
pyrtma
Release 2.2.4

RNEL

Apr 18, 2024

CONTENTS

1	pyrtma.client_context	3
2	pyrtma.get_header_cls	5
3	pyrtma.get_msg_cls	7
4	pyrtma.message_def	9
5	pyrtma.msg_def	11
6	pyrtma.Client	13
7	pyrtma.Message	19
8	pyrtma.MessageData	21
9	pyrtma.MessageHeader	25
10	pyrtma.AcknowledgementTimeout	29
11	pyrtma.ClientError	31
12	pyrtma.ConnectionLost	33
13	pyrtma.InvalidDestinationHost	35
14	pyrtma.InvalidDestinationModule	37
15	pyrtma.MessageManagerNotFound	39
16	pyrtma.NotConnectedError	41
17	pyrtma.client	43
17.1	pyrtma.client.cast	43
17.2	pyrtma.client.client_context	43
17.3	pyrtma.client.contextmanager	44
17.4	pyrtma.client.get_header_cls	45
17.5	pyrtma.client.get_msg_cls	45
17.6	pyrtma.client.requires_connection	45
17.7	pyrtma.client.warn	45
17.8	pyrtma.client.wraps	46
17.9	pyrtma.client.Client	46

17.10	pyrtma.client.Message	51
17.11	pyrtma.client.MessageData	52
17.12	pyrtma.client.MessageHeader	55
17.13	pyrtma.client.TypeVar	57
17.14	pyrtma.client.AcknowledgementTimeout	58
17.15	pyrtma.client.ClientError	58
17.16	pyrtma.client.ConnectionLost	58
17.17	pyrtma.client.InvalidDestinationHost	58
17.18	pyrtma.client.InvalidDestinationModule	59
17.19	pyrtma.client.InvalidMessageDefinition	59
17.20	pyrtma.client.MessageManagerNotFound	59
17.21	pyrtma.client.NotConnectedError	59
17.22	pyrtma.client.SocketOptionError	59
17.23	pyrtma.client.UnknownMessageType	59
18	pyrtma.compile	61
18.1	pyrtma.compile.compile	61
18.2	pyrtma.compile.install	62
18.3	pyrtma.compile.main	63
18.4	pyrtma.compile.CDefCompiler	63
18.5	pyrtma.compile.InfoCompiler	65
18.6	pyrtma.compile.JSDefCompiler	65
18.7	pyrtma.compile.MatlabDefCompiler	66
18.8	pyrtma.compile.Parser	69
18.9	pyrtma.compile.PyDefCompiler	71
18.10	pyrtma.compile.YAMLCompiler	73
18.11	pyrtma.compile.FileFormatError	73
18.12	pyrtma.compile.ParserError	73
19	pyrtma.constants	75
19.1	pyrtma.constants.warn	75
19.2	pyrtma.constants.HOST_ID	75
19.3	pyrtma.constants.MODULE_ID	76
19.4	pyrtma.constants.MSG_COUNT	76
19.5	pyrtma.constants.MSG_TYPE	76
20	pyrtma.core_defs	77
20.1	pyrtma.core_defs.check_compiled_version	77
20.2	pyrtma.core_defs.get_context	77
20.3	pyrtma.core_defs.Byte	79
20.4	pyrtma.core_defs.ByteArray	80
20.5	pyrtma.core_defs.Char	81
20.6	pyrtma.core_defs.Double	81
20.7	pyrtma.core_defs.Float	82
20.8	pyrtma.core_defs.FloatArray	83
20.9	pyrtma.core_defs.HOST_ID	84
20.10	pyrtma.core_defs.Int16	84
20.11	pyrtma.core_defs.Int32	85
20.12	pyrtma.core_defs.Int64	86
20.13	pyrtma.core_defs.Int8	87
20.14	pyrtma.core_defs.IntArray	88
20.15	pyrtma.core_defs.MDF_ACKNOWLEDGE	89
20.16	pyrtma.core_defs.MDF_CONNECT	91
20.17	pyrtma.core_defs.MDF_DISCONNECT	94

20.18	pyrtma.core_defs.MDF_DUMP_MESSAGE_LOG	96
20.19	pyrtma.core_defs.MDF_EXIT	99
20.20	pyrtma.core_defs.MDF_FAILED_MESSAGE	101
20.21	pyrtma.core_defs.MDF_FAIL_SUBSCRIBE	104
20.22	pyrtma.core_defs.MDF_FORCE_DISCONNECT	106
20.23	pyrtma.core_defs.MDF_KILL	109
20.24	pyrtma.core_defs.MDF_LM_EXIT	111
20.25	pyrtma.core_defs.MDF_LM_READY	114
20.26	pyrtma.core_defs.MDF_MESSAGE_LOG_SAVED	116
20.27	pyrtma.core_defs.MDF_MODULE_READY	119
20.28	pyrtma.core_defs.MDF_PAUSE_MESSAGE_LOGGING	121
20.29	pyrtma.core_defs.MDF_PAUSE_SUBSCRIPTION	124
20.30	pyrtma.core_defs.MDF_RESET_MESSAGE_LOG	126
20.31	pyrtma.core_defs.MDF_RESUME_MESSAGE_LOGGING	129
20.32	pyrtma.core_defs.MDF_RESUME_SUBSCRIPTION	131
20.33	pyrtma.core_defs.MDF_SAVE_MESSAGE_LOG	134
20.34	pyrtma.core_defs.MDF_SUBSCRIBE	136
20.35	pyrtma.core_defs.MDF_TIMING_MESSAGE	139
20.36	pyrtma.core_defs.MDF_UNSUBSCRIBE	141
20.37	pyrtma.core_defs.MODULE_ID	144
20.38	pyrtma.core_defs.MSG_COUNT	144
20.39	pyrtma.core_defs.MSG_TYPE	144
20.40	pyrtma.core_defs.MessageBase	144
20.41	pyrtma.core_defs.MessageData	146
20.42	pyrtma.core_defs.MessageMeta	149
20.43	pyrtma.core_defs.RTMA_MSG_HEADER	150
20.44	pyrtma.core_defs.String	152
20.45	pyrtma.core_defs.Struct	153
20.46	pyrtma.core_defs.StructArray	153
20.47	pyrtma.core_defs.Uint16	155
20.48	pyrtma.core_defs.Uint32	156
20.49	pyrtma.core_defs.Uint64	157
20.50	pyrtma.core_defs.Uint8	158
21	pyrtma.exceptions	159
21.1	pyrtma.exceptions.AcknowledgementTimeout	159
21.2	pyrtma.exceptions.ClientError	159
21.3	pyrtma.exceptions.ConnectionLost	160
21.4	pyrtma.exceptions.InvalidDestinationHost	160
21.5	pyrtma.exceptions.InvalidDestinationModule	160
21.6	pyrtma.exceptions.InvalidMessageDefinition	160
21.7	pyrtma.exceptions.JSONDecodingError	160
21.8	pyrtma.exceptions.MessageManagerNotFound	160
21.9	pyrtma.exceptions.NotConnectedError	160
21.10	pyrtma.exceptions.RTMAMessageError	161
21.11	pyrtma.exceptions.SocketOptionError	161
21.12	pyrtma.exceptions.UnknownMessageType	161
21.13	pyrtma.exceptions.VersionMismatchWarning	161
22	pyrtma.header	163
22.1	pyrtma.header.get_header_cls	163
22.2	pyrtma.header.Double	164
22.3	pyrtma.header.HOST_ID	164
22.4	pyrtma.header.Int16	164

22.5	pyrtma.header.Int32	165
22.6	pyrtma.header.MODULE_ID	166
22.7	pyrtma.header.MSG_COUNT	167
22.8	pyrtma.header.MSG_TYPE	167
22.9	pyrtma.header.MessageBase	167
22.10	pyrtma.header.MessageHeader	169
22.11	pyrtma.header.MessageMeta	172
22.12	pyrtma.header.TimeCodeMessageHeader	172
22.13	pyrtma.header.Uint32	175
23	pyrtma.manager	177
23.1	pyrtma.manager.dataclass	177
23.2	pyrtma.manager.get_header_cls	177
23.3	pyrtma.manager.get_msg_cls	178
23.4	pyrtma.manager.main	178
23.5	pyrtma.manager.Counter	178
23.6	pyrtma.manager.Message	182
23.7	pyrtma.manager.MessageData	183
23.8	pyrtma.manager.MessageHeader	186
23.9	pyrtma.manager.MessageManager	188
23.10	pyrtma.manager.Module	192
23.11	pyrtma.manager.defaultdict	193
24	pyrtma.message	197
24.1	pyrtma.message.clear_msg_defs	197
24.2	pyrtma.message.get_header_cls	197
24.3	pyrtma.message.get_msg_cls	198
24.4	pyrtma.message.get_msg_defs	198
24.5	pyrtma.message.message_def	198
24.6	pyrtma.message.msg_def	198
24.7	pyrtma.message.set_msg_defs	199
24.8	pyrtma.message.update_msg_defs	199
24.9	pyrtma.message.Message	199
24.10	pyrtma.message.MessageData	201
24.11	pyrtma.message.MessageHeader	203
24.12	pyrtma.message.RTMJSONEncoder	206
24.13	pyrtma.message.TypeVar	207
24.14	pyrtma.message.InvalidMessageDefinition	208
24.15	pyrtma.message.UnknownMessageType	208
25	pyrtma.message_base	209
25.1	pyrtma.message_base.hexdump	209
25.2	pyrtma.message_base.is_dataclass	209
25.3	pyrtma.message_base.print_ctype_array	209
25.4	pyrtma.message_base.CStructType	210
25.5	pyrtma.message_base.MessageBase	210
25.6	pyrtma.message_base.MessageMeta	212
25.7	pyrtma.message_base.RTMJSONEncoder	213
25.8	pyrtma.message_base.TypeVar	214
25.9	pyrtma.message_base.JSONDecodingError	215
26	pyrtma.message_data	217
26.1	pyrtma.message_data.MessageBase	217
26.2	pyrtma.message_data.MessageData	219
26.3	pyrtma.message_data.MessageMeta	222

27	pyrtma.parser	223
27.1	pyrtma.parser.asdict	223
27.2	pyrtma.parser.copy	224
27.3	pyrtma.parser.dataclass	224
27.4	pyrtma.parser.field	224
27.5	pyrtma.parser.is_dataclass	224
27.6	pyrtma.parser.sha256	224
27.7	pyrtma.parser.CompilerOptions	225
27.8	pyrtma.parser.ConstantExpr	226
27.9	pyrtma.parser.ConstantString	227
27.10	pyrtma.parser.CustomEncoder	227
27.11	pyrtma.parser.Field	229
27.12	pyrtma.parser.HID	230
27.13	pyrtma.parser.Import	230
27.14	pyrtma.parser.MDF	231
27.15	pyrtma.parser.MID	232
27.16	pyrtma.parser.MT	232
27.17	pyrtma.parser.Metadata	233
27.18	pyrtma.parser.NativeType	234
27.19	pyrtma.parser.Parser	234
27.20	pyrtma.parser.SDF	237
27.21	pyrtma.parser.TypeAlias	237
27.22	pyrtma.parser.YAML	238
27.23	pyrtma.parser.AlignmentError	243
27.24	pyrtma.parser.CircularRefError	243
27.25	pyrtma.parser.DuplicateNameError	243
27.26	pyrtma.parser.ExpressionExpansionError	243
27.27	pyrtma.parser.FileFormatError	244
27.28	pyrtma.parser.HostIDError	244
27.29	pyrtma.parser.InvalidMessageSize	244
27.30	pyrtma.parser.InvalidTypeError	244
27.31	pyrtma.parser.MessageIDError	244
27.32	pyrtma.parser.ModuleIDError	244
27.33	pyrtma.parser.ParserError	244
27.34	pyrtma.parser.RTMASyntaxError	245
27.35	pyrtma.parser.RecurisionError	245
27.36	pyrtma.parser.YAMLSyntaxError	245
28	pyrtma.validators	247
28.1	pyrtma.validators.abstractmethod	247
28.2	pyrtma.validators.contextmanager	247
28.3	pyrtma.validators.disable_message_validation	248
28.4	pyrtma.validators.overload	248
28.5	pyrtma.validators.ABCMeta	249
28.6	pyrtma.validators.ArrayField	250
28.7	pyrtma.validators.Byte	251
28.8	pyrtma.validators.ByteArray	252
28.9	pyrtma.validators.Char	253
28.10	pyrtma.validators.ContextVar	253
28.11	pyrtma.validators.Double	254
28.12	pyrtma.validators.FieldValidator	255
28.13	pyrtma.validators.Float	255
28.14	pyrtma.validators.FloatArray	256
28.15	pyrtma.validators.FloatValidatorBase	257

