

## My Project

Generated by Doxygen 1.7.6.1

Sat Jun 2 2012 23:28:26



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	AtCommandRequest Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	6
3.1.2	Member Function Documentation . . . . .	6
3.1.2.1	clearCommandValue . . . . .	6
3.1.2.2	getFrameData . . . . .	6
3.1.2.3	getFrameDataLength . . . . .	6
3.2	AtCommandResponse Class Reference . . . . .	6
3.2.1	Detailed Description . . . . .	7
3.2.2	Member Function Documentation . . . . .	7
3.2.2.1	getCommand . . . . .	7
3.2.2.2	getStatus . . . . .	7
3.2.2.3	getValue . . . . .	7
3.2.2.4	getValueLength . . . . .	8
3.2.2.5	isOk . . . . .	8
3.3	FrameldResponse Class Reference . . . . .	8
3.3.1	Detailed Description . . . . .	8
3.4	ModemStatusResponse Class Reference . . . . .	9
3.4.1	Detailed Description . . . . .	9
3.5	PayloadRequest Class Reference . . . . .	9

3.5.1	Detailed Description	10
3.5.2	Member Function Documentation	10
3.5.2.1	getPayload	10
3.5.2.2	getPayloadLength	10
3.5.2.3	setPayload	10
3.5.2.4	setPayloadLength	10
3.6	RemoteAtCommandRequest Class Reference	10
3.6.1	Detailed Description	11
3.6.2	Constructor & Destructor Documentation	11
3.6.2.1	RemoteAtCommandRequest	11
3.6.2.2	RemoteAtCommandRequest	12
3.6.2.3	RemoteAtCommandRequest	12
3.6.2.4	RemoteAtCommandRequest	12
3.6.3	Member Function Documentation	12
3.6.3.1	getFrameData	12
3.6.3.2	getFrameDataLength	12
3.7	RemoteAtCommandResponse Class Reference	12
3.7.1	Detailed Description	13
3.7.2	Member Function Documentation	13
3.7.2.1	getCommand	13
3.7.2.2	getRemoteAddress16	13
3.7.2.3	getRemoteAddress64	13
3.7.2.4	getStatus	14
3.7.2.5	getValue	14
3.7.2.6	getValueLength	14
3.7.2.7	isOk	14
3.8	Rx16IoSampleResponse Class Reference	14
3.9	Rx16Response Class Reference	15
3.9.1	Detailed Description	15
3.10	Rx64IoSampleResponse Class Reference	16
3.11	Rx64Response Class Reference	16
3.11.1	Detailed Description	17
3.12	RxDataResponse Class Reference	17
3.12.1	Detailed Description	17

3.12.2	Member Function Documentation	18
3.12.2.1	getData	18
3.12.2.2	getData	18
3.12.2.3	getDataLength	18
3.12.2.4	getDataOffset	18
3.13	RxIoSampleBaseResponse Class Reference	18
3.13.1	Detailed Description	19
3.13.2	Member Function Documentation	19
3.13.2.1	getAnalog	19
3.13.2.2	getSampleSize	19
3.13.2.3	isAnalogEnabled	19
3.13.2.4	isDigitalEnabled	19
3.13.2.5	isDigitalOn	19
3.14	RxResponse Class Reference	20
3.14.1	Detailed Description	20
3.14.2	Member Function Documentation	20
3.14.2.1	getDataLength	20
3.14.2.2	getDataOffset	20
3.15	Tx16Request Class Reference	21
3.15.1	Detailed Description	21
3.15.2	Constructor & Destructor Documentation	21
3.15.2.1	Tx16Request	21
3.15.2.2	Tx16Request	22
3.15.3	Member Function Documentation	22
3.15.3.1	getFrameData	22
3.15.3.2	getFrameDataLength	22
3.16	Tx64Request Class Reference	22
3.16.1	Detailed Description	23
3.16.2	Constructor & Destructor Documentation	23
3.16.2.1	Tx64Request	23
3.16.2.2	Tx64Request	23
3.16.3	Member Function Documentation	23
3.16.3.1	getFrameData	23
3.16.3.2	getFrameDataLength	23

3.17 TxStatusResponse Class Reference . . . . .	24
3.17.1 Detailed Description . . . . .	24
3.18 XBee Class Reference . . . . .	24
3.18.1 Detailed Description . . . . .	25
3.18.2 Member Function Documentation . . . . .	25
3.18.2.1 begin . . . . .	25
3.18.2.2 getNextFrameId . . . . .	25
3.18.2.3 getResponse . . . . .	25
3.18.2.4 readPacket . . . . .	26
3.18.2.5 readPacket . . . . .	26
3.18.2.6 readPacketUntilAvailable . . . . .	26
3.18.2.7 send . . . . .	26
3.18.2.8 setSerial . . . . .	26
3.19 XBeeAddress Class Reference . . . . .	26
3.20 XBeeAddress64 Class Reference . . . . .	27
3.20.1 Detailed Description . . . . .	27
3.21 XBeeRequest Class Reference . . . . .	27
3.21.1 Detailed Description . . . . .	28
3.21.2 Constructor & Destructor Documentation . . . . .	28
3.21.2.1 XBeeRequest . . . . .	28
3.21.3 Member Function Documentation . . . . .	28
3.21.3.1 getApId . . . . .	28
3.21.3.2 getFrameData . . . . .	29
3.21.3.3 getFrameDataLength . . . . .	29
3.21.3.4 getFrameId . . . . .	29
3.21.3.5 setFrameId . . . . .	29
3.22 XBeeResponse Class Reference . . . . .	29
3.22.1 Detailed Description . . . . .	30
3.22.2 Constructor & Destructor Documentation . . . . .	31
3.22.2.1 XBeeResponse . . . . .	31
3.22.3 Member Function Documentation . . . . .	31
3.22.3.1 getApId . . . . .	31
3.22.3.2 getAtCommandResponse . . . . .	31
3.22.3.3 getChecksum . . . . .	31

3.22.3.4	<a href="#">getErrorCode</a>	31
3.22.3.5	<a href="#">getFrameData</a>	31
3.22.3.6	<a href="#">getFrameDataLength</a>	32
3.22.3.7	<a href="#">getLsbLength</a>	32
3.22.3.8	<a href="#">getModemStatusResponse</a>	32
3.22.3.9	<a href="#">getMsbLength</a>	32
3.22.3.10	<a href="#">getPacketLength</a>	32
3.22.3.11	<a href="#">getRemoteAtCommandResponse</a>	32
3.22.3.12	<a href="#">getRx16IoSampleResponse</a>	32
3.22.3.13	<a href="#">getRx16Response</a>	32
3.22.3.14	<a href="#">getRx64IoSampleResponse</a>	33
3.22.3.15	<a href="#">getRx64Response</a>	33
3.22.3.16	<a href="#">getTxStatusResponse</a>	33
3.22.3.17	<a href="#">getZBRxIoSampleResponse</a>	33
3.22.3.18	<a href="#">getZBRxResponse</a>	33
3.22.3.19	<a href="#">getZBTxStatusResponse</a>	33
3.22.3.20	<a href="#">init</a>	33
3.22.3.21	<a href="#">isAvailable</a>	33
3.22.3.22	<a href="#">isError</a>	33
3.22.3.23	<a href="#">reset</a>	34
3.23	<a href="#">ZBRxIoSampleResponse Class Reference</a>	34
3.23.1	<a href="#">Detailed Description</a>	34
3.23.2	<a href="#">Member Function Documentation</a>	35
3.23.2.1	<a href="#">getAnalog</a>	35
3.23.2.2	<a href="#">isAnalogEnabled</a>	35
3.23.2.3	<a href="#">isDigitalEnabled</a>	35
3.23.2.4	<a href="#">isDigitalOn</a>	35
3.24	<a href="#">ZBRxResponse Class Reference</a>	35
3.24.1	<a href="#">Detailed Description</a>	36
3.24.2	<a href="#">Member Function Documentation</a>	36
3.24.2.1	<a href="#">getDataLength</a>	36
3.24.2.2	<a href="#">getDataOffset</a>	36
3.25	<a href="#">ZBTxRequest Class Reference</a>	36
3.25.1	<a href="#">Detailed Description</a>	37

