma.web_manager.ws_client_connect

ws_client_connect(*client*, *server*)

Websocket client connect

Called for every client connecting (after handshake)

Parameters

- **client** (*Dict[str, Any]*) – Client dictionary
- **server** (*WebMessageManager*) – WebMessageManager Server object

29.5 pyrtma.web_manager.ws_client_disconnect

ws_client_disconnect(*client*, *server*)

Websocket client disconnect

Called for every client disconnecting

Parameters

- **client** (*Dict[str, Any]*) – Client dictionary
- **server** (*WebMessageManager*) – WebMessageManager Server object

Classes

Client	RTMA Client interface
Message	Message class
MessageHeader	RTMA Message Header class
RTMAWebSocketHandler	RTMA Web Socket Handler class
RichHandler	A logging handler that renders output with Rich.
TCPServer	Base class for various socket-based server classes.
WebMessageManager	WebMessageManager class
WebSocketHandler	
WebsocketServer	A websocket server waiting for clients to connect.

29.6 pyrtma.web_manager.Client

class Client(*module_id=0*, *host_id=0*, *timecode=False*)

Bases: object

RTMA Client interface

Parameters

- **module_id** (*optional*) – Static module ID, which must be unique. Defaults to 0, which generates a dynamic module ID.
- **host_id** (*optional*) – Host ID. Defaults to 0.
- **timecode** (*optional*) – Add additional timecode fields to message header, used by some projects at RNEL. Defaults to False.

Methods

connect	Connect to message manager server
discard_messages	Read and discard messages in socket buffer up to timeout
disconnect	Disconnect from message manager server
forward_message	Forward a message
pause_all_subscriptions	Pause all subscribed types
pause_subscription	Pause subscription to message types
paused_subscription_context	Context manager to pause subscriptions to a list of message types
read_message	Read a message
resume_all_subscriptions	Resume all paused subscriptions
resume_subscription	Resume subscription to message types
send_message	Send a message
send_module_ready	Send a signal to message manager that client is ready
send_signal	Send a signal
subscribe	Subscribe to message types
subscription_context	Context manager to subscribe to a list of message types
unsubscribe	Unsubscribe from message types
unsubscribe_from_all	Unsubscribe from all subscribed types

Attributes

connected	Status of connection to message manager server
header_cls	Class defining the RTMA message header
ip_addr	Message manager IP address string
module_id	Numeric module ID of client
msg_count	Count of messages that have been sent
paused_subscribed_types	Subscriptions on pause
port	Message manager port number
server	Message manager server address as a (IP_addr, port_num) tuple
sock	Underlying socket connection with MessageManager
subscribed_types	List of subscribed message types

connect(*server_name='localhost:7111'*, *logger_status=False*, *daemon_status=False*)

Connect to message manager server

Parameters

- **server_name** (*optional*) – IP_addr:port_num string associated with message manager. Defaults to "localhost:7111".
- **logger_status** (*optional*) – Flag to declare client as a logger module. Logger modules are automatically subscribed to all message types. Defaults to False.
- **daemon_status** (*optional*) – Flag to declare client as a daemon. Defaults to False.

Raises

MessageManagerNotFound – Unable to connect to message manager

property connected: bool

Status of connection to message manager server

discard_messages(*timeout=1*)

Read and discard messages in socket buffer up to timeout

Parameters

timeout (*optional*) – Maximum time in seconds to loop through message buffer. Defaults to 1.

Return type

bool

Returns

True if all messages have been read, False if messages remain in buffer

disconnect()

Disconnect from message manager server

forward_message(*msg_hdr*, *msg_data=None*, *timeout=-1*)

Forward a message

A message is a packet that contains a defined data payload. To send a message without associated data, see *send_signal()*.

Parameters

- **msg_hdr** (*MessageHeader*) – Object containing RTMA header to send

- **msg_data** (Optional[*MessageData*]) – Object containing the message to send
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

property header_cls: Type[*MessageHeader*]

Class defining the RTMA message header

property ip_addr: str

Message manager IP address string

property module_id: int

Numeric module ID of client

property msg_count: int

Count of messages that have been sent

pause_all_subscriptions()

Pause all subscribed types

pause_subscription(*msg_list*)

Pause subscription to message types

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to temporarily unsubscribe to

property paused_subscribed_types: Set[int]

Subscriptions on pause

paused_subscription_context(*msg_list*)

Context manager to pause subscriptions to a list of message types

Message types will automatically resume subscriptions after exiting context.