28.16	pyrtma.validators.Generic	258
28.17	pyrtma.validators.Int16	258
28.18	pyrtma.validators.Int32	259
28.19	pyrtma.validators.Int64	260
28.20	pyrtma.validators.Int8	261
28.21	pyrtma.validators.IntArray	262
28.22	pyrtma.validators.IntValidatorBase	263
28.23	pyrtma.validators.MessageBase	264
28.24	pyrtma.validators.String	267
28.25	pyrtma.validators.Struct	267
28.26	pyrtma.validators.StructArray	268
28.27	pyrtma.validators.TypeVar	269
28.28	pyrtma.validators.Uint16	270
28.29	pyrtma.validators.Uint32	271
28.30	pyrtma.validators.Uint64	272
28.31	pyrtma.validators.Uint8	273
29	pyrtma.web_manager	275
29.1	pyrtma.web_manager.cast	275
29.2	pyrtma.web_manager.get_msg_cls	275
29.3	pyrtma.web_manager.main	276
29.4	pyrtma.web_manager.ws_client_connect	276
29.5	pyrtma.web_manager.ws_client_disconnect	276
29.6	pyrtma.web_manager.Client	277
29.7	pyrtma.web_manager.Message	281
29.8	pyrtma.web_manager.MessageHeader	283
29.9	pyrtma.web_manager.RTMASocketHandler	285
29.10	pyrtma.web_manager.RichHandler	287
29.11	pyrtma.web_manager.TCPServer	291
29.12	pyrtma.web_manager.WebMessageManager	295
29.13	pyrtma.web_manager.WebSocketHandler	298
29.14	pyrtma.web_manager.WebsocketServer	299
29.15	pyrtma.web_manager.ClientError	303
29.16	pyrtma.web_manager.RTMAMessageError	303
29.17	pyrtma.web_manager.SocketError	303
30	pyrtma	305
30.1	Installation	305
30.2	Usage	305
30.3	Examples	308
	Python Module Index	309
	Index	311

Functions

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

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

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*]

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*]

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]`) –

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

<i>Client</i>	RTMA Client interface
<i>Message</i>	Message class
<i>MessageData</i>	MessageData base class
<i>MessageHeader</i>	RTMA Message Header class

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

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

Attributes

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

PYRTMA.MESSAGE

class Message(*header, data*)

Bases: `object`

Message class

Contains message header and data

Methods

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

Attributes

<code>name</code>
<code>type_id</code>

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 definition

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

PYRTMA.MESSAGEDATA

class MessageData

Bases: *MessageBase*

MessageData base class

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

Methods

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

Attributes

<i>size</i>
<i>type_def</i>
<i>type_id</i>
<i>type_name</i>
<i>type_size</i>
<i>type_source</i>
<i>type_hash</i>

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

PYRTMA.MESSAGEHEADER

class MessageHeader

Bases: *MessageBase*

RTMA Message Header class

Methods

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

Attributes

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

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

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

PYRTMA.ACKNOWLEDGEMENTTIMEOUT

exception AcknowledgementTimeout

Raised when client does not receive ack from message manager.

PYRTMA.CLIENTERROR

exception ClientError

Base exception for all Client Errors.

PYRTMA.CONNECTIONLOST

exception `ConnectionLost`

Raised when there is a connection error with the server.

PYRTMA.INVALIDDESTINATIONHOST

exception `InvalidDestinationHost`

Raised when client tries to send to an invalid host.

PYRTMA.INVALIDDESTINATIONMODULE

exception InvalidDestinationModule

Raised when client tries to send to an invalid module.

PYRTMA.MESSAGEMANAGERNOTFOUND

exception `MessageManagerNotFound`

Raised when unable to connect to message manager.

PYRTMA.NOTCONNECTEDERROR

exception `NotConnectedError`

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

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

PYRTMA.CLIENT

pyrtma.client module

Includes *Client* class and associated exception classes

Functions

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

<i>Client</i>	RTMA Client interface
<i>Message</i>	Message class
<i>MessageData</i>	MessageData base class
<i>MessageHeader</i>	RTMA Message Header class
<i>TypeVar</i>	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

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

Attributes

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

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

Attributes

<code>name</code>
<code>type_id</code>

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 definition

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 should not be directly instantiated.

Methods

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

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

Attributes

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

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

<i>AcknowledgementTimeout</i>	Raised when client does not receive ack from message manager.
<i>ClientError</i>	Base exception for all Client Errors.
<i>ConnectionLost</i>	Raised when there is a connection error with the server.
<i>InvalidDestinationHost</i>	Raised when client tries to send to an invalid host.
<i>InvalidDestinationModule</i>	Raised when client tries to send to an invalid module.
<i>InvalidMessageDefinition</i>	Raised when there is message definition is out of sync with sent data.
<i>MessageManagerNotFound</i>	Raised when unable to connect to message manager.
<i>NotConnectedError</i>	Raised when the client tries to read/write while not connected.
<i>SocketOptionError</i>	Raised when unable to set socket options.
<i>UnknownMessageType</i>	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.

PYRTMA.COMPILE

pyrtma.compile Message Type Compiler

Functions

<i>compile</i>	compile message defs
<i>install</i>	Install a rich traceback handler.
<i>main</i>	

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 definition 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

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

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

<i>FileFormatError</i>	Raised when the wrong file extension is referenced.
<i>ParserError</i>	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

PYRTMA.CONSTANTS

pyrtma.constants module

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

Functions

<i>warn</i>	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

<i>HOST_ID</i>	alias of <code>c_short</code>
<i>MODULE_ID</i>	alias of <code>c_short</code>
<i>MSG_COUNT</i>	alias of <code>c_int</code>
<i>MSG_TYPE</i>	alias of <code>c_int</code>

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`

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

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

continues on next page

Table 1 – continued from previous page

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

continues on next page

Table 1 – continued from previous page

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

<code>validate_many</code>	Validate multiple byte values
<code>validate_one</code>	validate a single byte value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<i>count</i>	
<i>index</i>	Raises ValueError if the value is not present.
<i>validate_array</i>	Validate array
<i>validate_many</i>	Validate multiple values
<i>validate_one</i>	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

<i>validate_many</i>	Validate multiple strings
<i>validate_one</i>	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

<code>validate_many</code>	Validate multiple float values
<code>validate_one</code>	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

<code>validate_many</code>	Validate multiple float values
<code>validate_one</code>	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

<i>count</i>	
<i>index</i>	Raises ValueError if the value is not present.
<i>validate_array</i>	Validate array
<i>validate_many</i>	Validate multiple values
<i>validate_one</i>	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

<i>validate_many</i>	Validate multiple integer values
<i>validate_one</i>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<i>validate_many</i>	Validate multiple integer values
<i>validate_one</i>	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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>count</code>	
<code>index</code>	Raises <code>ValueError</code> if the value is not present.
<code>validate_array</code>	Validate array
<code>validate_many</code>	Validate multiple values
<code>validate_one</code>	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

<i>copy</i>	Generate a copy of a message structure
<i>from_dict</i>	Generate message instance from dictionary
<i>from_json</i>	Generate message instance from JSON string
<i>from_random</i>	Generate message instance with random values
<i>get_field_raw</i>	return copy of raw bytes for ctypes field
<i>hexdump</i>	hexdump of message
<i>pretty_print</i>	Generate formatted message structure string for pretty printing
<i>to_dict</i>	Convert message to dictionary
<i>to_json</i>	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_CONNECTBases: *MessageData***Methods**

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

<i>copy</i>	Generate a copy of a message structure
<i>from_dict</i>	Generate message instance from dictionary
<i>from_json</i>	Generate message instance from JSON string
<i>from_random</i>	Generate message instance with random values
<i>get_field_raw</i>	return copy of raw bytes for ctypes field
<i>hexdump</i>	hexdump of message
<i>pretty_print</i>	Generate formatted message structure string for pretty printing
<i>to_dict</i>	Convert message to dictionary
<i>to_json</i>	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_LOGBases: *MessageData***Methods**

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

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

Attributes

<code>size</code>
<code>type_def</code>
<code>type_hash</code>
<code>type_id</code>
<code>type_name</code>
<code>type_size</code>
<code>type_source</code>

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_MESSAGEBases: *MessageData***Methods**

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

Attributes

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

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

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

Attributes

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

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_DISCONNECTBases: *MessageData***Methods**

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

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

Attributes

<code>size</code>
<code>type_def</code>
<code>type_hash</code>
<code>type_id</code>
<code>type_name</code>
<code>type_size</code>
<code>type_source</code>

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_EXITBases: *MessageData***Methods**

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

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

Attributes

<code>size</code>
<code>type_def</code>
<code>type_hash</code>
<code>type_id</code>
<code>type_name</code>
<code>type_size</code>
<code>type_source</code>

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_SAVEDBases: *MessageData***Methods**

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

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

Attributes

<i>pid</i>	Validator for 32-bit integers
<i>size</i>	
<i>type_def</i>	
<i>type_hash</i>	
<i>type_id</i>	
<i>type_name</i>	
<i>type_size</i>	
<i>type_source</i>	

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_LOGGINGBases: *MessageData***Methods**

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

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

Attributes

<code>msg_type</code>	Validator for 32-bit integers
<code>size</code>	
<code>type_def</code>	
<code>type_hash</code>	
<code>type_id</code>	
<code>type_name</code>	
<code>type_size</code>	
<code>type_source</code>	

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_LOGBases: *MessageData***Methods**

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

<i>copy</i>	Generate a copy of a message structure
<i>from_dict</i>	Generate message instance from dictionary
<i>from_json</i>	Generate message instance from JSON string
<i>from_random</i>	Generate message instance with random values
<i>get_field_raw</i>	return copy of raw bytes for ctypes field
<i>hexdump</i>	hexdump of message
<i>pretty_print</i>	Generate formatted message structure string for pretty printing
<i>to_dict</i>	Convert message to dictionary
<i>to_json</i>	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_SUBSCRIPTIONBases: *MessageData***Methods**

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

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

Attributes

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

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_SUBSCRIBEBases: *MessageData***Methods**

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

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

Attributes

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

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_UNSUBSCRIBEBases: *MessageData***Methods**

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

<i>copy</i>	Generate a copy of a message structure
<i>from_dict</i>	Generate message instance from dictionary
<i>from_json</i>	Generate message instance from JSON string
<i>from_random</i>	Generate message instance with random values
<i>get_field_raw</i>	return copy of raw bytes for ctypes field
<i>hexdump</i>	hexdump of message
<i>pretty_print</i>	Generate formatted message structure string for pretty printing
<i>to_dict</i>	Convert message to dictionary
<i>to_json</i>	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 should not be directly instantiated.