3.25.2	Constructor & Destructor Documentation . . . . .	37
3.25.2.1	ZBTxRequest . . . . .	37
3.25.2.2	ZBTxRequest . . . . .	37
3.25.3	Member Function Documentation . . . . .	38
3.25.3.1	getFrameData . . . . .	38
3.25.3.2	getFrameDataLength . . . . .	38
3.26	ZBTxStatusResponse Class Reference . . . . .	38
3.26.1	Detailed Description . . . . .	39



# Chapter 1

## Class Index

### 1.1 Class Hierarchy

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

XBee . . . . .	24
XBeeAddress . . . . .	26
XBeeAddress64 . . . . .	27
XBeeRequest . . . . .	27
AtCommandRequest . . . . .	5
RemoteAtCommandRequest . . . . .	10
PayloadRequest . . . . .	9
Tx16Request . . . . .	21
Tx64Request . . . . .	22
ZBTxRequest . . . . .	36
XBeeResponse . . . . .	29
FrameldResponse . . . . .	8
AtCommandResponse . . . . .	6
RemoteAtCommandResponse . . . . .	12
TxStatusResponse . . . . .	24
ZBTxStatusResponse . . . . .	38
ModemStatusResponse . . . . .	9
RxDataResponse . . . . .	17
RxResponse . . . . .	20
Rx16Response . . . . .	15
Rx64Response . . . . .	16
RxIoSampleBaseResponse . . . . .	18
Rx16IoSampleResponse . . . . .	14
Rx64IoSampleResponse . . . . .	16
ZBRxResponse . . . . .	35
ZBRxIoSampleResponse . . . . .	34



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AtCommandRequest</a>	5
<a href="#">AtCommandResponse</a>	6
<a href="#">FrameIdResponse</a>	8
<a href="#">ModemStatusResponse</a>	9
<a href="#">PayloadRequest</a>	9
<a href="#">RemoteAtCommandRequest</a>	10
<a href="#">RemoteAtCommandResponse</a>	12
<a href="#">Rx16IoSampleResponse</a>	14
<a href="#">Rx16Response</a>	15
<a href="#">Rx64IoSampleResponse</a>	16
<a href="#">Rx64Response</a>	16
<a href="#">RxDataResponse</a>	17
<a href="#">RxIoSampleBaseResponse</a>	18
<a href="#">RxResponse</a>	20
<a href="#">Tx16Request</a>	21
<a href="#">Tx64Request</a>	22
<a href="#">TxStatusResponse</a>	24
<a href="#">XBee</a>	24
<a href="#">XBeeAddress</a>	26
<a href="#">XBeeAddress64</a>	27
<a href="#">XBeeRequest</a>	27
<a href="#">XBeeResponse</a>	29
<a href="#">ZBRxIoSampleResponse</a>	34
<a href="#">ZBRxResponse</a>	35
<a href="#">ZBTxRequest</a>	36
<a href="#">ZBTxStatusResponse</a>	38



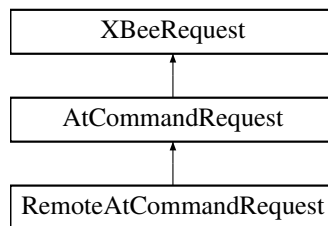
## Chapter 3

# Class Documentation

### 3.1 AtCommandRequest Class Reference

```
#include <XBee.h>
```

Inheritance diagram for AtCommandRequest:



#### Public Member Functions

- **AtCommandRequest** (uint8\_t \*command)
- **AtCommandRequest** (uint8\_t \*command, uint8\_t \*commandValue, uint8\_t commandValueLength)
- uint8\_t [getFrameData](#) (uint8\_t pos)
- uint8\_t [getFrameDataLength](#) ()
- uint8\_t \* **getCommand** ()
- void **setCommand** (uint8\_t \*command)
- uint8\_t \* **getCommandValue** ()
- void **setCommandValue** (uint8\_t \*command)
- uint8\_t **getCommandValueLength** ()
- void **setCommandValueLength** (uint8\_t length)
- void [clearCommandValue](#) ()

### 3.1.1 Detailed Description

Represents an AT Command TX packet The command is used to configure the serially connected [XBee](#) radio

### 3.1.2 Member Function Documentation

#### 3.1.2.1 void `AtCommandRequest::clearCommandValue` ( )

Clears the optional `commandValue` and `commandValueLength` so that a query may be sent

#### 3.1.2.2 `uint8_t AtCommandRequest::getFrameData` ( `uint8_t pos` ) [virtual]

Starting after the frame id (`pos = 0`) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements [XBeeRequest](#).

Reimplemented in [RemoteAtCommandRequest](#).

#### 3.1.2.3 `uint8_t AtCommandRequest::getFrameDataLength` ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements [XBeeRequest](#).

Reimplemented in [RemoteAtCommandRequest](#).

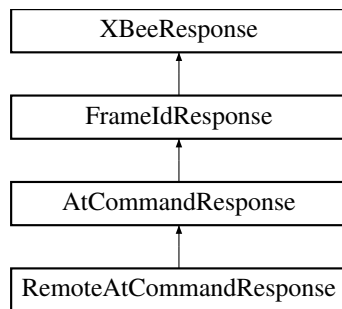
The documentation for this class was generated from the following files:

- `XBee.h`
- `XBee.cpp`

## 3.2 AtCommandResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for `AtCommandResponse`:



### Public Member Functions

- `uint8_t * getCommand ()`
- `uint8_t getStatus ()`
- `uint8_t * getValue ()`
- `uint8_t getValueLength ()`
- `bool isOk ()`

#### 3.2.1 Detailed Description

Represents an AT Command RX packet

#### 3.2.2 Member Function Documentation

##### 3.2.2.1 `uint8_t * AtCommandResponse::getCommand ( )`

Returns an array containing the two character command

Reimplemented in [RemoteAtCommandResponse](#).

##### 3.2.2.2 `uint8_t AtCommandResponse::getStatus ( )`

Returns the command status code. Zero represents a successful command

Reimplemented in [RemoteAtCommandResponse](#).

##### 3.2.2.3 `uint8_t * AtCommandResponse::getValue ( )`

Returns an array containing the command value. This is only applicable to query commands.

Reimplemented in [RemoteAtCommandResponse](#).

### 3.2.2.4 `uint8_t AtCommandResponse::getValueLength ( )`

Returns the length of the command value array.