Parameters

msg_list (*Iterable[int]*) – A list of numeric message IDs to temporarily unsubscribe to

property port: int

Message manager port number

read_message(*timeout=-1*, *ack=False*, *sync_check=False*)

Read a message

Parameters

- **timeout** (*optional*) – Timeout to wait for a message to be available for reading. Defaults to -1 (blocking).
- **ack** (*optional*) – Primarily for internal use. When True, will not discard ACK messages. Defaults to False.
- **sync_check** (*optional*) – Validate message definition matches header version. Defaults to False.

Raises

ConnectionLost – Connection error to message manager server

Return type

Optional[*Message*]

Returns

Message object. If no message is read before timeout, returns None.

resume_all_subscriptions()

Resume all paused subscriptions

resume_subscription(*msg_list*)

Resume subscription to message types

Parameters

msg_list (*Iterable[int]*) – A list of paused message IDs to resubscribe to

send_message(*msg_data*, *dest_mod_id=0*, *dest_host_id=0*, *timeout=-1*)

Send a message

A message is a packet that contains a defined data payload. To send a message without associated data, see *send_signal()*.

Parameters

- **msg_data** (*MessageData*) – Object containing the message to send
- **dest_mod_id** (*optional*) – Specific module ID to send to. Defaults to 0 (broadcast).
- **dest_host_id** (*optional*) – Specific host ID to send to. Defaults to 0 (broadcast).
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

Raises

- ***InvalidDestinationModule*** – Specified destination module is invalid
- ***InvalidDestinationHost*** – Specified destination host is invalid

send_module_ready()

Send a signal to message manager that client is ready

This method also sends the client's process ID to message manager.

send_signal(*signal_type*, *dest_mod_id=0*, *dest_host_id=0*, *timeout=-1*)

Send a signal

A signal is a message type without an associated data payload. Only a unique message type ID is required to send a signal. To send a message with data, see *send_message()*.

Parameters

- **signal_type** (*int*) – Numeric message type ID of signal
- **dest_mod_id** (*optional*) – Specific module ID to send to. Defaults to 0 (broadcast).
- **dest_host_id** (*optional*) – Specific host ID to send to. Defaults to 0 (broadcast).
- **timeout** (*optional*) – Timeout in seconds to wait for socket to be available for sending. Defaults to -1 (blocking).

Raises

- ***InvalidDestinationModule*** – Specified destination module is invalid
- ***InvalidDestinationHost*** – Specified destination host is invalid

property server: Tuple[str, int]

Message manager server address as a (IP_addr, port_num) tuple

property sock: socket

Underlying socket connection with MessageManager

subscribe(*msg_list*)

Subscribe to message types

Calling this method multiple times will add to, and not replace, the list of subscribed messages.

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to subscribe to

property subscribed_types: Set[int]

List of subscribed message types

subscription_context(*msg_list*)

Context manager to subscribe to a list of message types

Message types will automatically unsubscribe after exiting context.

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to subscribe to

unsubscribe(*msg_list*)

Unsubscribe from message types

Parameters
msg_list (*Iterable[int]*) – A list of numeric message IDs to unsubscribe to

unsubscribe_from_all()

Unsubscribe from all subscribed types

29.7 pyrtma.web_manager.Message

class Message(*header*, *data*)

Bases: object

Message class

Contains message header and data

Methods

copy	Generate a copy of a message structure
from_json	Create message object from JSON string
pretty_print	Generate formatted string for pretty printing of message
to_dict	Convert message to dictionary
to_json	Convert message to JSON string

Attributes

name
type_id

Parameters

- **header** (MessageHeader) –
- **data** (MessageData) –

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*Message*) – Message structure to copy

Return type
Message

classmethod from_json(*s*)

Create message object from JSON string

Parameters
s (*str*) – JSON message string

Raises
InvalidMessageDefinition – JSON data does not match expected message defintion

Returns
Message object

Return type
Message

pretty_print(*add_tabs=0*)

Generate formatted string for pretty printing of message

Parameters
add_tabs (*int, optional*) – Indent level. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to JSON string

Parameters
minify (*bool, optional*) – Flag to minify (compact format). Defaults to False.