Methods

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

Attributes

<code>size</code>
<code>type_def</code>
<code>type_id</code>
<code>type_name</code>
<code>type_size</code>
<code>type_source</code>
<code>type_hash</code>

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

<code>from_address</code>	access a C instance at the specified address
<code>from_buffer</code>	create a C instance from a writeable buffer
<code>from_buffer_copy</code>	create a C instance from a readable buffer
<code>from_param</code>	Convert a Python object into a function call parameter.
<code>in_dll</code>	access a C instance in a dll
<code>mro</code>	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

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

Attributes

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

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

<i>validate_many</i>	Validate multiple strings
<i>validate_one</i>	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

<code>validate_many</code>	Validate multiple structures
<code>validate_one</code>	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

<i>count</i>	
<i>index</i>	Raises ValueError if the value is not present.
<i>pretty_print</i>	Generate formatted string for structure array
<i>validate_array</i>	Validate structure array
<i>validate_many</i>	Validate multiple structures
<i>validate_one</i>	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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

PYRTMA.EXCEPTIONS

Exceptions

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

PYRTMA.HEADER

Functions

<code>get_header_cls</code>	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

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

22.2 pyrtma.header.Double

class `Double(*args)`

Bases: `FloatValidatorBase`

Double (64-bit float) validator class

Methods

<code>validate_many</code>	Validate multiple float values
<code>validate_one</code>	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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

`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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

`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

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

Attributes

<code>size</code>

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

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

Attributes

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

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

<code>from_address</code>	access a C instance at the specified address
<code>from_buffer</code>	create a C instance from a writeable buffer
<code>from_buffer_copy</code>	create a C instance from a readable buffer
<code>from_param</code>	Convert a Python object into a function call parameter.
<code>in_dll</code>	access a C instance in a dll
<code>mro</code>	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

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

Attributes

<code>size</code>	
<code>utc_fraction</code>	Validator for unsigned 32-bit integers
<code>utc_seconds</code>	Validator for unsigned 32-bit integers
<code>version</code>	

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

PYRTMA.MANAGER

pyrtma.manager module

Contains *MessageManager* class

Functions

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

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

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

23.5 pyrtma.manager.Counter

`class Counter(iterable=None, /, **kws)`

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

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

`__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, /, **kws)

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, /, **kws)

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

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

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 typeDict[*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 MessageDataBases: *MessageBase*

MessageData base class

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

Methods

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

Attributes

<code>size</code>
<code>type_def</code>
<code>type_id</code>
<code>type_name</code>
<code>type_size</code>
<code>type_source</code>
<code>type_hash</code>

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 MessageHeaderBases: *MessageBase*

RTMA Message Header class

Methods

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

Attributes

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

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

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

Attributes

<code>header</code>
<code>message</code>

`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

<i>close</i>	Close connection
<i>send_ack</i>	Send ACKNOWLEDGE signal header
<i>send_message</i>	Send a message

Attributes

<i>connected</i>
<i>id</i>
<i>is_logger</i>
<i>pid</i>
<i>conn</i>
<i>address</i>
<i>header_cls</i>

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

<i>clear</i>	
<i>copy</i>	
<i>fromkeys</i>	Create a new dictionary with keys from iterable and values set to value.
<i>get</i>	Return the value for key if key is in the dictionary, else default.
<i>items</i>	
<i>keys</i>	
<i>pop</i>	If key is not found, d is returned if given, otherwise KeyError is raised
<i>popitem</i>	Remove and return a (key, value) pair as a 2-tuple.
<i>setdefault</i>	Insert key with a value of default if key is not in the dictionary.
<i>update</i>	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]
<i>values</i>	

Attributes

<i>default_factory</i>	Factory for default value called by <code>__missing__()</code> .
------------------------	--

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

PYRTMA.MESSAGE

pyrtma.messaage: RTMA message classes

Functions

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

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

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

24.9 pyrtma.message.Message

`class Message(header, data)`

Bases: `object`

Message class

Contains message header and data

Methods

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

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 should not be directly instantiated.

Methods

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

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

Attributes

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

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 typeDict[*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

<i>default</i>	Default method to return serializable objects
<i>encode</i>	Return a JSON string representation of a Python data structure.
<i>iterencode</i>	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

<i>InvalidMessageDefinition</i>	Raised when there is message definition is out of sync with sent data.
<i>UnknownMessageType</i>	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.

PYRTMA.MESSAGE_BASE

Functions

<code>hexdump</code>	Dump bytes as hex
<code>is_dataclass</code>	Returns True if obj is a dataclass or an instance of a dataclass.
<code>print_ctype_array</code>	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

<i>CStructType</i>	alias of PyCStructType
<i>MessageBase</i>	MessageBase base class
<i>MessageMeta</i>	MessageMeta metaclass
<i>RTMAJSONEncoder</i>	JSONEncoder object used to convert MessageData to json
<i>TypeVar</i>	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

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

<i>from_address</i>	access a C instance at the specified address
<i>from_buffer</i>	create a C instance from a writeable buffer
<i>from_buffer_copy</i>	create a C instance from a readable buffer
<i>from_param</i>	Convert a Python object into a function call parameter.
<i>in_dll</i>	access a C instance in a dll
<i>mro</i>	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

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

Attributes

<i>item_separator</i>
<i>key_separator</i>

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.

PYRTMA.MESSAGE_DATA

Classes

<i>MessageBase</i>	MessageBase base class
<i>MessageData</i>	MessageData base class
<i>MessageMeta</i>	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

<i>copy</i>	Generate a copy of a message structure
<i>from_dict</i>	Generate message instance from dictionary
<i>from_json</i>	Generate message instance from JSON string
<i>from_random</i>	Generate message instance with random values
<i>get_field_raw</i>	return copy of raw bytes for ctypes field
<i>hexdump</i>	hexdump of message
<i>pretty_print</i>	Generate formatted message structure string for pretty printing
<i>to_dict</i>	Convert message to dictionary
<i>to_json</i>	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 should not be directly instantiated.

Methods

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

Attributes

<code>size</code>
<code>type_def</code>
<code>type_id</code>
<code>type_name</code>
<code>type_size</code>
<code>type_source</code>
<code>type_hash</code>

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

<code>from_address</code>	access a C instance at the specified address
<code>from_buffer</code>	create a C instance from a writeable buffer
<code>from_buffer_copy</code>	create a C instance from a readable buffer
<code>from_param</code>	Convert a Python object into a function call parameter.
<code>in_dll</code>	access a C instance in a dll
<code>mro</code>	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.

PYRTMA.PARSER

Message definition YAML parser

Functions

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

<i>CompilerOptions</i>	
<i>ConstantExpr</i>	
<i>ConstantString</i>	
<i>CustomEncoder</i>	Constructor for JSONEncoder, with sensible defaults.
<i>Field</i>	
<i>HID</i>	
<i>Import</i>	
<i>MDF</i>	
<i>MID</i>	
<i>MT</i>	
<i>Metadata</i>	
<i>NativeType</i>	
<i>Parser</i>	Parser class
<i>SDF</i>	
<i>TypeAlias</i>	
<i>YAML</i>	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

<code>default</code>	Implement this method in a subclass such that it returns a serializable object for <code>o</code> , or calls the base implementation (to raise a <code>TypeError</code>).
<code>encode</code>	Return a JSON string representation of a Python data structure.
<code>iterencode</code>	Encode the given object and yield each string representation as available.

Attributes

<code>item_separator</code>
<code>key_separator</code>

`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	
<i>check_alignment</i>	Confirm 64 bit alignment of structures
<i>check_duplicate_name</i>	Check namespaces for conflicting names.
check_key_value_separation	
<i>check_name</i>	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
<i>handle_alias</i>	Find the base type ultimately represented by the type-def 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

<i>Xdump_all</i>	Serialize a sequence of Python objects into a YAML stream.
compact	rtype None
<i>compose</i>	Parse the first YAML document in a stream and produce the corresponding representation tree.
<i>compose_all</i>	Parse all YAML documents in a stream and produce corresponding representation trees.
dump	rtype Any
dump_all	rtype Any
<i>emit</i>	Emit YAML parsing events into a stream.
<i>get_constructor_parser</i>	the old cyaml needs special setup, and therefore the stream
get_serializer_representer_emitter	rtype Any
<i>load</i>	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
<i>official_plug_ins</i>	search for list of subdirs that are plug-ins, if <code>__file__</code> is not available, e.g.
<i>parse</i>	Parse a YAML stream and produce parsing events.
<i>register_class</i>	register a class for dumping/loading :rtype: Any
<i>scan</i>	Scan a YAML stream and produce scanning tokens.
seq	rtype Any
<i>serialize</i>	Serialize a representation tree into a YAML stream.
<i>serialize_all</i>	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

<i>AlignmentError</i>	Raised when a struct is not 64-bit aligned
<i>CircularRefError</i>	Raised when an expression contains a circular reference
<i>DuplicateNameError</i>	Raised when the a name is already in use
<i>ExpressionExpansionError</i>	Raised when an expression can not be expanded.
<i>FileFormatError</i>	Raised when the wrong file extension is referenced.
<i>HostIDError</i>	Raised when the a host id is already in use
<i>InvalidMessageSize</i>	Raised when a message is too large
<i>InvalidTypeError</i>	Raised when a field contains the wrong type of data
<i>MessageIDError</i>	Raised when the a message id is already in use
<i>ModuleIDError</i>	Raised when the a module id is already in use
<i>ParserError</i>	Base class for all parser exceptions
<i>RTMASyntaxError</i>	Raised when the parser encounters invalid RTMA syntax
<i>RecurisionError</i>	Raised when recursion limit is exceeded evaluating aliases and expressions.
<i>YAMLSyntaxError</i>	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.RTMSyntaxError

exception `RTMSyntaxError`

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

PYRTMA.VALIDATORS

Functions

<i>abstractmethod</i>	A decorator indicating abstract methods.
<i>contextmanager</i>	@contextmanager decorator.
<i>disable_message_validation</i>	Context manager function to temporarily disable message field validation Use with <i>with</i> keyword: <i>with disable_message_validation()</i> :
<i>overload</i>	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

<i>ABCMeta</i>	Metaclass for defining Abstract Base Classes (ABCs).
<i>ArrayField</i>	Array field validator base class
<i>Byte</i>	Validator for single byte values
<i>ByteArray</i>	Validator class for Bytes arrays
<i>Char</i>	Validator for scalar char values
<i>ContextVar</i>	
<i>Double</i>	Double (64-bit float) validator class
<i>FieldValidator</i>	Abstract base class for all message field validator descriptors
<i>Float</i>	32-bit Float validator class
<i>FloatArray</i>	Validator class for float arrays
<i>FloatValidatorBase</i>	Abstract base class for float type validators
<i>Generic</i>	Abstract base class for generic types.
<i>Int16</i>	Validator for 16-bit integers
<i>Int32</i>	Validator for 32-bit integers
<i>Int64</i>	Validator for 64-bit integers
<i>Int8</i>	Validator for 8-bit integers
<i>IntArray</i>	IntArray validator class
<i>IntValidatorBase</i>	Abstract base class for integer type validators
<i>MessageBase</i>	MessageBase base class
<i>String</i>	Validator for strings (char arrays)
<i>Struct</i>	Validator class for Structures
<i>StructArray</i>	Validator for structure arrays
<i>TypeVar</i>	Type variable.
<i>UInt16</i>	Validator for unsigned 16-bit integers
<i>UInt32</i>	Validator for unsigned 32-bit integers
<i>UInt64</i>	Validator for unsigned 64-bit integers
<i>UInt8</i>	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

<code>mro</code>	Return a type's method resolution order.
<code>register</code>	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

<code>count</code>	
<code>index</code>	Raises <code>ValueError</code> if the value is not present.
<code>validate_array</code>	Validate array
<code>validate_many</code>	Validate multiple values
<code>validate_one</code>	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

<i>validate_many</i>	Validate multiple byte values
<i>validate_one</i>	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

<i>count</i>	
<i>index</i>	Raises ValueError if the value is not present.
<i>validate_array</i>	Validate array
<i>validate_many</i>	Validate multiple values
<i>validate_one</i>	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

<i>validate_many</i>	Validate multiple strings
<i>validate_one</i>	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

<code>get</code>	Return a value for the context variable for the current context.
<code>reset</code>	Reset the context variable.
<code>set</code>	Call to set a new value for the context variable in the current context.