Reimplemented in [RemoteAtCommandResponse](#).

### 3.2.2.5 `bool AtCommandResponse::isOk ( )`

Returns true if status equals AT\_OK

Reimplemented in [RemoteAtCommandResponse](#).

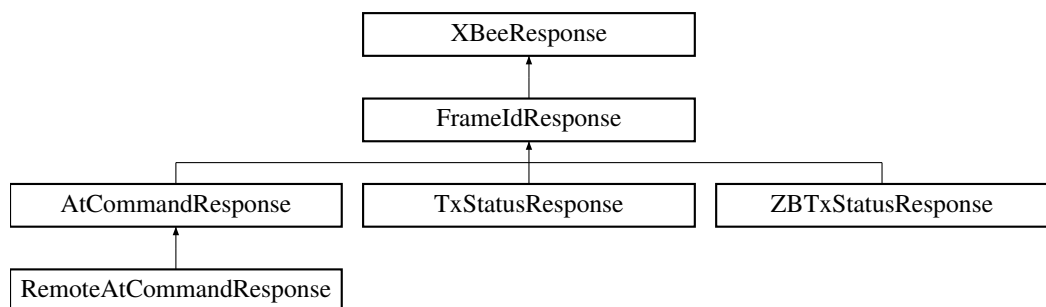
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.3 FrameldResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for FrameldResponse:



### Public Member Functions

- `uint8_t getFrameld ( )`

### 3.3.1 Detailed Description

This class is extended by all Responses that include a frame id

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

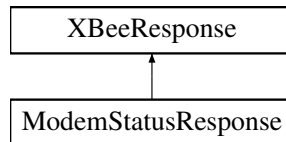
- XBee.h
- XBee.cpp



## 3.4 ModemStatusResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for ModemStatusResponse:



### Public Member Functions

- `uint8_t` **getStatus** ()

#### 3.4.1 Detailed Description

Represents a Modem Status RX packet

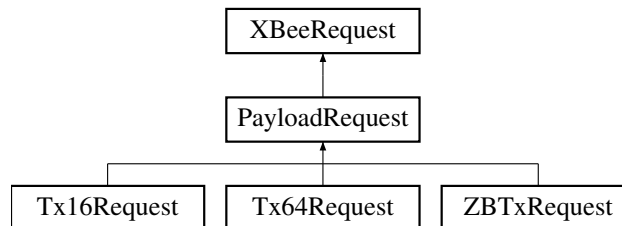
The documentation for this class was generated from the following files:

- `XBee.h`
- `XBee.cpp`

## 3.5 PayloadRequest Class Reference

```
#include <XBee.h>
```

Inheritance diagram for PayloadRequest:



### Public Member Functions

- **PayloadRequest** (`uint8_t` apid, `uint8_t` frameId, `uint8_t` \*payload, `uint8_t` payloadLength)
- `uint8_t` \* [getPayload](#) ()

- void [setPayload](#) (uint8\_t \*payloadPtr)
- uint8\_t [getPayloadLength](#) ()
- void [setPayloadLength](#) (uint8\_t payloadLength)

### 3.5.1 Detailed Description

All TX packets that support payloads extend this class

### 3.5.2 Member Function Documentation

#### 3.5.2.1 uint8\_t \* PayloadRequest::getPayload ( )

Returns the payload of the packet, if not null

#### 3.5.2.2 uint8\_t PayloadRequest::getPayloadLength ( )

Returns the length of the payload array, as specified by the user.

#### 3.5.2.3 void PayloadRequest::setPayload ( uint8\_t \* payloadPtr )

Sets the payload array

#### 3.5.2.4 void PayloadRequest::setPayloadLength ( uint8\_t payloadLength )

Sets the length of the payload to include in the request. For example if the payload array is 50 bytes and you only want the first 10 to be included in the packet, set the length to 10. Length must be  $\leq$  to the array length.

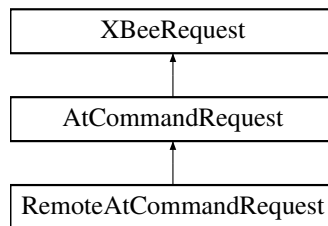
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.6 RemoteAtCommandRequest Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RemoteAtCommandRequest:



### Public Member Functions

- [RemoteAtCommandRequest](#) (uint16\_t remoteAddress16, uint8\_t \*command, uint8\_t \*commandValue, uint8\_t commandValueLength)
- [RemoteAtCommandRequest](#) (uint16\_t remoteAddress16, uint8\_t \*command)
- [RemoteAtCommandRequest](#) ([XBeeAddress64](#) &remoteAddress64, uint8\_t \*command, uint8\_t \*commandValue, uint8\_t commandValueLength)
- [RemoteAtCommandRequest](#) ([XBeeAddress64](#) &remoteAddress64, uint8\_t \*command)
- uint16\_t **getRemoteAddress16** ()
- void **setRemoteAddress16** (uint16\_t remoteAddress16)
- [XBeeAddress64](#) & **getRemoteAddress64** ()
- void **setRemoteAddress64** ([XBeeAddress64](#) &remoteAddress64)
- bool **getApplyChanges** ()
- void **setApplyChanges** (bool applyChanges)
- uint8\_t **getFrameData** (uint8\_t pos)
- uint8\_t **getFrameDataLength** ()

### Static Public Attributes

- static [XBeeAddress64](#) **broadcastAddress64** = [XBeeAddress64](#)(0x0, BROADCAST\_ADDRESS)

#### 3.6.1 Detailed Description

Represents an Remote AT Command TX packet The command is used to configure a remote [XBee](#) radio

#### 3.6.2 Constructor & Destructor Documentation

##### 3.6.2.1 RemoteAtCommandRequest::RemoteAtCommandRequest ( uint16\_t remoteAddress16, uint8\_t \* command, uint8\_t \* commandValue, uint8\_t commandValueLength )

Creates a [RemoteAtCommandRequest](#) with 16-bit address to set a command. 64-bit address defaults to broadcast and applyChanges is true.

### 3.6.2.2 RemoteAtCommandRequest::RemoteAtCommandRequest ( uint16\_t remoteAddress16, uint8\_t \* command )

Creates a [RemoteAtCommandRequest](#) with 16-bit address to query a command. 64-bit address defaults to broadcast and applyChanges is true.

### 3.6.2.3 RemoteAtCommandRequest::RemoteAtCommandRequest ( XBeeAddress64 & remoteAddress64, uint8\_t \* command, uint8\_t \* commandValue, uint8\_t commandValueLength )

Creates a [RemoteAtCommandRequest](#) with 64-bit address to set a command. 16-bit address defaults to broadcast and applyChanges is true.

### 3.6.2.4 RemoteAtCommandRequest::RemoteAtCommandRequest ( XBeeAddress64 & remoteAddress64, uint8\_t \* command )

Creates a [RemoteAtCommandRequest](#) with 16-bit address to query a command. 16-bit address defaults to broadcast and applyChanges is true.

## 3.6.3 Member Function Documentation

### 3.6.3.1 uint8\_t RemoteAtCommandRequest::getFrameData ( uint8\_t pos ) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Reimplemented from [AtCommandRequest](#).

### 3.6.3.2 uint8\_t RemoteAtCommandRequest::getFrameDataLength ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Reimplemented from [AtCommandRequest](#).