Returns
JSON message string

Return type
str

29.8 pyrtma.web_manager.MessageHeader

class MessageHeader

Bases: *MessageBase*

RTMA Message Header class

Methods

copy	Generate a copy of a message structure
from_dict	Generate message instance from dictionary
from_json	Generate message instance from JSON string
from_random	Generate message instance with random values
get_field_raw	return copy of raw bytes for ctypes field
hexdump	hexdump of message
pretty_print	Generate formatted message structure string for pretty printing
to_dict	Convert message to dictionary
to_json	Convert message to json string

Attributes

dest_host_id	Validator for 16-bit integers
dest_mod_id	Validator for 16-bit integers
is_dynamic	Validator for 32-bit integers
msg_count	Validator for 32-bit integers
msg_type	Validator for 32-bit integers
num_data_bytes	Validator for 32-bit integers
recv_time	Double (64-bit float) validator class
remaining_bytes	Validator for 32-bit integers
reserved	Validator for unsigned 32-bit integers
send_time	Double (64-bit float) validator class
size	
src_host_id	Validator for 16-bit integers
src_mod_id	Validator for 16-bit integers
version	

classmethod copy(*m*)

Generate a copy of a message structure

Parameters
m (*TypeVar*(MB, bound= MessageBase)) – Message structure to copy

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_dict(*data*)

Generate message instance from dictionary

Parameters
data (*Dict[str, Any]*) – Message dictionary

Raises
JSONDecodingError – Unable to decode dictionary

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_json(*s*)

Generate message instance from JSON string

Parameters
s (*str*) – Message JSON string

Return type
TypeVar(MB, bound= MessageBase)

classmethod from_random()

Generate message instance with random values

Return type
TypeVar(MB, bound= MessageBase)

get_field_raw(*name*)

return copy of raw bytes for ctypes field

Parameters
name (*str*) – Message fieldname

Raises
KeyError – Invalid fieldname

Returns
Copy of message field data bytes

Return type
bytes

hexdump(*length=16*, *sep=' '*)

hexdump of message

Parameters

- **length** (*int, optional*) – Row length. Defaults to 16.
- **sep** (*str, optional*) – Separator for non-printable ascii chars. Defaults to ” “.

pretty_print(*add_tabs=0*)

Generate formatted message structure string for pretty printing

Parameters
add_tabs (*int, optional*) – Indentation level, used for recursively calling. Defaults to 0.

Returns
Formatted string

Return type
str

to_dict()

Convert message to dictionary

Returns
Message dictionary

Return type
Dict[str, Any]

to_json(*minify=False*, ***kwargs*)

Convert message to json string

Parameters

- **minify** (*bool, optional*) – Flag to minify (compact format). Defaults to False.
- **json.dumps** (*kwargs for*) –

Returns
json string

Return type
str

29.9 pyrtma.web_manager.RTMAWebSocketHandler

class RTMAWebSocketHandler(*socket*, *addr*, *server*)

Bases: *WebSocketHandler*

RTMA Web Socket Handler class

RTMA Web Socket Handler

Initializes and handles RTMA Proxy connection

Parameters

- **socket** – socket object
- **addr** – client address
- **server** – server object

Methods

calculate_response_key	
finish	Close RTMA connection
handle	Handle RTMA proxy connection
handle_connect	
handshake	
make_handshake_response	
pong_received	Log that pong was received
process_json_message	Process incoming json message
read_bytes	
read_http_headers	
read_next_message	
read_ws_message	Read websocket message
send_close	Send CLOSE to client
send_failed_message	Send FAILED_MESSAGE when we cannot forward to websocket
send_message	
send_pong	
send_text	Important: Fragmented(=continuation) messages are not supported since their usage cases are limited - when we don't know the payload length.
setup	

Attributes

disable_nagle_algorithm
rbufsize
timeout
wbufsize
ws_ready_to_send

finish()

Close RTMA connection

handle()

Handle RTMA proxy connection

pong_received(*msg*)

Log that pong was received

Parameters
msg (*str*) – Ignored

process_json_message(*message*)

Process incoming json message

Called when a client receives a message over websocket

Parameters
message (*str*) – JSON message string

read_ws_message()

Read websocket message

Returns
Websocket message string

Return type
Optional[str]

send_close(*status=1000*, *reason=b''*)

Send CLOSE to client

Parameters

- **status** – Status as defined in https://datatracker.ietf.org/doc/html/rfc6455#section-7.4.1
- **reason** – Text with reason of closing the connection

send_failed_message(*header*, *time_of_failure*)

Send FAILED_MESSAGE when we cannot forward to websocket

Parameters

- **header** (*MessageHeader*) –
- **time_of_failure** (*float*) –

send_text(*message*, *opcode=1*)

Important: Fragmented(=continuation) messages are not supported since their usage cases are limited - when we don't know the payload length.