Attributes

<code>name</code>

`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

<code>validate_many</code>	Validate multiple float values
<code>validate_one</code>	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

<code>validate_many</code>
<code>validate_one</code>

28.13 pyrtma.validators.Float

class `Float(*args)`

Bases: *FloatValidatorBase*

32-bit Float validator class

Methods

<code>validate_many</code>	Validate multiple float values
<code>validate_one</code>	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

<code>count</code>	
<code>index</code>	Raises ValueError if the value is not present.
<code>validate_array</code>	Validate array
<code>validate_many</code>	Validate multiple values
<code>validate_one</code>	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

<code>validate_many</code>	Validate multiple float values
<code>validate_one</code>	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, **kwargs)`

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	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

<i>validate_many</i>	Validate multiple integer values
<i>validate_one</i>	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

<i>validate_many</i>	Validate multiple integer values
<i>validate_one</i>	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

<i>count</i>	
<i>index</i>	Raises ValueError if the value is not present.
<i>validate_array</i>	Validate array
<i>validate_many</i>	Validate multiple values
<i>validate_one</i>	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

<i>validate_many</i>	Validate multiple integer values
<i>validate_one</i>	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

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

Attributes

<code>size</code>

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

<code>validate_many</code>	Validate multiple strings
<code>validate_one</code>	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

<code>validate_many</code>	Validate multiple structures
<code>validate_one</code>	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

<i>count</i>	
<i>index</i>	Raises ValueError if the value is not present.
<i>pretty_print</i>	Generate formatted string for structure array
<i>validate_array</i>	Validate structure array
<i>validate_many</i>	Validate multiple structures
<i>validate_one</i>	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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

<code>validate_many</code>	Validate multiple integer values
<code>validate_one</code>	Validate an integer value

Attributes

<code>max</code>
<code>min</code>
<code>size</code>
<code>unsigned</code>

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

PYRTMA.WEB_MANAGER

Functions

<code>cast</code>	Cast a value to a type.
<code>get_msg_cls</code>	get msg class for a given message type ID
<code>main</code>	Main function for starting web_manager
<code>ws_client_connect</code>	Websocket client connect
<code>ws_client_disconnect</code>	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

<i>Client</i>	RTMA Client interface
<i>Message</i>	Message class
<i>MessageHeader</i>	RTMA Message Header class
<i>RTMAWebSocketHandler</i>	RTMA Web Socket Handler class
<i>RichHandler</i>	A logging handler that renders output with Rich.
<i>TCPServer</i>	Base class for various socket-based server classes.
<i>WebMessageManager</i>	WebMessageManager class
<i>WebSocketHandler</i>	
<i>WebSocketServer</i>	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

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

Attributes

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

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

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

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

Attributes

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

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	
<i>finish</i>	Close RTMA connection
<i>handle</i>	Handle RTMA proxy connection
handle_connect	
handshake	
make_handshake_response	
<i>pong_received</i>	Log that pong was received
<i>process_json_message</i>	Process incoming json message
read_bytes	
read_http_headers	
read_next_message	
<i>read_ws_message</i>	Read websocket message
<i>send_close</i>	Send CLOSE to client
<i>send_failed_message</i>	Send FAILED_MESSAGE when we cannot forward to websocket
send_message	
send_pong	
<i>send_text</i>	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 traceback 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

<i>acquire</i>	Acquire the I/O thread lock.
<i>addFilter</i>	Add the specified filter to this handler.
<i>close</i>	Tidy up any resources used by the handler.
<i>createLock</i>	Acquire a thread lock for serializing access to the underlying I/O.
<i>emit</i>	Invoked by logging.
<i>filter</i>	Determine if a record is loggable by consulting all the filters.
<i>flush</i>	Ensure all logging output has been flushed.
<i>format</i>	Format the specified record.
<i>get_level_text</i>	Get the level name from the record.
<i>get_name</i>	
<i>handle</i>	Conditionally emit the specified logging record.
<i>handleError</i>	Handle errors which occur during an emit() call.
<i>release</i>	Release the I/O thread lock.
<i>removeFilter</i>	Remove the specified filter from this handler.
<i>render</i>	Render log for display.
<i>render_message</i>	Render message text in to Text.
<i>setFormatter</i>	Set the formatter for this handler.
<i>setLevel</i>	Set the logging level of this handler.
<i>set_name</i>	

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

<i>close_request</i>	Called to clean up an individual request.
<i>fileno</i>	Return socket file number.
<i>finish_request</i>	Finish one request by instantiating RequestHandlerClass.
<i>get_request</i>	Get the request and client address from the socket.
<i>handle_error</i>	Handle an error gracefully.
<i>handle_request</i>	Handle one request, possibly blocking.
<i>handle_timeout</i>	Called if no new request arrives within self.timeout.
<i>process_request</i>	Call finish_request.
<i>serve_forever</i>	Handle one request at a time until shutdown.
<i>server_activate</i>	Called by constructor to activate the server.
<i>server_bind</i>	Called by constructor to bind the socket.
<i>server_close</i>	Called to clean-up the server.
<i>service_actions</i>	Called by the serve_forever() loop.
<i>shutdown</i>	Stops the serve_forever loop.
<i>shutdown_request</i>	Called to shutdown and close an individual request.
<i>verify_request</i>	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*) – Logging level. Defaults to logging.WARNING.
- **key** (*optional*) – Path to SSL key. Defaults to None.
- **cert** (*optional*) – Path to SSL cert. Defaults to None.

Methods

<code>allow_new_connections</code>	
<code>client_left</code>	
<code>close_request</code>	Called to clean up an individual request.
<code>deny_new_connections</code>	
<code>disconnect_clients_abruptly</code>	
<code>disconnect_clients_gracefully</code>	
<code>fileno</code>	Return socket file number.
<code>finish_request</code>	Finish one request by instantiating RequestHandler-Class.
<code>get_request</code>	Get the request and client address from the socket.
<code>handle_error</code>	Handle an error gracefully.
<code>handle_request</code>	Handle one request, possibly blocking.
<code>handle_timeout</code>	Called if no new request arrives within self.timeout.
<code>handler_to_client</code>	
<code>message_received</code>	
<code>new_client</code>	
<code>process_request</code>	Start a new thread to process the request.
<code>process_request_thread</code>	Same as in BaseServer but as a thread.

continues on next page

Table 1 – continued from previous page

<code>run_forever</code>	
<code>send_message</code>	
<code>send_message_to_all</code>	
<code><i>serve_forever</i></code>	Handle one request at a time until shutdown.
<code><i>server_activate</i></code>	Called by constructor to activate the server.
<code><i>server_bind</i></code>	Called by constructor to bind the socket.
<code><i>server_close</i></code>	Called to clean-up the server.
<code><i>service_actions</i></code>	Called by the <code>serve_forever()</code> loop.
<code>set_fn_client_left</code>	
<code>set_fn_message_received</code>	
<code>set_fn_new_client</code>	
<code><i>shutdown</i></code>	Stops the <code>serve_forever</code> loop.
<code>shutdown_abruptly</code>	
<code>shutdown_gracefully</code>	
<code><i>shutdown_request</i></code>	Called to shutdown and close an individual request.
<code><i>verify_request</i></code>	Verify the request.

Attributes

<code>address_family</code>
<code>allow_reuse_address</code>
<code>block_on_close</code>
<code>daemon_threads</code>
<code>request_queue_size</code>
<code>socket_type</code>
<code>timeout</code>

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

<code>calculate_response_key</code>	
<code>finish</code>	
<code>handle</code>	
<code>handshake</code>	
<code>make_handshake_response</code>	
<code>read_bytes</code>	
<code>read_http_headers</code>	
<code>read_next_message</code>	
<code>send_close</code>	Send CLOSE to client
<code>send_message</code>	
<code>send_pong</code>	
<code>send_text</code>	Important: Fragmented(=continuation) messages are not supported since their usage cases are limited - when we don't know the payload length.
<code>setup</code>	

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

<code>allow_new_connections</code>	
<code>client_left</code>	
<code>close_request</code>	Called to clean up an individual request.
<code>deny_new_connections</code>	
<code>disconnect_clients_abruptly</code>	
<code>disconnect_clients_gracefully</code>	
<code>fileno</code>	Return socket file number.
<code>finish_request</code>	Finish one request by instantiating RequestHandler-Class.
<code>get_request</code>	Get the request and client address from the socket.
<code>handle_error</code>	Handle an error gracefully.
<code>handle_request</code>	Handle one request, possibly blocking.
<code>handle_timeout</code>	Called if no new request arrives within self.timeout.
<code>handler_to_client</code>	
<code>message_received</code>	
<code>new_client</code>	
<code>process_request</code>	Start a new thread to process the request.
<code>process_request_thread</code>	Same as in BaseServer but as a thread.
<code>run_forever</code>	
<code>send_message</code>	
<code>send_message_to_all</code>	
<code>serve_forever</code>	Handle one request at a time until shutdown.
<code>server_activate</code>	Called by constructor to activate the server.
<code>server_bind</code>	Called by constructor to bind the socket.
<code>server_close</code>	Called to clean-up the server.
<code>service_actions</code>	Called by the serve_forever() loop.
<code>set_fn_client_left</code>	
<code>set_fn_message_received</code>	
<code>set_fn_new_client</code>	
<code>shutdown</code>	Stops the serve_forever loop.
<code>shutdown_abruptly</code>	
<code>shutdown_gracefully</code>	
<code>shutdown_request</code>	Called to shutdown and close an individual request.

continues on next page

Table 2 – continued from previous page

<i>verify_request</i>	Verify the request.
-----------------------	---------------------

Attributes

<code>address_family</code>
<code>allow_reuse_address</code>
<code>block_on_close</code>
<code>daemon_threads</code>
<code>request_queue_size</code>
<code>socket_type</code>
<code>timeout</code>

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

<i>ClientError</i>	Base exception for all Client Errors.
<i>RTMAMessageError</i>	Base exception for message errors.
<i>SocketError</i>	alias of <i>OSError</i>

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`

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 definitions 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 definition
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|ml|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

- pyrtma, ??
- pyrtma.client, 43
- pyrtma.compile, 61
- pyrtma.constants, 75
- pyrtma.core_defs, 77
- pyrtma.exceptions, 159
- pyrtma.header, 163
- pyrtma.manager, 177
- pyrtma.message, 197
- pyrtma.message_base, 209
- pyrtma.message_data, 217
- pyrtma.parser, 223
- pyrtma.validators, 247
- pyrtma.web_manager, 275