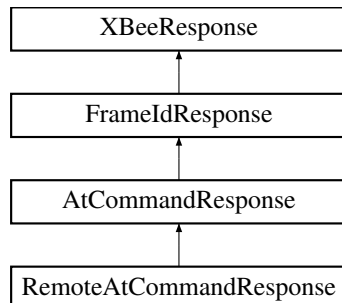
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.7 RemoteAtCommandResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RemoteAtCommandResponse:



### Public Member Functions

- `uint8_t * getCommand ()`
- `uint8_t getStatus ()`
- `uint8_t * getValue ()`
- `uint8_t getValueLength ()`
- `uint16_t getRemoteAddress16 ()`
- `XBeeAddress64 & getRemoteAddress64 ()`
- `bool isOk ()`

#### 3.7.1 Detailed Description

Represents a Remote AT Command RX packet

#### 3.7.2 Member Function Documentation

##### 3.7.2.1 `uint8_t * RemoteAtCommandResponse::getCommand ( )`

Returns an array containing the two character command

Reimplemented from [AtCommandResponse](#).

##### 3.7.2.2 `uint16_t RemoteAtCommandResponse::getRemoteAddress16 ( )`

Returns the 16-bit address of the remote radio

##### 3.7.2.3 `XBeeAddress64 & RemoteAtCommandResponse::getRemoteAddress64 ( )`

Returns the 64-bit address of the remote radio

#### 3.7.2.4 `uint8_t RemoteAtCommandResponse::getStatus ( )`

Returns the command status code. Zero represents a successful command

Reimplemented from [AtCommandResponse](#).

#### 3.7.2.5 `uint8_t * RemoteAtCommandResponse::getValue ( )`

Returns an array containing the command value. This is only applicable to query commands.

Reimplemented from [AtCommandResponse](#).

#### 3.7.2.6 `uint8_t RemoteAtCommandResponse::getValueLength ( )`

Returns the length of the command value array.

Reimplemented from [AtCommandResponse](#).

#### 3.7.2.7 `bool RemoteAtCommandResponse::isOk ( )`

Returns true if command was successful

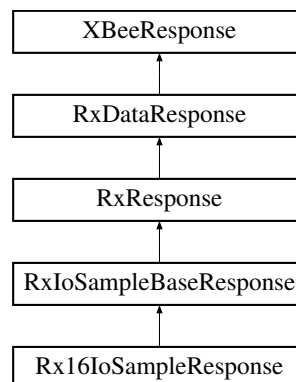
Reimplemented from [AtCommandResponse](#).

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

- XBee.h
- XBee.cpp

## 3.8 Rx16IoSampleResponse Class Reference

Inheritance diagram for Rx16IoSampleResponse:



### Public Member Functions

- uint16\_t **getRemoteAddress16** ()
- uint8\_t **getRssiOffset** ()

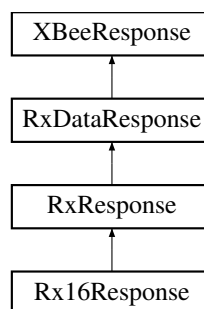
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.9 Rx16Response Class Reference

```
#include <XBee.h>
```

Inheritance diagram for Rx16Response:



### Public Member Functions

- uint8\_t **getRssiOffset** ()
- uint16\_t **getRemoteAddress16** ()

### Protected Attributes

- uint16\_t **\_remoteAddress**

#### 3.9.1 Detailed Description

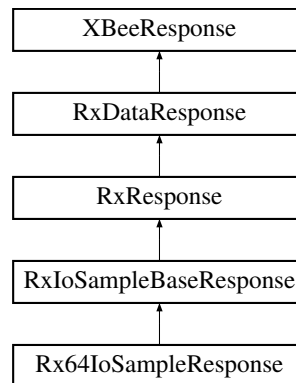
Represents a Series 1 16-bit address RX packet

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

- XBee.h
- XBee.cpp

### 3.10 Rx64IoSampleResponse Class Reference

Inheritance diagram for Rx64IoSampleResponse:



#### Public Member Functions

- [XBeeAddress64](#) & **getRemoteAddress64** ()
- `uint8_t` **getRssiOffset** ()

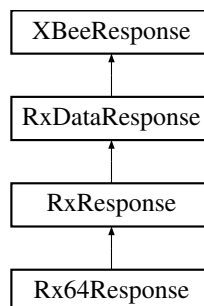
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

### 3.11 Rx64Response Class Reference

```
#include <XBee.h>
```

Inheritance diagram for Rx64Response:





### Public Member Functions

- uint8\_t **getRssiOffset** ()
- [XBeeAddress64](#) & **getRemoteAddress64** ()

#### 3.11.1 Detailed Description

Represents a Series 1 64-bit address RX packet

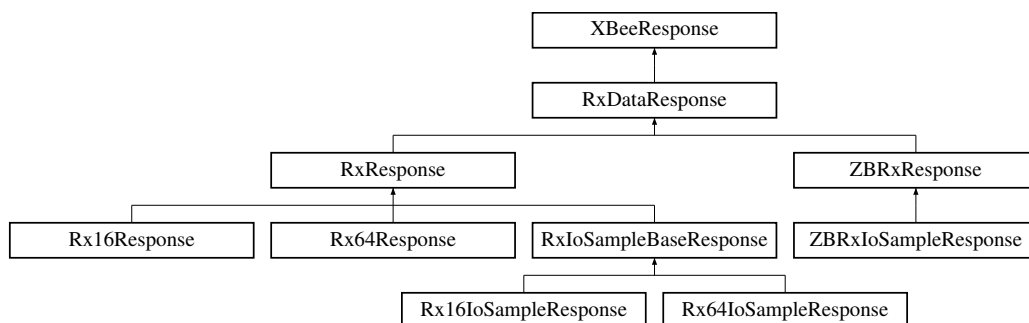
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.12 RxDataResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RxDataResponse:



### Public Member Functions

- uint8\_t **getData** (int index)
- uint8\_t \* **getData** ()
- virtual uint8\_t **getDataLength** ()=0
- virtual uint8\_t **getDataOffset** ()=0

#### 3.12.1 Detailed Description

Common functionality for both Series 1 and 2 data RX data packets

### 3.12.2 Member Function Documentation

#### 3.12.2.1 `uint8_t RxDataResponse::getData ( int index )`

Returns the specified index of the payload. The index may be 0 to `getDataLength()` - 1  
This method is deprecated; use `uint8_t* getData()`

#### 3.12.2.2 `uint8_t* RxDataResponse::getData ( )`

Returns the payload array. This may be accessed from index 0 to `getDataLength()` - 1

#### 3.12.2.3 `virtual uint8_t RxDataResponse::getDataLength ( )` [pure virtual]

Returns the length of the payload

Implemented in [RxResponse](#), and [ZBRxResponse](#).

#### 3.12.2.4 `virtual uint8_t RxDataResponse::getDataOffset ( )` [pure virtual]

Returns the position in the frame data where the data begins

Implemented in [RxResponse](#), and [ZBRxResponse](#).