29.10 pyrtma.web_manager.RichHandler

class RichHandler(*level=0*, *console=None*, ***, *show_time=True*, *omit_repeated_times=True*, *show_level=True*, *show_path=True*, *enable_link_path=True*, *highlighter=None*, *markup=False*, *rich_tracebacks=False*, *tracebacks_width=None*, *tracebacks_extra_lines=3*, *tracebacks_theme=None*, *tracebacks_word_wrap=True*, *tracebacks_show_locals=False*, *tracebacks_suppress=()*, *locals_max_length=10*, *locals_max_string=80*, *log_time_format='[%x %X]'*, *keywords=None*)

Bases: Handler

A logging handler that renders output with Rich. The time / level / message and file are displayed in columns. The level is color coded, and the message is syntax highlighted.

Note: Be careful when enabling console markup in log messages if you have configured logging for libraries not under your control. If a dependency writes messages containing square brackets, it may not produce the intended output.

Parameters

- **level** (*Union[int, str], optional*) – Log level. Defaults to logging.NOTSET.
- **console** (Console, optional) – Optional console instance to write logs. Default will use a global console instance writing to stdout.
- **show_time** (*bool, optional*) – Show a column for the time. Defaults to True.
- **omit_repeated_times** (*bool, optional*) – Omit repetition of the same time. Defaults to True.
- **show_level** (*bool, optional*) – Show a column for the level. Defaults to True.
- **show_path** (*bool, optional*) – Show the path to the original log call. Defaults to True.
- **enable_link_path** (*bool, optional*) – Enable terminal link of path column to file. Defaults to True.
- **highlighter** (*Highlighter, optional*) – Highlighter to style log messages, or None to use ReprHighlighter. Defaults to None.
- **markup** (*bool, optional*) – Enable console markup in log messages. Defaults to False.
- **rich_tracebacks** (*bool, optional*) – Enable rich tracebacks with syntax highlighting and formatting. Defaults to False.
- **tracebacks_width** (*Optional[int], optional*) – Number of characters used to render tracebacks, or None for full width. Defaults to None.
- **tracebacks_extra_lines** (*int, optional*) – Additional lines of code to render tracebacks, or None for full width. Defaults to None.
- **tracebacks_theme** (*str, optional*) – Override pygments theme used in traceback.
- **tracebacks_word_wrap** (*bool, optional*) – Enable word wrapping of long tracebacks lines. Defaults to True.
- **tracebacks_show_locals** (*bool, optional*) – Enable display of locals in tracebacks. Defaults to False.
- **tracebacks_suppress** (*Sequence[Union[str, ModuleType]]*) – Optional sequence of modules or paths to exclude from traceback.
- **locals_max_length** (*int, optional*) – Maximum length of containers before abbreviating, or None for no abbreviation. Defaults to 10.
- **locals_max_string** (*int, optional*) – Maximum length of string before truncating, or None to disable. Defaults to 80.
- **log_time_format** (*Union[str, TimeFormatterCallable], optional*) – If log_time is enabled, either string for strftime or callable that formats the time. Defaults to "[%x %X] ".

- **keywords** (*List[str], optional*) – List of words to highlight instead of `RichHandler.KEYWORDS`.

Initializes the instance - basically setting the formatter to None and the filter list to empty.

Methods

acquire	Acquire the I/O thread lock.
addFilter	Add the specified filter to this handler.
close	Tidy up any resources used by the handler.
createLock	Acquire a thread lock for serializing access to the underlying I/O.
emit	Invoked by logging.
filter	Determine if a record is loggable by consulting all the filters.
flush	Ensure all logging output has been flushed.
format	Format the specified record.
get_level_text	Get the level name from the record.
get_name	
handle	Conditionally emit the specified logging record.
handleError	Handle errors which occur during an emit() call.
release	Release the I/O thread lock.
removeFilter	Remove the specified filter from this handler.
render	Render log for display.
render_message	Render message text in to Text.
setFormatter	Set the formatter for this handler.
setLevel	Set the logging level of this handler.
set_name	

Attributes

KEYWORDS
name

HIGHLIGHTER_CLASS

alias of `ReprHighlighter`

acquire()

Acquire the I/O thread lock.

addFilter(*filter*)

Add the specified filter to this handler.

close()

Tidy up any resources used by the handler.

This version removes the handler from an internal map of handlers, _handlers, which is used for handler lookup by name. Subclasses should ensure that this gets called from overridden close() methods.

createLock()

Acquire a thread lock for serializing access to the underlying I/O.

emit(*record*)

Invoked by logging.

Return type
None

Parameters
record (*LogRecord*) –

filter(*record*)

Determine if a record is loggable by consulting all the filters.