Symbols

`__add__()` (*Counter method*), 180
`__call__()` (*ABCMeta method*), 250
`__call__()` (*MessageMeta method*), 149, 172, 212, 222
`__mul__()` (*MessageMeta method*), 149, 172, 212, 222

A

ABCMeta (*class in pyrtma.validators*), 249
`abstractmethod()` (*in module pyrtma.validators*), 247
AcknowledgementTimeout, 29, 58, 159
`acquire()` (*RichHandler method*), 289
`add_subscription()` (*MessageManager method*), 189
`addFilter()` (*RichHandler method*), 289
AlignmentError, 243
ArrayField (*class in pyrtma.validators*), 250
`asdict()` (*in module pyrtma.parser*), 223
`assign_module_id()` (*MessageManager method*), 189

B

Byte (*class in pyrtma.core_defs*), 79
Byte (*class in pyrtma.validators*), 251
ByteArray (*class in pyrtma.core_defs*), 80
ByteArray (*class in pyrtma.validators*), 252

C

`cast()` (*in module pyrtma.client*), 43
`cast()` (*in module pyrtma.web_manager*), 275
CDefCompiler (*class in pyrtma.compile*), 63
Char (*class in pyrtma.core_defs*), 81
Char (*class in pyrtma.validators*), 253
`check_alignment()` (*Parser method*), 70, 236
`check_compiled_version()` (*in module pyrtma.core_defs*), 77
`check_duplicate_name()` (*Parser method*), 70, 236
`check_name()` (*Parser method*), 71, 236
CircularRefError, 243
`clear()` (*Counter method*), 180
`clear()` (*defaultdict method*), 194
`clear_msg_defs()` (*in module pyrtma.message*), 197
Client (*class in pyrtma*), 13
Client (*class in pyrtma.client*), 46

Client (*class in pyrtma.web_manager*), 277
`client_context()` (*in module pyrtma*), 3
`client_context()` (*in module pyrtma.client*), 43
ClientError, 31, 58, 159, 303
`close()` (*MessageManager method*), 189
`close()` (*Module method*), 193
`close()` (*RichHandler method*), 289
`close_request()` (*TCPServer method*), 293
`close_request()` (*WebMessageManager method*), 296
`close_request()` (*WebsocketServer method*), 301
`compile()` (*in module pyrtma.compile*), 61
CompilerOptions (*class in pyrtma.parser*), 225
`compose()` (*YAML method*), 240
`compose_all()` (*YAML method*), 241
`connect()` (*Client method*), 14, 47, 278
`connect_module()` (*MessageManager method*), 189
connected (*Client property*), 14, 47, 278
ConnectionLost, 33, 58, 160
ConstantExpr (*class in pyrtma.parser*), 226
ConstantString (*class in pyrtma.parser*), 227
`contextmanager()` (*in module pyrtma.client*), 44
`contextmanager()` (*in module pyrtma.validators*), 247
ContextVar (*class in pyrtma.validators*), 253
`copy()` (*Counter method*), 180
`copy()` (*defaultdict method*), 194
`copy()` (*in module pyrtma.parser*), 224
`copy()` (*MDF_ACKNOWLEDGE class method*), 89
`copy()` (*MDF_CONNECT class method*), 92
`copy()` (*MDF_DISCONNECT class method*), 94
`copy()` (*MDF_DUMP_MESSAGE_LOG class method*), 97
`copy()` (*MDF_EXIT class method*), 99
`copy()` (*MDF_FAIL_SUBSCRIBE class method*), 104
`copy()` (*MDF_FAILED_MESSAGE class method*), 102
`copy()` (*MDF_FORCE_DISCONNECT class method*), 107
`copy()` (*MDF_KILL class method*), 109
`copy()` (*MDF_LM_EXIT class method*), 112
`copy()` (*MDF_LM_READY class method*), 114
`copy()` (*MDF_MESSAGE_LOG_SAVED class method*), 117
`copy()` (*MDF_MODULE_READY class method*), 119

- copy() (*MDF_PAUSE_MESSAGE_LOGGING* class method), 122
- copy() (*MDF_PAUSE_SUBSCRIPTION* class method), 124
- copy() (*MDF_RESET_MESSAGE_LOG* class method), 127
- copy() (*MDF_RESUME_MESSAGE_LOGGING* class method), 129
- copy() (*MDF_RESUME_SUBSCRIPTION* class method), 132
- copy() (*MDF_SAVE_MESSAGE_LOG* class method), 134
- copy() (*MDF_SUBSCRIBE* class method), 137
- copy() (*MDF_TIMING_MESSAGE* class method), 139
- copy() (*MDF_UNSUBSCRIBE* class method), 142
- copy() (*Message* class method), 19, 51, 182, 200, 282
- copy() (*MessageBase* class method), 145, 167, 210, 218, 265
- copy() (*MessageData* class method), 21, 53, 147, 184, 201, 220
- copy() (*MessageHeader* class method), 25, 55, 170, 186, 204, 284
- copy() (*RTMA_MSG_HEADER* class method), 150
- copy() (*TimeCodeMessageHeader* class method), 173
- count() (*ArrayField* method), 250
- count() (*ByteArray* method), 80, 252
- count() (*FloatArray* method), 83, 256
- count() (*IntArray* method), 88, 263
- count() (*StructArray* method), 154, 268
- Counter (class in *pyrtma.manager*), 178
- createLock() (*RichHandler* method), 290
- CStructType (in module *pyrtma.message_base*), 210
- CustomEncoder (class in *pyrtma.parser*), 227
- ## D
- dataclass() (in module *pyrtma.manager*), 177
- dataclass() (in module *pyrtma.parser*), 224
- default() (*CustomEncoder* method), 228
- default() (*RTMAJSONEncoder* method), 207, 214
- default_factory (*defaultdict* attribute), 194
- defaultdict (class in *pyrtma.manager*), 193
- disable_message_validation() (in module *pyrtma.validators*), 248
- discard_messages() (*Client* method), 14, 48, 278
- disconnect() (*Client* method), 14, 48, 278
- disconnect_module() (*MessageManager* method), 190
- Double (class in *pyrtma.core_defs*), 81
- Double (class in *pyrtma.header*), 164
- Double (class in *pyrtma.validators*), 254
- DuplicateNameError, 243
- ## E
- elements() (*Counter* method), 180
- emit() (*RichHandler* method), 290
- emit() (*YAML* method), 241
- encode() (*CustomEncoder* method), 228
- encode() (*RTMAJSONEncoder* method), 207, 214
- ExpressionExpansionError, 243
- ## F
- Field (class in *pyrtma.parser*), 229
- field() (in module *pyrtma.parser*), 224
- FieldValidator (class in *pyrtma.validators*), 255
- FileFormatError, 73, 244
- fileno() (*TCPServer* method), 293
- fileno() (*WebMessageManager* method), 296
- fileno() (*WebsocketServer* method), 301
- filter() (*RichHandler* method), 290
- finish() (*RTMAWebSocketHandler* method), 286
- finish_request() (*TCPServer* method), 293
- finish_request() (*WebMessageManager* method), 296
- finish_request() (*WebsocketServer* method), 301
- Float (class in *pyrtma.core_defs*), 82
- Float (class in *pyrtma.validators*), 255
- FloatArray (class in *pyrtma.core_defs*), 83
- FloatArray (class in *pyrtma.validators*), 256
- FloatValidatorBase (class in *pyrtma.validators*), 257
- flush() (*RichHandler* method), 290
- format() (*RichHandler* method), 290
- forward_message() (*Client* method), 14, 48, 278
- forward_message() (*MessageManager* method), 190
- from_address() (*MessageMeta* method), 149, 172, 212, 222
- from_buffer() (*MessageMeta* method), 149, 172, 212, 222
- from_buffer_copy() (*MessageMeta* method), 149, 172, 213, 222
- from_dict() (*MDF_ACKNOWLEDGE* class method), 89
- from_dict() (*MDF_CONNECT* class method), 92
- from_dict() (*MDF_DISCONNECT* class method), 94
- from_dict() (*MDF_DUMP_MESSAGE_LOG* class method), 97
- from_dict() (*MDF_EXIT* class method), 99
- from_dict() (*MDF_FAIL_SUBSCRIBE* class method), 104
- from_dict() (*MDF_FAILED_MESSAGE* class method), 102
- from_dict() (*MDF_FORCE_DISCONNECT* class method), 107
- from_dict() (*MDF_KILL* class method), 109
- from_dict() (*MDF_LM_EXIT* class method), 112
- from_dict() (*MDF_LM_READY* class method), 114
- from_dict() (*MDF_MESSAGE_LOG_SAVED* class method), 117
- from_dict() (*MDF_MODULE_READY* class method), 119

`from_dict()` (*MDF_PAUSE_MESSAGE_LOGGING* class method), 122
`from_dict()` (*MDF_PAUSE_SUBSCRIPTION* class method), 124
`from_dict()` (*MDF_RESET_MESSAGE_LOG* class method), 127
`from_dict()` (*MDF_RESUME_MESSAGE_LOGGING* class method), 129
`from_dict()` (*MDF_RESUME_SUBSCRIPTION* class method), 132
`from_dict()` (*MDF_SAVE_MESSAGE_LOG* class method), 134
`from_dict()` (*MDF_SUBSCRIBE* class method), 137
`from_dict()` (*MDF_TIMING_MESSAGE* class method), 139
`from_dict()` (*MDF_UNSUBSCRIBE* class method), 142
`from_dict()` (*MessageBase* class method), 145, 167, 211, 218, 265
`from_dict()` (*MessageData* class method), 22, 53, 147, 184, 202, 220
`from_dict()` (*MessageHeader* class method), 26, 55, 170, 187, 204, 284
`from_dict()` (*RTMA_MSG_HEADER* class method), 151
`from_dict()` (*TimeCodeMessageHeader* class method), 173
`from_json()` (*MDF_ACKNOWLEDGE* class method), 90
`from_json()` (*MDF_CONNECT* class method), 92
`from_json()` (*MDF_DISCONNECT* class method), 95
`from_json()` (*MDF_DUMP_MESSAGE_LOG* class method), 97
`from_json()` (*MDF_EXIT* class method), 100
`from_json()` (*MDF_FAIL_SUBSCRIBE* class method), 105
`from_json()` (*MDF_FAILED_MESSAGE* class method), 102
`from_json()` (*MDF_FORCE_DISCONNECT* class method), 107
`from_json()` (*MDF_KILL* class method), 110
`from_json()` (*MDF_LM_EXIT* class method), 112
`from_json()` (*MDF_LM_READY* class method), 115
`from_json()` (*MDF_MESSAGE_LOG_SAVED* class method), 117
`from_json()` (*MDF_MODULE_READY* class method), 120
`from_json()` (*MDF_PAUSE_MESSAGE_LOGGING* class method), 122
`from_json()` (*MDF_PAUSE_SUBSCRIPTION* class method), 125
`from_json()` (*MDF_RESET_MESSAGE_LOG* class method), 127
`from_json()` (*MDF_RESUME_MESSAGE_LOGGING* class method), 130
`from_json()` (*MDF_RESUME_SUBSCRIPTION* class method), 130
`from_json()` (*MDF_RESUME_SUBSCRIPTION* class method), 132
`from_json()` (*MDF_SAVE_MESSAGE_LOG* class method), 135
`from_json()` (*MDF_SUBSCRIBE* class method), 137
`from_json()` (*MDF_TIMING_MESSAGE* class method), 140
`from_json()` (*MDF_UNSUBSCRIBE* class method), 142
`from_json()` (*Message* class method), 19, 51, 182, 200, 282
`from_json()` (*MessageBase* class method), 145, 168, 211, 218, 265
`from_json()` (*MessageData* class method), 22, 53, 147, 184, 202, 220
`from_json()` (*MessageHeader* class method), 26, 56, 170, 187, 204, 284
`from_json()` (*RTMA_MSG_HEADER* class method), 151
`from_json()` (*TimeCodeMessageHeader* class method), 173
`from_param()` (*MessageMeta* method), 149, 172, 213, 222
`from_random()` (*MDF_ACKNOWLEDGE* class method), 90
`from_random()` (*MDF_CONNECT* class method), 92
`from_random()` (*MDF_DISCONNECT* class method), 95
`from_random()` (*MDF_DUMP_MESSAGE_LOG* class method), 97
`from_random()` (*MDF_EXIT* class method), 100
`from_random()` (*MDF_FAIL_SUBSCRIBE* class method), 105
`from_random()` (*MDF_FAILED_MESSAGE* class method), 102
`from_random()` (*MDF_FORCE_DISCONNECT* class method), 107
`from_random()` (*MDF_KILL* class method), 110
`from_random()` (*MDF_LM_EXIT* class method), 112
`from_random()` (*MDF_LM_READY* class method), 115
`from_random()` (*MDF_MESSAGE_LOG_SAVED* class method), 117
`from_random()` (*MDF_MODULE_READY* class method), 120
`from_random()` (*MDF_PAUSE_MESSAGE_LOGGING* class method), 122
`from_random()` (*MDF_PAUSE_SUBSCRIPTION* class method), 125
`from_random()` (*MDF_RESET_MESSAGE_LOG* class method), 127
`from_random()` (*MDF_RESUME_MESSAGE_LOGGING* class method), 130
`from_random()` (*MDF_RESUME_SUBSCRIPTION*