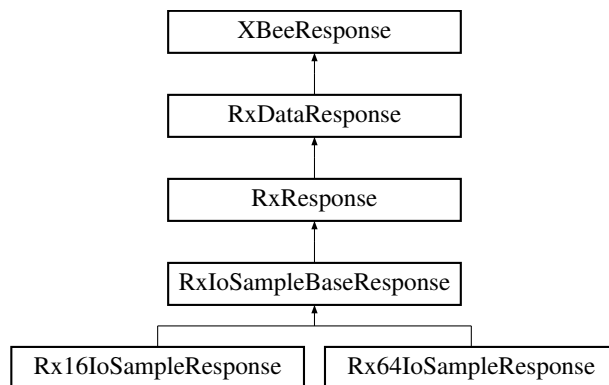
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.13 RxIoSampleBaseResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RxIoSampleBaseResponse:



## Public Member Functions

- uint8\_t [getSampleSize](#) ()
- bool **containsAnalog** ()
- bool **containsDigital** ()
- bool [isAnalogEnabled](#) (uint8\_t pin)
- bool [isDigitalEnabled](#) (uint8\_t pin)
- uint16\_t [getAnalog](#) (uint8\_t pin, uint8\_t sample)
- bool [isDigitalOn](#) (uint8\_t pin, uint8\_t sample)
- uint8\_t **getSampleOffset** ()

### 3.13.1 Detailed Description

Represents a Series 1 RX I/O Sample packet

### 3.13.2 Member Function Documentation

#### 3.13.2.1 uint16\_t RxIoSampleBaseResponse::getAnalog ( uint8\_t *pin*, uint8\_t *sample* )

Returns the 10-bit analog reading of the specified pin. Valid pins include ADC:0-5. Sample index starts at 0

#### 3.13.2.2 uint8\_t RxIoSampleBaseResponse::getSampleSize ( )

Returns the number of samples in this packet

#### 3.13.2.3 bool RxIoSampleBaseResponse::isAnalogEnabled ( uint8\_t *pin* )

Returns true if the specified analog pin is enabled

#### 3.13.2.4 bool RxIoSampleBaseResponse::isDigitalEnabled ( uint8\_t *pin* )

Returns true if the specified digital pin is enabled

#### 3.13.2.5 bool RxIoSampleBaseResponse::isDigitalOn ( uint8\_t *pin*, uint8\_t *sample* )

Returns true if the specified pin is high/on. Valid pins include DIO:0-8. Sample index starts at 0

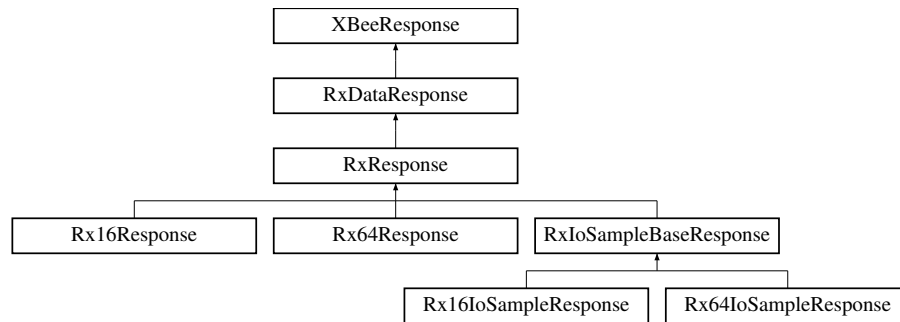
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

### 3.14 RxResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for RxResponse:



#### Public Member Functions

- uint8\_t **getRssi** ()
- uint8\_t **getOption** ()
- bool **isAddressBroadcast** ()
- bool **isPanBroadcast** ()
- uint8\_t [getDataLength](#) ()
- uint8\_t [getDataOffset](#) ()
- virtual uint8\_t **getRssiOffset** ()=0

#### 3.14.1 Detailed Description

Represents a Series 1 RX packet

#### 3.14.2 Member Function Documentation

##### 3.14.2.1 uint8\_t RxResponse::getDataLength ( ) [virtual]

Returns the length of the payload

Implements [RxDataResponse](#).

##### 3.14.2.2 uint8\_t RxResponse::getDataOffset ( ) [virtual]

Returns the position in the frame data where the data begins

Implements [RxDataResponse](#).

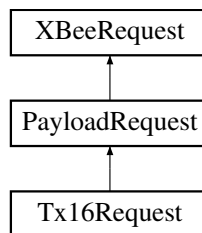
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.15 Tx16Request Class Reference

```
#include <XBee.h>
```

Inheritance diagram for Tx16Request:



### Public Member Functions

- **Tx16Request** (uint16\_t addr16, uint8\_t option, uint8\_t \*payload, uint8\_t payloadLength, uint8\_t frameId)
- [Tx16Request](#) (uint16\_t addr16, uint8\_t \*payload, uint8\_t payloadLength)
- [Tx16Request](#) ()
- uint16\_t **getAddress16** ()
- void **setAddress16** (uint16\_t addr16)
- uint8\_t **getOption** ()
- void **setOption** (uint8\_t option)
- uint8\_t [getFrameData](#) (uint8\_t pos)
- uint8\_t [getFrameDataLength](#) ()

#### 3.15.1 Detailed Description

Represents a Series 1 TX packet that corresponds to Api Id: TX\_16\_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 100 bytes is the maximum, although that could change in future firmware.

#### 3.15.2 Constructor & Destructor Documentation

3.15.2.1 **Tx16Request::Tx16Request** ( uint16\_t *addr16*, uint8\_t \* *payload*, uint8\_t *payloadLength* )

Creates a Unicast [Tx16Request](#) with the ACK option and DEFAULT\_FRAME\_ID

### 3.15.2.2 Tx16Request::Tx16Request ( )

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

## 3.15.3 Member Function Documentation

### 3.15.3.1 uint8\_t Tx16Request::getFrameData ( uint8\_t pos ) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements [XBeeRequest](#).

### 3.15.3.2 uint8\_t Tx16Request::getFrameDataLength ( ) [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements [XBeeRequest](#).

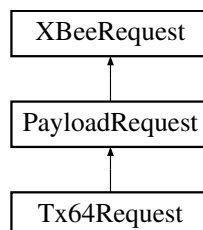
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.16 Tx64Request Class Reference

```
#include <XBee.h>
```

Inheritance diagram for Tx64Request:



### Public Member Functions

- **Tx64Request** ([XBeeAddress64](#) &addr64, uint8\_t option, uint8\_t \*payload, uint8\_t payloadLength, uint8\_t frameId)

- [Tx64Request](#) ([XBeeAddress64](#) &addr64, uint8\_t \*payload, uint8\_t payloadLength)
- [Tx64Request](#) ()
- [XBeeAddress64](#) & [getAddress64](#) ()
- void [setAddress64](#) ([XBeeAddress64](#) &addr64)
- uint8\_t [getOption](#) ()
- void [setOption](#) (uint8\_t option)
- uint8\_t [getFrameData](#) (uint8\_t pos)
- uint8\_t [getFrameDataLength](#) ()

### 3.16.1 Detailed Description

Represents a Series 1 TX packet that corresponds to Api Id: TX\_64\_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 100 bytes is the maximum, although that could change in future firmware.