The default is to allow the record to be logged; any filter can veto this and the record is then dropped. Returns a zero value if a record is to be dropped, else non-zero.

Changed in version 3.2: Allow filters to be just callables.

flush()

Ensure all logging output has been flushed.

This version does nothing and is intended to be implemented by subclasses.

format(*record*)

Format the specified record.

If a formatter is set, use it. Otherwise, use the default formatter for the module.

get_level_text(*record*)

Get the level name from the record.

Parameters
record (*LogRecord*) – LogRecord instance.

Returns
A tuple of the style and level name.

Return type
Text

handle(*record*)

Conditionally emit the specified logging record.

Emission depends on filters which may have been added to the handler. Wrap the actual emission of the record with acquisition/release of the I/O thread lock. Returns whether the filter passed the record for emission.

handleError(*record*)

Handle errors which occur during an emit() call.

This method should be called from handlers when an exception is encountered during an emit() call. If raiseExceptions is false, exceptions get silently ignored. This is what is mostly wanted for a logging system - most users will not care about errors in the logging system, they are more interested in application errors. You could, however, replace this with a custom handler if you wish. The record which was being processed is passed in to this method.

release()

Release the I/O thread lock.

removeFilter(*filter*)

Remove the specified filter from this handler.

render(*, *record*, *traceback*, *message_renderable*)

Render log for display.

Parameters

- **record** (*LogRecord*) – logging Record.
- **traceback** (*Optional[Traceback]*) – Traceback instance or None for no Traceback.
- **message_renderable** (*ConsoleRenderable*) – Renderable (typically Text) containing log message contents.

Returns
Renderable to display log.

Return type
ConsoleRenderable

render_message(*record*, *message*)

Render message text in to Text.

Parameters

- **record** (*LogRecord*) – logging Record.
- **message** (*str*) – String containing log message.

Returns
Renderable to display log message.

Return type
ConsoleRenderable

setFormatter(*fmt*)

Set the formatter for this handler.

setLevel(*level*)

Set the logging level of this handler. level must be an int or a str.

29.11 pyrtma.web_manager.TCPServer

class TCPServer(*server_address*, *RequestHandlerClass*, *bind_and_activate=True*)

Bases: BaseServer

Base class for various socket-based server classes.

Defaults to synchronous IP stream (i.e., TCP).

Methods for the caller:

- __init__(server_address, RequestHandlerClass, bind_and_activate=True)
- serve_forever(poll_interval=0.5)
- shutdown()

- handle_request() # if you don't use serve_forever()
- fileno() -> int # for selector

Methods that may be overridden:

- server_bind()
- server_activate()
- get_request() -> request, client_address
- handle_timeout()
- verify_request(request, client_address)
- process_request(request, client_address)
- shutdown_request(request)
- close_request(request)
- handle_error()

Methods for derived classes:

- finish_request(request, client_address)

Class variables that may be overridden by derived classes or instances:

- timeout
- address_family
- socket_type
- request_queue_size (only for stream sockets)
- allow_reuse_address

Instance variables:

- server_address
- RequestHandlerClass
- socket

Constructor. May be extended, do not override.

Methods

close_request	Called to clean up an individual request.
fileno	Return socket file number.
finish_request	Finish one request by instantiating RequestHandler-Class.
get_request	Get the request and client address from the socket.
handle_error	Handle an error gracefully.
handle_request	Handle one request, possibly blocking.
handle_timeout	Called if no new request arrives within self.timeout.
process_request	Call finish_request.
serve_forever	Handle one request at a time until shutdown.
server_activate	Called by constructor to activate the server.
server_bind	Called by constructor to bind the socket.
server_close	Called to clean-up the server.
service_actions	Called by the serve_forever() loop.
shutdown	Stops the serve_forever loop.
shutdown_request	Called to shutdown and close an individual request.
verify_request	Verify the request.

Attributes

address_family
allow_reuse_address
request_queue_size
socket_type
timeout

close_request(*request*)

Called to clean up an individual request.

fileno()

Return socket file number.

Interface required by selector.

finish_request(*request*, *client_address*)

Finish one request by instantiating RequestHandlerClass.

get_request()

Get the request and client address from the socket.

May be overridden.

handle_error(*request*, *client_address*)

Handle an error gracefully. May be overridden.

The default is to print a traceback and continue.

handle_request()

Handle one request, possibly blocking.

Respects self.timeout.

handle_timeout()

Called if no new request arrives within self.timeout.

