# Module `ftp.parser.message`

## Classes

**`class Message (header_size: int, type: MethodType)`**

Encapsulates a packet object that is sent to the socket

message is represented as: [header | data_1 | data_2 | … | payload?]

### Parameters

**`header_size`** : `int`
Size of the header packet

**`type`** : `MethodType`
Type of message that will be sent

### Returns

`Returns nothing`

### Methods

**`def add_payload(self, value: str) -> None`**

Attach payload to this message

#### Parameters

**`value`** : `str`
Value to be parsed into packet. Note that value must be represented in binary format

**`def has_payload(self) -> bool`**

Verifies if this message has a payload attached

#### Returns

`Returns true if contains a paylaod; otherwise false`

**`def parse(self, value: str) -> Tuple[packet, List[packet]]`**

Parses input values into its respective data field

#### Parameters

**`value`** : `str`
Inserts the values its respective data. Note that value must be represented in binary format

#### Returns

`Returns a tuple containing the header packet and its data packets`

**class Util**

## Static methods

**def bit2byte(msg: Message) -> List[bytes]**

Converts a Message object to a list of byte equivalent values

### Parameters

**msg** : Message
Message object to be converted

### Returns

List of bytes containing all packets encapsulated by the Message object

**def deserialize(binary: bytes, type: MessageType) -> Message**

Deserialize the byte value into a Message object

### Parameters

**binary** : bytes
Input byte object to be deserialized

**type** : MessageType
Type of message that is to be deserialized

### Returns

Returns a message object

**def serialize(msg: Message) -> bytes**

Serializes a Message object into bytes following its binary representation

### Parameters

**msg** : Message
Message object

### Returns

Returns the extended byte representation of that message object

**def str2bit(val: str, size: int, with_count: bool = True, size_count: int = None) -> str**

Converts a string value its equivalent binary representation in utf-8 encoding

Note that python does not use standard representation of variable size Hence, manual conversion should be done

### Parameters

**val** : str
String to be converted

**size** : int

Size of the expected binary value

`with_count` : `bool`
Attach the binary represented size of paramters val

`size_count` : `int`
Size of the binary represented value for the portion representing the size of val

## Returns

Retuns a binary representation of the value passed

# Index

## Super-module

ftp.parser

## Classes

**Message**
add_payload
has_payload
parse

**Util**
bit2byte
deserialize
serialize
str2bit