### 3.16.2 Constructor & Destructor Documentation

3.16.2.1 `Tx64Request::Tx64Request ( XBeeAddress64 & addr64, uint8_t * payload, uint8_t payloadLength )`

Creates a unicast [Tx64Request](#) with the ACK option and DEFAULT\_FRAME\_ID

3.16.2.2 `Tx64Request::Tx64Request ( )`

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

### 3.16.3 Member Function Documentation

3.16.3.1 `uint8_t Tx64Request::getFrameData ( uint8_t pos ) [virtual]`

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements [XBeeRequest](#).

3.16.3.2 `uint8_t Tx64Request::getFrameDataLength ( ) [virtual]`

Returns the size of the api frame (not including frame id or api id or checksum).

Implements [XBeeRequest](#).

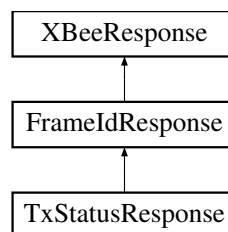
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

### 3.17 TxStatusResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for TxStatusResponse:



#### Public Member Functions

- `uint8_t` **getStatus** ()
- `bool` **isSuccess** ()

#### 3.17.1 Detailed Description

Represents a Series 1 TX Status packet

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

- XBee.h
- XBee.cpp

### 3.18 XBee Class Reference

```
#include <XBee.h>
```

#### Public Member Functions

- `void` [readPacket](#) ()
- `bool` [readPacket](#) (int timeout)
- `void` [readPacketUntilAvailable](#) ()



- void [begin](#) (long baud)
- void [getResponse](#) ([XBeeResponse](#) &response)
- [XBeeResponse](#) & [getResponse](#) ()
- void [send](#) ([XBeeRequest](#) &request)
- uint8\_t [getNextFrameId](#) ()
- void [setSerial](#) (HardwareSerial &serial)

### 3.18.1 Detailed Description

Primary interface for communicating with an [XBee](#) Radio. This class provides methods for sending and receiving packets with an [XBee](#) radio via the serial port. The [XBee](#) radio must be configured in API (packet) mode (AP=2) in order to use this software.

Since this code is designed to run on a microcontroller, with only one thread, you are responsible for reading the data off the serial buffer in a timely manner. This involves a call to a variant of `readPacket(...)`. If your serial port is receiving data faster than you are reading, you can expect to lose packets. Arduino only has a 128 byte serial buffer so it can easily overflow if two or more packets arrive without a call to `readPacket(...)`

In order to conserve resources, this class only supports storing one response packet in memory at a time. This means that you must fully consume the packet prior to calling `readPacket(...)`, because calling `readPacket(...)` overwrites the previous response.

This class creates an array of size `MAX_FRAME_DATA_SIZE` for storing the response packet. You may want to adjust this value to conserve memory.

#### Author

Andrew Rapp

### 3.18.2 Member Function Documentation

#### 3.18.2.1 void XBee::begin ( long *baud* )

Starts the serial connection at the supplied baud rate

#### 3.18.2.2 uint8\_t XBee::getNextFrameId ( )

Returns a sequential frame id between 1 and 255

#### 3.18.2.3 XBeeResponse & XBee::getResponse ( )

Returns a reference to the current response Note: once `readPacket` is called again this response will be overwritten!

#### 3.18.2.4 void XBee::readPacket ( )

Reads all available serial bytes until a packet is parsed, an error occurs, or the buffer is empty. You may call `xbee.getResponse().isAvailable()` after calling this method to determine if a packet is ready, or `xbee.getResponse().isError()` to determine if a error occurred.

This method should always return quickly since it does not wait for serial data to arrive. You will want to use this method if you are doing other timely stuff in your loop, where a delay would cause problems. NOTE: calling this method resets the current response, so make sure you first consume the current response

#### 3.18.2.5 bool XBee::readPacket ( int *timeout* )

Waits a maximum of *timeout* milliseconds for a response packet before timing out; returns true if packet is read. Returns false if timeout or error occurs.

#### 3.18.2.6 void XBee::readPacketUntilAvailable ( )

Reads until a packet is received or an error occurs. Caution: use this carefully since if you don't get a response, your Arduino code will hang on this call forever!! often it's better to use a timeout: [readPacket\(int\)](#)

#### 3.18.2.7 void XBee::send ( XBeeRequest & *request* )

Sends a [XBeeRequest](#) (TX packet) out the serial port

#### 3.18.2.8 void XBee::setSerial ( HardwareSerial & *serial* )

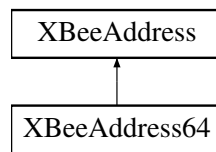
Specify the serial port. Only relevant for Arduinos that support multiple serial ports (e.g. Mega)

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

- XBee.h
- XBee.cpp

## 3.19 XBeeAddress Class Reference

Inheritance diagram for XBeeAddress:



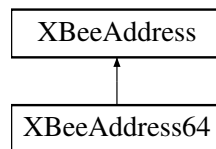
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.20 XBeeAddress64 Class Reference

```
#include <XBee.h>
```

Inheritance diagram for XBeeAddress64:



### Public Member Functions

- **XBeeAddress64** (uint32\_t msb, uint32\_t lsb)
- uint32\_t **getMsb** ()
- uint32\_t **getLsb** ()
- void **setMsb** (uint32\_t msb)
- void **setLsb** (uint32\_t lsb)

#### 3.20.1 Detailed Description

Represents a 64-bit [XBee](#) Address

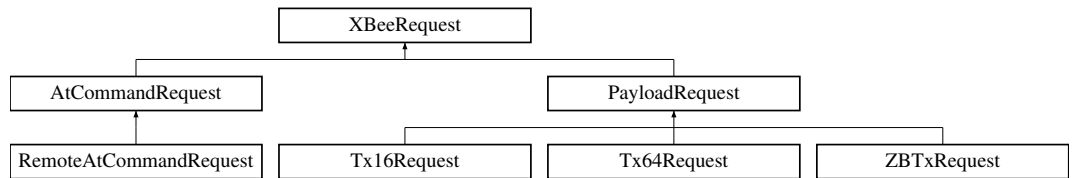
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.21 XBeeRequest Class Reference

```
#include <XBee.h>
```

Inheritance diagram for XBeeRequest:



## Public Member Functions

- [XBeeRequest](#) (uint8\_t apild, uint8\_t frameld)
- void [setFrameld](#) (uint8\_t frameld)
- uint8\_t [getFrameld](#) ()
- uint8\_t [getApild](#) ()
- virtual uint8\_t [getFrameData](#) (uint8\_t pos)=0
- virtual uint8\_t [getFrameDataLength](#) ()=0

## Protected Member Functions

- void **setApild** (uint8\_t apild)

### 3.21.1 Detailed Description

Super class of all [XBee](#) requests (TX packets) Users should never create an instance of this class; instead use an subclass of this class It is recommended to reuse Subclasses of the class to conserve memory

This class allocates a buffer to

### 3.21.2 Constructor & Destructor Documentation

#### 3.21.2.1 [XBeeRequest::XBeeRequest](#) ( uint8\_t *apild*, uint8\_t *frameld* )

Constructor TODO make protected

### 3.21.3 Member Function Documentation

#### 3.21.3.1 uint8\_t [XBeeRequest::getApild](#) ( )

Returns the API id

**3.21.3.2** `virtual uint8_t XBeeRequest::getFrameData ( uint8_t pos )` [pure virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implemented in [RemoteAtCommandRequest](#), [AtCommandRequest](#), [ZBTxRequest](#), - [Tx64Request](#), and [Tx16Request](#).