Overridden by ForkingMixIn.

process_request(*request*, *client_address*)

Call finish_request.

Overridden by ForkingMixIn and ThreadingMixIn.

serve_forever(*poll_interval=0.5*)

Handle one request at a time until shutdown.

Polls for shutdown every poll_interval seconds. Ignores self.timeout. If you need to do periodic tasks, do them in another thread.

server_activate()

Called by constructor to activate the server.

May be overridden.

server_bind()

Called by constructor to bind the socket.

May be overridden.

server_close()

Called to clean-up the server.

May be overridden.

service_actions()

Called by the serve_forever() loop.

May be overridden by a subclass / Mixin to implement any code that needs to be run during the loop.

shutdown()

Stops the serve_forever loop.

Blocks until the loop has finished. This must be called while serve_forever() is running in another thread, or it will deadlock.

shutdown_request(*request*)

Called to shutdown and close an individual request.

verify_request(*request*, *client_address*)

Verify the request. May be overridden.

Return True if we should proceed with this request.

29.12 pyrtma.web_manager.WebMessageManager

class WebMessageManager(*host=''*, *port=0*, *mm_ip='127.0.0.1:7111'*, *loglevel=30*, *key=None*, *cert=None*)

Bases: *WebsocketServer*

WebMessageManager class

WebMessageManager class

Parameters

- **host** (*str, optional*) – IP for WebMessageManager to listen for connections. Defaults to "" (any local IP).
- **port** (*int, optional*) – Port for WebMessageManager to bind to. Defaults to 0.
- **mm_ip** (*str, optional*) – Address for RTMA MessageManager. Defaults to "127.0.0.1:7111".
- **loglevel** (*int, optional*) – Loging level. Defaults to logging.WARNING.
- **key** (*optional*) – Path to SSL key. Defaults to None.
- **cert** (*optional*) – Path to SSL cert. Defaults to None.

Methods

allow_new_connections	
client_left	
close_request	Called to clean up an individual request.
deny_new_connections	
disconnect_clients_abruptly	
disconnect_clients_gracefully	
fileno	Return socket file number.
finish_request	Finish one request by instantiating RequestHandler-Class.
get_request	Get the request and client address from the socket.
handle_error	Handle an error gracefully.
handle_request	Handle one request, possibly blocking.
handle_timeout	Called if no new request arrives within self.timeout.
handler_to_client	
message_received	
new_client	
process_request	Start a new thread to process the request.
process_request_thread	Same as in BaseServer but as a thread.

continues on next page

Table 1 – continued from previous page

run_forever	
send_message	
send_message_to_all	
serve_forever	Handle one request at a time until shutdown.
server_activate	Called by constructor to activate the server.
server_bind	Called by constructor to bind the socket.
server_close	Called to clean-up the server.
service_actions	Called by the serve_forever() loop.
set_fn_client_left	
set_fn_message_received	
set_fn_new_client	
shutdown	Stops the serve_forever loop.
shutdown_abruptly	
shutdown_gracefully	
shutdown_request	Called to shutdown and close an individual request.
verify_request	Verify the request.

Attributes

address_family
allow_reuse_address
block_on_close
daemon_threads
request_queue_size
socket_type
timeout

close_request(*request*)

Called to clean up an individual request.

fileno()

Return socket file number.

Interface required by selector.

finish_request(*request*, *client_address*)

Finish one request by instantiating RequestHandlerClass.

get_request()

Get the request and client address from the socket.

May be overridden.

handle_error(*request*, *client_address*)

Handle an error gracefully. May be overridden.

The default is to print a traceback and continue.

handle_request()

Handle one request, possibly blocking.

Respects self.timeout.

handle_timeout()

Called if no new request arrives within self.timeout.

Overridden by ForkingMixIn.

process_request(*request*, *client_address*)

Start a new thread to process the request.

process_request_thread(*request*, *client_address*)

Same as in BaseServer but as a thread.

In addition, exception handling is done here.

serve_forever(*poll_interval=0.5*)

Handle one request at a time until shutdown.

Polls for shutdown every poll_interval seconds. Ignores self.timeout. If you need to do periodic tasks, do them in another thread.

server_activate()

Called by constructor to activate the server.

May be overridden.

server_bind()

Called by constructor to bind the socket.

May be overridden.

server_close()

Called to clean-up the server.

May be overridden.

service_actions()

Called by the serve_forever() loop.

May be overridden by a subclass / Mixin to implement any code that needs to be run during the loop.

shutdown()

Stops the serve_forever loop.