- class method*), 132
 - `from_random()` (*MDF_SAVE_MESSAGE_LOG class method*), 135
 - `from_random()` (*MDF_SUBSCRIBE class method*), 137
 - `from_random()` (*MDF_TIMING_MESSAGE class method*), 140
 - `from_random()` (*MDF_UNSUBSCRIBE class method*), 142
 - `from_random()` (*MessageBase class method*), 145, 168, 211, 218, 265
 - `from_random()` (*MessageData class method*), 22, 53, 148, 185, 202, 221
 - `from_random()` (*MessageHeader class method*), 26, 56, 170, 187, 204, 284
 - `from_random()` (*RTMA_MSG_HEADER class method*), 151
 - `from_random()` (*TimeCodeMessageHeader class method*), 173
 - `fromkeys()` (*Counter class method*), 180
 - `fromkeys()` (*defaultdict method*), 194
- ## G
- Generic (class in pyrtma.validators)*, 258
 - `get()` (*ContextVar method*), 254
 - `get()` (*Counter method*), 181
 - `get()` (*defaultdict method*), 194
 - `get_constructor_parser()` (*YAML method*), 241
 - `get_context()` (*in module pyrtma.core_defs*), 77
 - `get_field_raw()` (*MDF_ACKNOWLEDGE method*), 90
 - `get_field_raw()` (*MDF_CONNECT method*), 92
 - `get_field_raw()` (*MDF_DISCONNECT method*), 95
 - `get_field_raw()` (*MDF_DUMP_MESSAGE_LOG method*), 97
 - `get_field_raw()` (*MDF_EXIT method*), 100
 - `get_field_raw()` (*MDF_FAIL_SUBSCRIBE method*), 105
 - `get_field_raw()` (*MDF_FAILED_MESSAGE method*), 102
 - `get_field_raw()` (*MDF_FORCE_DISCONNECT method*), 107
 - `get_field_raw()` (*MDF_KILL method*), 110
 - `get_field_raw()` (*MDF_LM_EXIT method*), 112
 - `get_field_raw()` (*MDF_LM_READY method*), 115
 - `get_field_raw()` (*MDF_MESSAGE_LOG_SAVED method*), 117
 - `get_field_raw()` (*MDF_MODULE_READY method*), 120
 - `get_field_raw()` (*MDF_PAUSE_MESSAGE_LOGGING method*), 122
 - `get_field_raw()` (*MDF_PAUSE_SUBSCRIPTION method*), 125
 - `get_field_raw()` (*MDF_RESET_MESSAGE_LOG method*), 127
 - `get_field_raw()` (*MDF_RESUME_MESSAGE_LOGGING method*), 130
 - `get_field_raw()` (*MDF_RESUME_SUBSCRIPTION method*), 132
 - `get_field_raw()` (*MDF_SAVE_MESSAGE_LOG method*), 135
 - `get_field_raw()` (*MDF_SUBSCRIBE method*), 137
 - `get_field_raw()` (*MDF_TIMING_MESSAGE method*), 140
 - `get_field_raw()` (*MDF_UNSUBSCRIBE method*), 142
 - `get_field_raw()` (*MessageBase method*), 145, 168, 211, 218, 265
 - `get_field_raw()` (*MessageData method*), 22, 53, 148, 185, 202, 221
 - `get_field_raw()` (*MessageHeader method*), 26, 56, 170, 187, 204, 284
 - `get_field_raw()` (*RTMA_MSG_HEADER method*), 151
 - `get_field_raw()` (*TimeCodeMessageHeader method*), 174
 - `get_header_cls()` (*in module pyrtma*), 5
 - `get_header_cls()` (*in module pyrtma.client*), 45
 - `get_header_cls()` (*in module pyrtma.header*), 163
 - `get_header_cls()` (*in module pyrtma.manager*), 177
 - `get_header_cls()` (*in module pyrtma.message*), 197
 - `get_level_text()` (*RichHandler method*), 290
 - `get_msg_cls()` (*in module pyrtma*), 7
 - `get_msg_cls()` (*in module pyrtma.client*), 45
 - `get_msg_cls()` (*in module pyrtma.manager*), 178
 - `get_msg_cls()` (*in module pyrtma.message*), 198
 - `get_msg_cls()` (*in module pyrtma.web_manager*), 275
 - `get_msg_defs()` (*in module pyrtma.message*), 198
 - `get_request()` (*TCPServer method*), 293
 - `get_request()` (*WebMessageManager method*), 297
 - `get_request()` (*WebsocketServer method*), 301
- ## H
- `handle()` (*RichHandler method*), 290
 - `handle()` (*RTMAWebSocketHandler method*), 286
 - `handle_alias()` (*Parser method*), 71, 236
 - `handle_error()` (*TCPServer method*), 293
 - `handle_error()` (*WebMessageManager method*), 297
 - `handle_error()` (*WebsocketServer method*), 301
 - `handle_request()` (*TCPServer method*), 293
 - `handle_request()` (*WebMessageManager method*), 297
 - `handle_request()` (*WebsocketServer method*), 301
 - `handle_timeout()` (*TCPServer method*), 294
 - `handle_timeout()` (*WebMessageManager method*), 297
 - `handle_timeout()` (*WebsocketServer method*), 301
 - `handleError()` (*RichHandler method*), 290
 - `header_cls` (*Client property*), 15, 48, 279

- hexdump() (in module *pyrtma.message_base*), 209
 hexdump() (*MDF_ACKNOWLEDGE* method), 90
 hexdump() (*MDF_CONNECT* method), 93
 hexdump() (*MDF_DISCONNECT* method), 95
 hexdump() (*MDF_DUMP_MESSAGE_LOG* method), 98
 hexdump() (*MDF_EXIT* method), 100
 hexdump() (*MDF_FAIL_SUBSCRIBE* method), 105
 hexdump() (*MDF_FAILED_MESSAGE* method), 103
 hexdump() (*MDF_FORCE_DISCONNECT* method), 108
 hexdump() (*MDF_KILL* method), 110
 hexdump() (*MDF_LM_EXIT* method), 113
 hexdump() (*MDF_LM_READY* method), 115
 hexdump() (*MDF_MESSAGE_LOG_SAVED* method), 118
 hexdump() (*MDF_MODULE_READY* method), 120
 hexdump() (*MDF_PAUSE_MESSAGE_LOGGING* method), 123
 hexdump() (*MDF_PAUSE_SUBSCRIPTION* method), 125
 hexdump() (*MDF_RESET_MESSAGE_LOG* method), 128
 hexdump() (*MDF_RESUME_MESSAGE_LOGGING* method), 130
 hexdump() (*MDF_RESUME_SUBSCRIPTION* method), 133
 hexdump() (*MDF_SAVE_MESSAGE_LOG* method), 135
 hexdump() (*MDF_SUBSCRIBE* method), 138
 hexdump() (*MDF_TIMING_MESSAGE* method), 140
 hexdump() (*MDF_UNSUBSCRIBE* method), 143
 hexdump() (*MessageBase* method), 145, 168, 211, 218, 266
 hexdump() (*MessageData* method), 22, 54, 148, 185, 202, 221
 hexdump() (*MessageHeader* method), 26, 56, 171, 187, 205, 284
 hexdump() (*RTMA_MSG_HEADER* method), 151
 hexdump() (*TimeCodeMessageHeader* method), 174
 HID (class in *pyrtma.parser*), 230
 HIGHLIGHTER_CLASS (*RichHandler* attribute), 289
 HOST_ID (in module *pyrtma.constants*), 75
 HOST_ID (in module *pyrtma.core_defs*), 84
 HOST_ID (in module *pyrtma.header*), 164
 HostIDError, 244
 I
 Import (class in *pyrtma.parser*), 230
 in_dll() (*MessageMeta* method), 149, 172, 213, 222
 index() (*ArrayField* method), 250
 index() (*ByteArray* method), 80, 252
 index() (*FloatArray* method), 83, 256
 index() (*IntArray* method), 88, 263
 index() (*StructArray* method), 154, 268
 InfoCompiler (class in *pyrtma.compile*), 65
 install() (in module *pyrtma.compile*), 62
 Int16 (class in *pyrtma.core_defs*), 84
 Int16 (class in *pyrtma.header*), 164
 Int16 (class in *pyrtma.validators*), 258
 Int32 (class in *pyrtma.core_defs*), 85
 Int32 (class in *pyrtma.header*), 165
 Int32 (class in *pyrtma.validators*), 259
 Int64 (class in *pyrtma.core_defs*), 86
 Int64 (class in *pyrtma.validators*), 260
 Int8 (class in *pyrtma.core_defs*), 87
 Int8 (class in *pyrtma.validators*), 261
 IntArray (class in *pyrtma.core_defs*), 88
 IntArray (class in *pyrtma.validators*), 262
 IntValidatorBase (class in *pyrtma.validators*), 263
 InvalidDestinationHost, 35, 58, 160
 InvalidDestinationModule, 37, 59, 160
 InvalidMessageDefinition, 59, 160, 208
 InvalidMessageSize, 244
 InvalidTypeError, 244
 ip_addr (*Client* property), 15, 48, 279
 is_dataclass() (in module *pyrtma.message_base*), 209
 is_dataclass() (in module *pyrtma.parser*), 224
 items() (*Counter* method), 181
 items() (*defaultdict* method), 194
 iterencode() (*CustomEncoder* method), 229
 iterencode() (*RTMAJSONEncoder* method), 207, 214
 J
 JSDefCompiler (class in *pyrtma.compile*), 65
 JSONDecodingError, 160, 215
 K
 keys() (*Counter* method), 181
 keys() (*defaultdict* method), 194
 L
 load() (*YAML* method), 241
 M
 main() (in module *pyrtma.compile*), 63
 main() (in module *pyrtma.manager*), 178
 main() (in module *pyrtma.web_manager*), 276
 MatlabDefCompiler (class in *pyrtma.compile*), 66
 MDF (class in *pyrtma.parser*), 231
 MDF_ACKNOWLEDGE (class in *pyrtma.core_defs*), 89
 MDF_CONNECT (class in *pyrtma.core_defs*), 91
 MDF_DISCONNECT (class in *pyrtma.core_defs*), 94
 MDF_DUMP_MESSAGE_LOG (class in *pyrtma.core_defs*), 96
 MDF_EXIT (class in *pyrtma.core_defs*), 99
 MDF_FAIL_SUBSCRIBE (class in *pyrtma.core_defs*), 104
 MDF_FAILED_MESSAGE (class in *pyrtma.core_defs*), 101
 MDF_FORCE_DISCONNECT (class in *pyrtma.core_defs*), 106