**3.21.3.3** `virtual uint8_t XBeeRequest::getFrameDataLength ( )` [pure virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implemented in [RemoteAtCommandRequest](#), [AtCommandRequest](#), [ZBTxRequest](#), - [Tx64Request](#), and [Tx16Request](#).

**3.21.3.4** `uint8_t XBeeRequest::getFrameId ( )`

Returns the frame id

**3.21.3.5** `void XBeeRequest::setFrameId ( uint8_t frameId )`

Sets the frame id. Must be between 1 and 255 inclusive to get a TX status response.

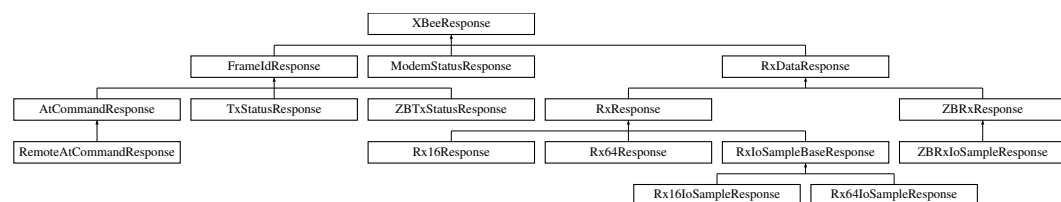
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.22 XBeeResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for XBeeResponse:



## Public Member Functions

- [XBeeResponse](#) ()
- uint8\_t [getApild](#) ()
- void **setApild** (uint8\_t apild)
- uint8\_t [getMsbLength](#) ()
- void **setMsbLength** (uint8\_t msbLength)
- uint8\_t [getLsbLength](#) ()
- void **setLsbLength** (uint8\_t lsbLength)
- uint8\_t [getChecksum](#) ()
- void **setChecksum** (uint8\_t checksum)
- uint8\_t [getFrameDataLength](#) ()
- void **setFrameData** (uint8\_t \*frameDataPtr)
- uint8\_t \* [getFrameData](#) ()
- void **setFrameLength** (uint8\_t frameLength)
- uint16\_t [getPacketLength](#) ()
- void [reset](#) ()
- void [init](#) ()
- void [getZBTxStatusResponse](#) (XBeeResponse &response)
- void [getZBRxResponse](#) (XBeeResponse &response)
- void [getZBRxIoSampleResponse](#) (XBeeResponse &response)
- void [getTxStatusResponse](#) (XBeeResponse &response)
- void [getRx16Response](#) (XBeeResponse &response)
- void [getRx64Response](#) (XBeeResponse &response)
- void [getRx16IoSampleResponse](#) (XBeeResponse &response)
- void [getRx64IoSampleResponse](#) (XBeeResponse &response)
- void [getAtCommandResponse](#) (XBeeResponse &responses)
- void [getRemoteAtCommandResponse](#) (XBeeResponse &response)
- void [getModemStatusResponse](#) (XBeeResponse &response)
- bool [isAvailable](#) ()
- void **setAvailable** (bool complete)
- bool [isError](#) ()
- uint8\_t [getErrorCode](#) ()
- void **setErrorCode** (uint8\_t errorCode)

## Protected Attributes

- uint8\_t \* [\\_frameDataPtr](#)

### 3.22.1 Detailed Description

The super class of all [XBee](#) responses (RX packets) Users should never attempt to create an instance of this class; instead create an instance of a subclass It is recommend to reuse subclasses to conserve memory

### 3.22.2 Constructor & Destructor Documentation

#### 3.22.2.1 XBeeResponse::XBeeResponse ( )

Default constructor

Copyright (c) 2009 Andrew Rapp. All rights reserved.

This file is part of XBee-Arduino.

XBee-Arduino is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

XBee-Arduino is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with XBee-Arduino. If not, see <<http://www.gnu.org/licenses/>>.

### 3.22.3 Member Function Documentation

#### 3.22.3.1 uint8\_t XBeeResponse::getApId ( )

Returns Api Id of the response

#### 3.22.3.2 void XBeeResponse::getAtCommandResponse ( XBeeResponse & responses )

Call with instance of [AtCommandResponse](#) only if [getApId\(\)](#) == AT\_COMMAND\_RESPONSE

#### 3.22.3.3 uint8\_t XBeeResponse::getChecksum ( )

Returns the packet checksum

#### 3.22.3.4 uint8\_t XBeeResponse::getErrorCode ( )

Returns an error code, or zero, if successful. Error codes include: CHECKSUM\_FAILURE, PACKET\_EXCEEDS\_BYTE\_ARRAY\_LENGTH, UNEXPECTED\_START\_BYTE

#### 3.22.3.5 uint8\_t \* XBeeResponse::getFrameData ( )

Returns the buffer that contains the response. Starts with byte that follows API ID and includes all bytes prior to the checksum Length is specified by [getFrameDataLength\(\)](#)

Note: Unlike Digi's definition of the frame data, this does not start with the API ID.. The

reason for this is all responses include an API ID, whereas my frame data includes only the API specific data.

### 3.22.3.6 `uint8_t XBeeResponse::getFrameDataLength ( )`

Returns the length of the frame data: all bytes after the api id, and prior to the checksum  
Note up to release 0.1.2, this was incorrectly including the checksum in the length.

### 3.22.3.7 `uint8_t XBeeResponse::getLsbLength ( )`

Returns the LSB length of the packet

### 3.22.3.8 `void XBeeResponse::getModemStatusResponse ( XBeeResponse & response )`

Call with instance of [ModemStatusResponse](#) only if `getApild() == MODEM_STATUS_RESPONSE`

### 3.22.3.9 `uint8_t XBeeResponse::getMsbLength ( )`

Returns the MSB length of the packet

### 3.22.3.10 `uint16_t XBeeResponse::getPacketLength ( )`

Returns the length of the packet

### 3.22.3.11 `void XBeeResponse::getRemoteAtCommandResponse ( XBeeResponse & response )`

Call with instance of [RemoteAtCommandResponse](#) only if `getApild() == REMOTE_AT_COMMAND_RESPONSE`

### 3.22.3.12 `void XBeeResponse::getRx16IoSampleResponse ( XBeeResponse & response )`

Call with instance of [Rx16IoSampleResponse](#) only if `getApild() == RX_16_IO_RESPONSE`

### 3.22.3.13 `void XBeeResponse::getRx16Response ( XBeeResponse & response )`

Call with instance of [Rx16Response](#) only if `getApild() == RX_16_RESPONSE`



**3.22.3.14** void XBeeResponse::getRx64IoSampleResponse ( XBeeResponse & response )

Call with instance of [Rx64IoSampleResponse](#) only if [getApild\(\)](#) == RX\_64\_IO\_RESPONSE

**3.22.3.15** void XBeeResponse::getRx64Response ( XBeeResponse & response )

Call with instance of [Rx64Response](#) only if [getApild\(\)](#) == RX\_64\_RESPONSE

**3.22.3.16** void XBeeResponse::getTxStatusResponse ( XBeeResponse & response )

Call with instance of [TxStatusResponse](#) only if [getApild\(\)](#) == TX\_STATUS\_RESPONSE