Blocks until the loop has finished. This must be called while serve_forever() is running in another thread, or it will deadlock.

shutdown_request(*request*)

Called to shutdown and close an individual request.

verify_request(*request*, *client_address*)

Verify the request. May be overridden.

Return True if we should proceed with this request.

29.13 pyrtma.web_manager.WebSocketHandler

class WebSocketHandler(*socket*, *addr*, *server*)

Bases: StreamRequestHandler

Methods

calculate_response_key	
finish	
handle	
handshake	
make_handshake_response	
read_bytes	
read_http_headers	
read_next_message	
send_close	Send CLOSE to client
send_message	
send_pong	
send_text	Important: Fragmented(=continuation) messages are not supported since their usage cases are limited - when we don't know the payload length.
setup	

Attributes

disable_nagle_algorithm
rbufsize
timeout
wbufsize

send_close(*status=1000*, *reason=b''*)

Send CLOSE to client

Parameters

- **status** – Status as defined in https://datatracker.ietf.org/doc/html/rfc6455#section-7.4.1
- **reason** – Text with reason of closing the connection

send_text(*message*, *opcode=1*)

Important: Fragmented(=continuation) messages are not supported since their usage cases are limited - when we don't know the payload length.

29.14 pyrtma.web_manager.WebsocketServer

class WebsocketServer(*host='127.0.0.1'*, *port=0*, *loglevel=30*, *key=None*, *cert=None*)

Bases: ThreadingMixIn, *TCPServer*, API

A websocket server waiting for clients to connect.

Parameters

- **port** (*int*) – Port to bind to
- **host** (*str*) – Hostname or IP to listen for connections. By default 127.0.0.1 is being used. To accept connections from any client, you should use 0.0.0.0.
- **loglevel** – Logging level from logging module to use for logging. By default warnings and errors are being logged.

Properties:

clients(list): A list of connected clients. A client is a dictionary

like below.

{
'id' : id, 'handler' : handler, 'address' : (addr, port)

}

Methods

allow_new_connections	
client_left	
close_request	Called to clean up an individual request.
deny_new_connections	
disconnect_clients_abruptly	
disconnect_clients_gracefully	
fileno	Return socket file number.
finish_request	Finish one request by instantiating RequestHandler-Class.
get_request	Get the request and client address from the socket.
handle_error	Handle an error gracefully.
handle_request	Handle one request, possibly blocking.
handle_timeout	Called if no new request arrives within self.timeout.
handler_to_client	
message_received	
new_client	
process_request	Start a new thread to process the request.
process_request_thread	Same as in BaseServer but as a thread.
run_forever	
send_message	
send_message_to_all	
serve_forever	Handle one request at a time until shutdown.
server_activate	Called by constructor to activate the server.
server_bind	Called by constructor to bind the socket.
server_close	Called to clean-up the server.
service_actions	Called by the serve_forever() loop.
set_fn_client_left	
set_fn_message_received	
set_fn_new_client	
shutdown	Stops the serve_forever loop.
shutdown_abruptly	
shutdown_gracefully	
shutdown_request	Called to shutdown and close an individual request.

continues on next page

Table 2 – continued from previous page

verify_request	Verify the request.

Attributes

address_family
allow_reuse_address
block_on_close
daemon_threads
request_queue_size
socket_type
timeout

close_request(*request*)

Called to clean up an individual request.

fileno()

Return socket file number.

Interface required by selector.

finish_request(*request*, *client_address*)

Finish one request by instantiating RequestHandlerClass.

get_request()

Get the request and client address from the socket.

May be overridden.

handle_error(*request*, *client_address*)

Handle an error gracefully. May be overridden.

The default is to print a traceback and continue.

handle_request()

Handle one request, possibly blocking.

Respects self.timeout.

handle_timeout()

Called if no new request arrives within self.timeout.

Overridden by ForkingMixIn.

process_request(*request*, *client_address*)

Start a new thread to process the request.

process_request_thread(*request*, *client_address*)

Same as in BaseServer but as a thread.

In addition, exception handling is done here.

serve_forever(*poll_interval=0.5*)

Handle one request at a time until shutdown.

Polls for shutdown every poll_interval seconds. Ignores self.timeout. If you need to do periodic tasks, do them in another thread.

server_activate()

Called by constructor to activate the server.

May be overridden.

server_bind()

Called by constructor to bind the socket.

May be overridden.

server_close()

Called to clean-up the server.

May be overridden.

service_actions()

Called by the serve_forever() loop.

May be overridden by a subclass / Mixin to implement any code that needs to be run during the loop.

shutdown()

Stops the serve_forever loop.