- MDF_KILL (class in *pyrtma.core_defs*), 109
 - MDF_LM_EXIT (class in *pyrtma.core_defs*), 111
 - MDF_LM_READY (class in *pyrtma.core_defs*), 114
 - MDF_MESSAGE_LOG_SAVED (class in *pyrtma.core_defs*), 116
 - MDF_MODULE_READY (class in *pyrtma.core_defs*), 119
 - MDF_PAUSE_MESSAGE_LOGGING (class in *pyrtma.core_defs*), 121
 - MDF_PAUSE_SUBSCRIPTION (class in *pyrtma.core_defs*), 124
 - MDF_RESET_MESSAGE_LOG (class in *pyrtma.core_defs*), 126
 - MDF_RESUME_MESSAGE_LOGGING (class in *pyrtma.core_defs*), 129
 - MDF_RESUME_SUBSCRIPTION (class in *pyrtma.core_defs*), 131
 - MDF_SAVE_MESSAGE_LOG (class in *pyrtma.core_defs*), 134
 - MDF_SUBSCRIBE (class in *pyrtma.core_defs*), 136
 - MDF_TIMING_MESSAGE (class in *pyrtma.core_defs*), 139
 - MDF_UNSUBSCRIBE (class in *pyrtma.core_defs*), 141
 - Message (class in *pyrtma*), 19
 - Message (class in *pyrtma.client*), 51
 - Message (class in *pyrtma.manager*), 182
 - Message (class in *pyrtma.message*), 199
 - Message (class in *pyrtma.web_manager*), 281
 - message_def() (in module *pyrtma*), 9
 - message_def() (in module *pyrtma.message*), 198
 - MessageBase (class in *pyrtma.core_defs*), 144
 - MessageBase (class in *pyrtma.header*), 167
 - MessageBase (class in *pyrtma.message_base*), 210
 - MessageBase (class in *pyrtma.message_data*), 217
 - MessageBase (class in *pyrtma.validators*), 264
 - MessageData (class in *pyrtma*), 21
 - MessageData (class in *pyrtma.client*), 52
 - MessageData (class in *pyrtma.core_defs*), 146
 - MessageData (class in *pyrtma.manager*), 183
 - MessageData (class in *pyrtma.message*), 201
 - MessageData (class in *pyrtma.message_data*), 219
 - MessageHeader (class in *pyrtma*), 25
 - MessageHeader (class in *pyrtma.client*), 55
 - MessageHeader (class in *pyrtma.header*), 169
 - MessageHeader (class in *pyrtma.manager*), 186
 - MessageHeader (class in *pyrtma.message*), 203
 - MessageHeader (class in *pyrtma.web_manager*), 283
 - MessageIDError, 244
 - MessageManager (class in *pyrtma.manager*), 188
 - MessageManagerNotFound, 39, 59, 160
 - MessageMeta (class in *pyrtma.core_defs*), 149
 - MessageMeta (class in *pyrtma.header*), 172
 - MessageMeta (class in *pyrtma.message_base*), 212
 - MessageMeta (class in *pyrtma.message_data*), 222
 - Metadata (class in *pyrtma.parser*), 233
 - MID (class in *pyrtma.parser*), 232
 - module
 - pyrtma, 1
 - pyrtma.client, 43
 - pyrtma.compile, 61
 - pyrtma.constants, 75
 - pyrtma.core_defs, 77
 - pyrtma.exceptions, 159
 - pyrtma.header, 163
 - pyrtma.manager, 177
 - pyrtma.message, 197
 - pyrtma.message_base, 209
 - pyrtma.message_data, 217
 - pyrtma.parser, 223
 - pyrtma.validators, 247
 - pyrtma.web_manager, 275
 - Module (class in *pyrtma.manager*), 192
 - module_id (Client property), 15, 48, 279
 - MODULE_ID (in module *pyrtma.constants*), 76
 - MODULE_ID (in module *pyrtma.core_defs*), 144
 - MODULE_ID (in module *pyrtma.header*), 166
 - ModuleIDError, 244
 - most_common() (Counter method), 181
 - mro() (ABCMeta method), 250
 - mro() (MessageMeta method), 149, 172, 213, 222
 - msg_count (Client property), 15, 48, 279
 - MSG_COUNT (in module *pyrtma.constants*), 76
 - MSG_COUNT (in module *pyrtma.core_defs*), 144
 - MSG_COUNT (in module *pyrtma.header*), 167
 - msg_def() (in module *pyrtma*), 11
 - msg_def() (in module *pyrtma.message*), 198
 - MSG_TYPE (in module *pyrtma.constants*), 76
 - MSG_TYPE (in module *pyrtma.core_defs*), 144
 - MSG_TYPE (in module *pyrtma.header*), 167
 - MT (class in *pyrtma.parser*), 232
- ## N
- NativeType (class in *pyrtma.parser*), 234
 - NotConnectedError, 41, 59, 160
- ## O
- official_plug_ins() (YAML method), 241
 - overload() (in module *pyrtma.validators*), 248
- ## P
- parse() (YAML method), 241
 - Parser (class in *pyrtma.compile*), 69
 - Parser (class in *pyrtma.parser*), 234
 - ParserError, 73, 244
 - pause_all_subscriptions() (Client method), 15, 48, 279
 - pause_subscription() (Client method), 15, 48, 279
 - pause_subscription() (MessageManager method), 190

- paused_subscribed_types (*Client property*), 15, 48, 279
 paused_subscription_context() (*Client method*), 15, 48, 279
 pong_received() (*RTMAWebSocketHandler method*), 287
 pop() (*Counter method*), 181
 pop() (*defaultdict method*), 194
 popitem() (*Counter method*), 181
 popitem() (*defaultdict method*), 194
 port (*Client property*), 15, 49, 279
 pretty_print() (*MDF_ACKNOWLEDGE method*), 90
 pretty_print() (*MDF_CONNECT method*), 93
 pretty_print() (*MDF_DISCONNECT method*), 95
 pretty_print() (*MDF_DUMP_MESSAGE_LOG method*), 98
 pretty_print() (*MDF_EXIT method*), 100
 pretty_print() (*MDF_FAIL_SUBSCRIBE method*), 105
 pretty_print() (*MDF_FAILED_MESSAGE method*), 103
 pretty_print() (*MDF_FORCE_DISCONNECT method*), 108
 pretty_print() (*MDF_KILL method*), 110
 pretty_print() (*MDF_LM_EXIT method*), 113
 pretty_print() (*MDF_LM_READY method*), 115
 pretty_print() (*MDF_MESSAGE_LOG_SAVED method*), 118
 pretty_print() (*MDF_MODULE_READY method*), 120
 pretty_print() (*MDF_PAUSE_MESSAGE_LOGGING method*), 123
 pretty_print() (*MDF_PAUSE_SUBSCRIPTION method*), 125
 pretty_print() (*MDF_RESET_MESSAGE_LOG method*), 128
 pretty_print() (*MDF_RESUME_MESSAGE_LOGGING method*), 130
 pretty_print() (*MDF_RESUME_SUBSCRIPTION method*), 133
 pretty_print() (*MDF_SAVE_MESSAGE_LOG method*), 135
 pretty_print() (*MDF_SUBSCRIBE method*), 138
 pretty_print() (*MDF_TIMING_MESSAGE method*), 140
 pretty_print() (*MDF_UNSUBSCRIBE method*), 143
 pretty_print() (*Message method*), 20, 51, 183, 200, 282
 pretty_print() (*MessageBase method*), 146, 168, 211, 219, 266
 pretty_print() (*MessageData method*), 22, 54, 148, 185, 202, 221
 pretty_print() (*MessageHeader method*), 26, 56, 171, 187, 205, 284
 pretty_print() (*RTMA_MSG_HEADER method*), 151
 pretty_print() (*StructArray method*), 154, 268
 pretty_print() (*TimeCodeMessageHeader method*), 174
 print_ctype_array() (*in module pyrtma.message_base*), 209
 process_json_message() (*RTMAWebSocketHandler method*), 287
 process_message() (*MessageManager method*), 190
 process_request() (*TCPServer method*), 294
 process_request() (*WebMessageManager method*), 297
 process_request() (*WebsocketServer method*), 301
 process_request_thread() (*WebMessageManager method*), 297
 process_request_thread() (*WebsocketServer method*), 301
 PyDefCompiler (*class in pyrtma.compile*), 71
 pyrtma
 module, 1
 pyrtma.client
 module, 43
 pyrtma.compile
 module, 61
 pyrtma.constants
 module, 75
 pyrtma.core_defs
 module, 77
 pyrtma.exceptions
 module, 159
 pyrtma.header
 module, 163
 pyrtma.manager
 module, 177
 pyrtma.message
 module, 197
 pyrtma.message_base
 module, 209
 pyrtma.message_data
 module, 217
 pyrtma.parser
 module, 223
 pyrtma.validators
 module, 247
 pyrtma.web_manager
 module, 275
R
 read_message() (*Client method*), 15, 49, 279
 read_message() (*MessageManager method*), 190
 read_ws_message() (*RTMAWebSocketHandler method*), 287
 RecursionError, 245
 register() (*ABCMeta method*), 250