**3.22.3.17** void XBeeResponse::getZBRxIoSampleResponse ( XBeeResponse & response )

Call with instance of [ZBRxIoSampleResponse](#) class only if [getApild\(\)](#) == ZB\_IO\_SAMPLE\_RESPONSE to populate response

**3.22.3.18** void XBeeResponse::getZBRxResponse ( XBeeResponse & response )

Call with instance of [ZBRxResponse](#) class only if [getApild\(\)](#) == ZB\_RX\_RESPONSE to populate response

**3.22.3.19** void XBeeResponse::getZBTxStatusResponse ( XBeeResponse & response )

Call with instance of [ZBTxStatusResponse](#) class only if [getApild\(\)](#) == ZB\_TX\_STATUS\_RESPONSE to populate response

**3.22.3.20** void XBeeResponse::init ( )

Initializes the response

**3.22.3.21** bool XBeeResponse::isAvailable ( )

Returns true if the response has been successfully parsed and is complete and ready for use

**3.22.3.22** bool XBeeResponse::isError ( )

Returns true if the response contains errors

### 3.22.3.23 void XBeeResponse::reset ( )

Resets the response to default values

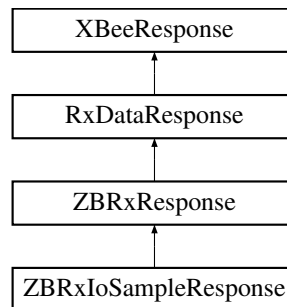
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.23 ZBRxIoSampleResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for ZBRxIoSampleResponse:



### Public Member Functions

- bool **containsAnalog** ( )
- bool **containsDigital** ( )
- bool **isAnalogEnabled** (uint8\_t pin)
- bool **isDigitalEnabled** (uint8\_t pin)
- uint16\_t **getAnalog** (uint8\_t pin)
- bool **isDigitalOn** (uint8\_t pin)
- uint8\_t **getDigitalMaskMsb** ( )
- uint8\_t **getDigitalMaskLsb** ( )
- uint8\_t **getAnalogMask** ( )

### 3.23.1 Detailed Description

Represents a Series 2 RX I/O Sample packet

### 3.23.2 Member Function Documentation

#### 3.23.2.1 uint16\_t ZBRxIoSampleResponse::getAnalog ( uint8\_t *pin* )

Returns the 10-bit analog reading of the specified pin. Valid pins include ADC:xxx.

#### 3.23.2.2 bool ZBRxIoSampleResponse::isAnalogEnabled ( uint8\_t *pin* )

Returns true if the pin is enabled

#### 3.23.2.3 bool ZBRxIoSampleResponse::isDigitalEnabled ( uint8\_t *pin* )

Returns true if the pin is enabled

#### 3.23.2.4 bool ZBRxIoSampleResponse::isDigitalOn ( uint8\_t *pin* )

Returns true if the specified pin is high/on. Valid pins include DIO:xxx.

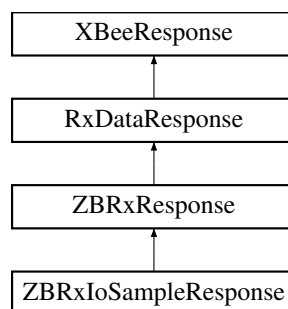
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.24 ZBRxResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for ZBRxResponse:



### Public Member Functions

- [XBeeAddress64](#) & `getRemoteAddress64 ()`
- uint16\_t `getRemoteAddress16 ()`
- uint8\_t `getOption ()`

- `uint8_t` [getLength](#) ()
- `uint8_t` [getDataOffset](#) ()

### 3.24.1 Detailed Description

Represents a Series 2 RX packet

### 3.24.2 Member Function Documentation

#### 3.24.2.1 `uint8_t ZB RxResponse::getLength ( )` [virtual]

Returns the length of the payload

Implements [RxDataResponse](#).

#### 3.24.2.2 `uint8_t ZB RxResponse::getDataOffset ( )` [virtual]

Returns the position in the frame data where the data begins

Implements [RxDataResponse](#).

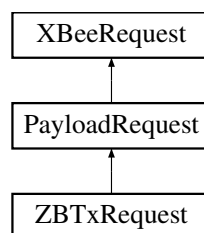
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.25 ZBTxRequest Class Reference

```
#include <XBee.h>
```

Inheritance diagram for ZBTxRequest:



### Public Member Functions

- [ZBTxRequest](#) ([XBeeAddress64](#) &addr64, `uint8_t` \*payload, `uint8_t` payloadLength)

- **ZBTxRequest** ([XBeeAddress64](#) &addr64, uint16\_t addr16, uint8\_t broadcastRadius, uint8\_t option, uint8\_t \*payload, uint8\_t payloadLength, uint8\_t frameId)
- [ZBTxRequest](#) ()
- [XBeeAddress64](#) & **getAddress64** ()
- uint16\_t **getAddress16** ()
- uint8\_t **getBroadcastRadius** ()
- uint8\_t **getOption** ()
- void **setAddress64** ([XBeeAddress64](#) &addr64)
- void **setAddress16** (uint16\_t addr16)
- void **setBroadcastRadius** (uint8\_t broadcastRadius)
- void **setOption** (uint8\_t option)

### Protected Member Functions

- uint8\_t [getFrameData](#) (uint8\_t pos)
- uint8\_t [getFrameDataLength](#) ()

### 3.25.1 Detailed Description

Represents a Series 2 TX packet that corresponds to Api Id: ZB\_TX\_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 72 bytes is the maximum for ZNet firmware and ZB Pro firmware provides the AT-NP command to get the max supported payload size. This command is useful since the maximum payload size varies according to certain settings, such as encryption. ZB Pro firmware provides a PAYLOAD\_TOO\_LARGE that is returned if payload size exceeds the maximum.

### 3.25.2 Constructor & Destructor Documentation

#### 3.25.2.1 ZBTxRequest::ZBTxRequest ( [XBeeAddress64](#) & addr64, uint8\_t \* payload, uint8\_t payloadLength )

Creates a unicast [ZBTxRequest](#) with the ACK option and DEFAULT\_FRAME\_ID

#### 3.25.2.2 ZBTxRequest::ZBTxRequest ( )

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

### 3.25.3 Member Function Documentation

#### 3.25.3.1 `uint8_t ZBTxRequest::getFrameData ( uint8_t pos )` [protected, virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements [XBeeRequest](#).

#### 3.25.3.2 `uint8_t ZBTxRequest::getFrameDataLength ( )` [protected, virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements [XBeeRequest](#).

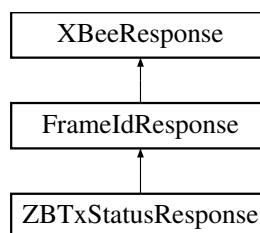
The documentation for this class was generated from the following files:

- XBee.h
- XBee.cpp

## 3.26 ZBTxStatusResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for ZBTxStatusResponse:



### Public Member Functions

- `uint16_t getRemoteAddress ()`
- `uint8_t getTxRetryCount ()`
- `uint8_t getDeliveryStatus ()`
- `uint8_t getDiscoveryStatus ()`
- `bool isSuccess ()`

### 3.26.1 Detailed Description

Represents a Series 2 TX status packet

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

- XBee.h
- XBee.cpp