Blocks until the loop has finished. This must be called while serve_forever() is running in another thread, or it will deadlock.

shutdown_request(*request*)

Called to shutdown and close an individual request.

verify_request(*request*, *client_address*)

Verify the request. May be overridden.

Return True if we should proceed with this request.

Exceptions

ClientError	Base exception for all Client Errors.
RTMAMessageError	Base exception for message errors.
SocketError	alias of OSError

29.15 pyrtma.web_manager.ClientError

exception ClientError

Base exception for all Client Errors.

29.16 pyrtma.web_manager.RTMAMessageError

exception RTMAMessageError

Base exception for message errors.

29.17 pyrtma.web_manager.SocketError

SocketError

alias of OSError

CHAPTER

THIRTY

PYRTMA

RTMA/Dragonfly client written in python with no external dependencies. Based on and compatible with Dragonfly Messaging

30.1 Installation

pyrtma is available on PyPI

```
$ pip install pyrtma
```

30.1.1 Installing for pyrtma development

This is only necessary for individuals who would like to contribute to pyrtma.

```
$ pip install --upgrade pip setuptools
$ pip install -e .
```

30.2 Usage

30.2.1 Launch Manager

```
$ python -m pyrtma.manager -a "127.0.0.1"
```

30.2.2 Create a message in message.yaml

Message definitions are created in a .yaml file.

The ruamel.yaml parser library is used internally (https://yaml.readthedocs.io/en/latest/)

Notes about yaml format:

- Whitespace sensitive. Use either 2 or 4 spaces for tab not '\t'
- Key-values must be separated by a colon followed by a space, (Key: Value, not Key:Value)
- Must follow the top-level headers shown below.
- Unused sections should be marked `null`

- Names must start with letter. (no _ or numeric prefixes allowed)
- Use **(.yaml)** extension not **(.yml)**

List of supported native data types:

- char
- unsigned char
- byte
- int
- signed int
- unsigned int
- short
- unsigned short
- long
- signed long
- unsigned long
- long long
- signed long long
- unsigned long long
- float
- double
- int8
- uint8
- int16
- uint16
- int32
- uint32
- int64
- uint64

Below is an example:

```
# message.yaml

imports: null

# Constant values and expressions
constants:
    STR_SIZE: 32
    LONG_STRING: STR_SIZE * 2

# Constant string values
string_constants:
```

(continues on next page)

(continued from previous page)

```
    default_msg: "hello_world"

host_ids: null

module_ids:
    PERSON_PUBLISHER: 212
    PERSON_SUBSCRIBER: 214

# Alias a type by another name
aliases:
    AGE_TYPE: int

# Non-message structured data (no id field)
struct_defs:
    TEST_STRUCT:
        fields:
            value_str: char[STR_SIZE]
            value_int: int

# Message defintions with user assigned id field
message_defs:
    PERSON_MESSAGE:
        id: 1234
        fields:
            name: char[STR_SIZE]
            age: AGE_TYPE

    ANOTHER_EXAMPLE:
        id: 5678
        fields:
            value_struct: TEST_STRUCT
            value_float: float
            value_double: double

    # Example signal definition
    USER_SIGNAL:
        id: 2468
        fields: null

    # Example using a nested message defintion
    PERSON_LIST:
        id: 1357
        fields:
            person: PERSON_MESSAGE[32]

    # Example reusing a message definition by another name
    EMPLOYEES:
        id: 1368
        fields: PERSON_LIST

    # A block of message ids can be reserved by a file for future use
    # Ranges are inclusive on both ends
```

(continues on next page)

(continued from previous page)

```
    _RESERVED_:
        id: [1000, 1002:1005, 1006 - 1008, 1009 to 1012]
```

Run the following command to compile the yaml file into Python, C, Matlab, or Javascript files. This will output a message.(py|h|m|js) file.

```
python -m pyrtma.compile -i examples/msg_defs/message.yaml --py --c --mat --js
```

The msg_defs directory should now have message def files created for each language.

The rtma objects are compiled into objects suitable for each language.

30.3 Examples

See /examples/example.py for pub/sub demo app

Compile the example message defintions:

```
python -m pyrtma.compile -i ./examples/msg_defs/message.yaml --py
```

Start the demo MessageManager server

```
python -m pyrtma.manager
```

Start the publisher in one console:

```
python ./examples/example.py --pub
```

Start the subscriber in another:

```
python ./examples/example.py --sub
```

PYTHON MODULE INDEX

p

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y