- register_class() (YAML method), 242
 register_module_ready() (MessageManager method), 191
 release() (RichHandler method), 290
 remove_module() (MessageManager method), 191
 remove_subscription() (MessageManager method), 191
 removeFilter() (RichHandler method), 291
 render() (RichHandler method), 291
 render_message() (RichHandler method), 291
 requires_connection() (in module *pyrtma.client*), 45
 reset() (ContextVar method), 254
 resume_all_subscriptions() (Client method), 15, 49, 279
 resume_subscription() (Client method), 16, 49, 280
 resume_subscription() (MessageManager method), 191
 RichHandler (class in *pyrtma.web_manager*), 287
 RTMA_MSG_HEADER (class in *pyrtma.core_defs*), 150
 RTMAJSONEncoder (class in *pyrtma.message*), 206
 RTMAJSONEncoder (class in *pyrtma.message_base*), 213
 RTMAMessageError, 161, 303
 RTMASyntaxError, 245
 RTMAWebSocketHandler (class in *pyrtma.web_manager*), 285
 run() (MessageManager method), 191
- ## S
- scan() (YAML method), 242
 SDF (class in *pyrtma.parser*), 237
 send_ack() (MessageManager method), 191
 send_ack() (Module method), 193
 send_close() (RTMAWebSocketHandler method), 287
 send_close() (WebSocketHandler method), 299
 send_failed_message() (MessageManager method), 191
 send_failed_message() (RTMAWebSocketHandler method), 287
 send_message() (Client method), 16, 49, 280
 send_message() (Module method), 193
 send_module_ready() (Client method), 16, 49, 280
 send_signal() (Client method), 16, 50, 280
 send_text() (RTMAWebSocketHandler method), 287
 send_text() (WebSocketHandler method), 299
 send_timing_message() (MessageManager method), 191
 send_to_loggers() (MessageManager method), 192
 serialize() (YAML method), 242
 serialize_all() (YAML method), 242
 serve_forever() (TCPServer method), 294
 serve_forever() (WebMessageManager method), 297
 serve_forever() (WebsocketServer method), 302
 server (Client property), 16, 50, 280
 server_activate() (TCPServer method), 294
 server_activate() (WebMessageManager method), 297
 server_activate() (WebsocketServer method), 302
 server_bind() (TCPServer method), 294
 server_bind() (WebMessageManager method), 297
 server_bind() (WebsocketServer method), 302
 server_close() (TCPServer method), 294
 server_close() (WebMessageManager method), 297
 server_close() (WebsocketServer method), 302
 service_actions() (TCPServer method), 294
 service_actions() (WebMessageManager method), 297
 service_actions() (WebsocketServer method), 302
 set() (ContextVar method), 254
 set_msg_defs() (in module *pyrtma.message*), 199
 setdefault() (Counter method), 181
 setdefault() (defaultdict method), 195
 setFormatter() (RichHandler method), 291
 setLevel() (RichHandler method), 291
 sha256() (in module *pyrtma.parser*), 224
 shutdown() (TCPServer method), 294
 shutdown() (WebMessageManager method), 297
 shutdown() (WebsocketServer method), 302
 shutdown_request() (TCPServer method), 294
 shutdown_request() (WebMessageManager method), 297
 shutdown_request() (WebsocketServer method), 302
 sock (Client property), 16, 50, 280
 SocketError (in module *pyrtma.web_manager*), 303
 SocketOptionError, 59, 161
 String (class in *pyrtma.core_defs*), 152
 String (class in *pyrtma.validators*), 267
 Struct (class in *pyrtma.core_defs*), 153
 Struct (class in *pyrtma.validators*), 267
 StructArray (class in *pyrtma.core_defs*), 153
 StructArray (class in *pyrtma.validators*), 268
 subscribe() (Client method), 16, 50, 280
 subscribed_types (Client property), 17, 50, 281
 subscription_context() (Client method), 17, 50, 281
 subtract() (Counter method), 181
- ## T
- TCPServer (class in *pyrtma.web_manager*), 291
 TimeCodeMessageHeader (class in *pyrtma.header*), 172
 to_dict() (MDF_ACKNOWLEDGE method), 90
 to_dict() (MDF_CONNECT method), 93
 to_dict() (MDF_DISCONNECT method), 95
 to_dict() (MDF_DUMP_MESSAGE_LOG method), 98
 to_dict() (MDF_EXIT method), 100
 to_dict() (MDF_FAIL_SUBSCRIBE method), 105
 to_dict() (MDF_FAILED_MESSAGE method), 103
 to_dict() (MDF_FORCE_DISCONNECT method), 108

- to_dict() (*MDF_KILL method*), 110
 - to_dict() (*MDF_LM_EXIT method*), 113
 - to_dict() (*MDF_LM_READY method*), 115
 - to_dict() (*MDF_MESSAGE_LOG_SAVED method*), 118
 - to_dict() (*MDF_MODULE_READY method*), 120
 - to_dict() (*MDF_PAUSE_MESSAGE_LOGGING method*), 123
 - to_dict() (*MDF_PAUSE_SUBSCRIPTION method*), 125
 - to_dict() (*MDF_RESET_MESSAGE_LOG method*), 128
 - to_dict() (*MDF_RESUME_MESSAGE_LOGGING method*), 130
 - to_dict() (*MDF_RESUME_SUBSCRIPTION method*), 133
 - to_dict() (*MDF_SAVE_MESSAGE_LOG method*), 135
 - to_dict() (*MDF_SUBSCRIBE method*), 138
 - to_dict() (*MDF_TIMING_MESSAGE method*), 140
 - to_dict() (*MDF_UNSUBSCRIBE method*), 143
 - to_dict() (*Message method*), 20, 52, 183, 200, 282
 - to_dict() (*MessageBase method*), 146, 168, 212, 219, 266
 - to_dict() (*MessageData method*), 23, 54, 148, 185, 203, 221
 - to_dict() (*MessageHeader method*), 27, 56, 171, 188, 205, 285
 - to_dict() (*RTMA_MSG_HEADER method*), 152
 - to_dict() (*TimeCodeMessageHeader method*), 174
 - to_json() (*MDF_ACKNOWLEDGE method*), 91
 - to_json() (*MDF_CONNECT method*), 93
 - to_json() (*MDF_DISCONNECT method*), 96
 - to_json() (*MDF_DUMP_MESSAGE_LOG method*), 98
 - to_json() (*MDF_EXIT method*), 101
 - to_json() (*MDF_FAIL_SUBSCRIBE method*), 106
 - to_json() (*MDF_FAILED_MESSAGE method*), 103
 - to_json() (*MDF_FORCE_DISCONNECT method*), 108
 - to_json() (*MDF_KILL method*), 111
 - to_json() (*MDF_LM_EXIT method*), 113
 - to_json() (*MDF_LM_READY method*), 116
 - to_json() (*MDF_MESSAGE_LOG_SAVED method*), 118
 - to_json() (*MDF_MODULE_READY method*), 121
 - to_json() (*MDF_PAUSE_MESSAGE_LOGGING method*), 123
 - to_json() (*MDF_PAUSE_SUBSCRIPTION method*), 126
 - to_json() (*MDF_RESET_MESSAGE_LOG method*), 128
 - to_json() (*MDF_RESUME_MESSAGE_LOGGING method*), 131
 - to_json() (*MDF_RESUME_SUBSCRIPTION method*), 133
 - to_json() (*MDF_SAVE_MESSAGE_LOG method*), 136
 - to_json() (*MDF_SUBSCRIBE method*), 138
 - to_json() (*MDF_TIMING_MESSAGE method*), 141
 - to_json() (*MDF_UNSUBSCRIBE method*), 143
 - to_json() (*Message method*), 20, 52, 183, 200, 282
 - to_json() (*MessageBase method*), 146, 169, 212, 219, 266
 - to_json() (*MessageData method*), 23, 54, 148, 185, 203, 221
 - to_json() (*MessageHeader method*), 27, 57, 171, 188, 205, 285
 - to_json() (*RTMA_MSG_HEADER method*), 152
 - to_json() (*TimeCodeMessageHeader method*), 174
 - TypeAlias (*class in pyrtma.parser*), 237
 - TypeVar (*class in pyrtma.client*), 57
 - TypeVar (*class in pyrtma.message*), 207
 - TypeVar (*class in pyrtma.message_base*), 214
 - TypeVar (*class in pyrtma.validators*), 269
- ## U
- UInt16 (*class in pyrtma.core_defs*), 155
 - UInt16 (*class in pyrtma.validators*), 270
 - UInt32 (*class in pyrtma.core_defs*), 156
 - UInt32 (*class in pyrtma.header*), 175
 - UInt32 (*class in pyrtma.validators*), 271
 - UInt64 (*class in pyrtma.core_defs*), 157
 - UInt64 (*class in pyrtma.validators*), 272
 - UInt8 (*class in pyrtma.core_defs*), 158
 - UInt8 (*class in pyrtma.validators*), 273
 - UnknownMessageType, 59, 161, 208
 - unsubscribe() (*Client method*), 17, 50, 281
 - unsubscribe_from_all() (*Client method*), 17, 50, 281
 - update() (*Counter method*), 181
 - update() (*defaultdict method*), 195
 - update_msg_defs() (*in module pyrtma.message*), 199
- ## V
- validate_array() (*ArrayField method*), 250
 - validate_array() (*ByteArray method*), 80, 252
 - validate_array() (*FloatArray method*), 83, 257
 - validate_array() (*IntArray method*), 88, 263
 - validate_array() (*StructArray method*), 154, 268
 - validate_many() (*ArrayField method*), 251
 - validate_many() (*Byte method*), 79, 251
 - validate_many() (*ByteArray method*), 80, 252
 - validate_many() (*Char method*), 81, 253
 - validate_many() (*Double method*), 82, 164, 255
 - validate_many() (*Float method*), 82, 256
 - validate_many() (*FloatArray method*), 83, 257
 - validate_many() (*FloatValidatorBase method*), 257
 - validate_many() (*Int16 method*), 84, 165, 259
 - validate_many() (*Int32 method*), 85, 166, 260
 - validate_many() (*Int64 method*), 86, 261

`validate_many()` (*Int8 method*), 87, 262
`validate_many()` (*IntArray method*), 88, 263
`validate_many()` (*IntValidatorBase method*), 264
`validate_many()` (*String method*), 152, 267
`validate_many()` (*Struct method*), 153, 267
`validate_many()` (*StructArray method*), 154, 269
`validate_many()` (*UInt16 method*), 155, 270
`validate_many()` (*UInt32 method*), 156, 175, 271
`validate_many()` (*UInt64 method*), 157, 272
`validate_many()` (*UInt8 method*), 158, 273
`validate_one()` (*ArrayField method*), 251
`validate_one()` (*Byte method*), 79, 251
`validate_one()` (*ByteArray method*), 80, 252
`validate_one()` (*Char method*), 81, 253
`validate_one()` (*Double method*), 82, 164, 255
`validate_one()` (*Float method*), 82, 256
`validate_one()` (*FloatArray method*), 83, 257
`validate_one()` (*FloatValidatorBase method*), 257
`validate_one()` (*Int16 method*), 84, 165, 259
`validate_one()` (*Int32 method*), 85, 166, 260
`validate_one()` (*Int64 method*), 86, 261
`validate_one()` (*Int8 method*), 87, 262
`validate_one()` (*IntArray method*), 88, 263
`validate_one()` (*IntValidatorBase method*), 264
`validate_one()` (*String method*), 152, 267
`validate_one()` (*Struct method*), 153, 267
`validate_one()` (*StructArray method*), 154, 269
`validate_one()` (*UInt16 method*), 155, 270
`validate_one()` (*UInt32 method*), 156, 175, 271
`validate_one()` (*UInt64 method*), 157, 272
`validate_one()` (*UInt8 method*), 158, 273
`values()` (*Counter method*), 181
`values()` (*defaultdict method*), 195
`verify_request()` (*TCPServer method*), 294
`verify_request()` (*WebMessageManager method*),
298
`verify_request()` (*WebsocketServer method*), 302
`VersionMismatchWarning`, 161

W

`warn()` (*in module pyrtma.client*), 45
`warn()` (*in module pyrtma.constants*), 75
`WebMessageManager` (*class in pyrtma.web_manager*),
295
`WebSocketHandler` (*class in pyrtma.web_manager*),
298
`WebsocketServer` (*class in pyrtma.web_manager*), 299
`wraps()` (*in module pyrtma.client*), 46
`ws_client_connect()` (*in module
pyrtma.web_manager*), 276
`ws_client_disconnect()` (*in module
pyrtma.web_manager*), 276

X

`Xdump_all()` (*YAML method*), 240

Y

`YAML` (*class in pyrtma.parser*), 238
`YAMLCompiler` (*class in pyrtma.compile*), 73
`YAMLSyntaxError`, 245