

# Package ‘RProtoBuf’

September 11, 2010

**Version** 0.2.0

**Date** \$Date: 2010-09-10 06:54:37 -0500 (Fri, 10 Sep 2010) \$

**Author** Romain Francois <romain@r-enthusiasts.com> and Dirk Eddelbuettel  
<edd@debian.org>

**Maintainer** Romain and Dirk <RomainAndDirk@r-enthusiasts.com>

**Title** R Interface to the Protocol Buffers API

**Description** Protocol Buffers are a way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.

**Depends** R (>= 2.11.0), RCurl, Rcpp (>= 0.8.6), methods

**LinkingTo** Rcpp

**Suggests** RUnit, highlight

**SystemRequirements** Protocol Buffer compiler (to create C++ header and source files from .proto descriptions) and library (version 2.2.0 or later)

**License** GPL-2

**URL** <http://r-forge.r-project.org/projects/rprotobuf/>,  
<http://romainfrancois.blog.free.fr/index.php?category/R-package/RProtoBuf>, <http://dirk.eddelbuettel.com/blog/code/rprotobuf/>

**MinimumLibProtoVersion** 2002000

**BugReports** [http://r-forge.r-project.org/tracker/index.php?group\\_id=576&atid=2338](http://r-forge.r-project.org/tracker/index.php?group_id=576&atid=2338)

**OS\_type** unix

**Repository** CRAN

**Date/Publication** 2010-09-11 18:00:20

**R topics documented:**

RProtoBuf-package . . . . .	3
add-methods . . . . .	4
ArrayInputStream-class . . . . .	5
ArrayInputStream-methods . . . . .	6
ArrayOutputStream-class . . . . .	6
ArrayOutputStream-methods . . . . .	7
as.list.Message . . . . .	7
asMessage . . . . .	9
BackUp-methods . . . . .	10
ByteCount-methods . . . . .	10
bytesize-methods . . . . .	10
clear-methods . . . . .	11
clone-methods . . . . .	11
completion . . . . .	12
ConnectionInputStream-class . . . . .	13
ConnectionInputStream-methods . . . . .	14
ConnectionOutputStream-class . . . . .	15
ConnectionOutputStream-methods . . . . .	15
containing_type-methods . . . . .	16
Descriptor-class . . . . .	16
descriptor-methods . . . . .	18
EnumDescriptor-class . . . . .	18
EnumValueDescriptor-class . . . . .	20
enum_type-methods . . . . .	20
enum_type_count-methods . . . . .	21
fetch-methods . . . . .	21
field-methods . . . . .	21
FieldDescriptor-class . . . . .	22
field_count-methods . . . . .	23
FileDescriptor-class . . . . .	24
fileDescriptor-methods . . . . .	24
FileInputStream-class . . . . .	25
FileInputStream-methods . . . . .	26
FileOutputStream-class . . . . .	26
FileOutputStream-methods . . . . .	27
GetErrno-methods . . . . .	28
has-methods . . . . .	28
invoke-methods . . . . .	28
isInitialized-methods . . . . .	29
is_extension-methods . . . . .	29
label-methods . . . . .	29
merge-methods . . . . .	30
Message-class . . . . .	30
MethodDescriptor-class . . . . .	33
name . . . . .	34
nested_type-methods . . . . .	34

nested_type_count-methods . . . . .	34
Next-methods . . . . .	35
number-methods . . . . .	35
P . . . . .	35
read-methods . . . . .	36
readASCII-methods . . . . .	37
readProtoFiles . . . . .	37
RpcHTTP-class . . . . .	38
ServiceDescriptor-class . . . . .	39
set-methods . . . . .	39
SetCloseOnDelete-methods . . . . .	40
size-methods . . . . .	40
sizegets . . . . .	40
Skip-methods . . . . .	40
swap-methods . . . . .	41
type-methods . . . . .	41
with.Message . . . . .	41
ZeroCopyInputStream-class . . . . .	42
ZeroCopyOutputStream-class . . . . .	44
<b>Index</b>	<b>45</b>

---

RProtoBuf-package *R Interface to the Protocol Buffers API*

---

## Description

Protocol Buffers are a way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.

This package provides R API to create, manipulate, parse and serialize protocol buffer messages from R

## Details

Package:	RProtoBuf
Version:	0.1-0
Date:	\$Date\$
Depends:	Rcpp (>= 0.7.2), methods
SystemRequirements:	ProtoBuf compiler (to create C++ header and source files from .proto descriptions) and library
License:	GPL-2
URL:	<a href="http://code.google.com/p/protobuf/">http://code.google.com/p/protobuf/</a>

**Author(s)**

Romain Francois <francoisromain@free.fr> and Dirk Eddelbuettel <edd@debian.org>

Maintainer: Romain and Dirk <rprotobuf-yada@lists.r-forge.r-project.org>

**References**

<http://code.google.com/p/protobuf/>

The project is maintained in r-forge: <http://r-forge.r-project.org/projects/rprotobuf/>  
and has a mailing list: <https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/rprotobuf-yada>

**See Also**

[Message](#) for some examples

**Examples**

```
## Not run:
# an example proto file
system.file( "proto", "AddressBook.proto", package = "RProtoBuf" )

# create a message of type AddressBook, defined in the example proto file
demo( "addressbook", package = "RProtoBuf" )

# using R binary connections and files to read and write messages
demo( "io", package = "RProtoBuf" )

# more documentation in the vignette
vignette( "RProtoBuf", package = "RProtoBuf" )

## End(Not run)
```

---

add-methods

*add elements of a repeated field of a message*

---

**Description**

add elements to a repeated field of a message.

**Methods**

signature(object = "Message") add elements to a repeated field of a message

---

ArrayInputStream-class  
Class "ArrayInputStream"

---

**Description**

A [ZeroCopyInputStream](#) backed by an in-memory array of bytes

**Objects from the Class**

Objects can be created by the [ArrayInputStream](#) function

**Slots**

pointer: External pointer to the `google::protobuf::io::ArrayInputStream` C++ object

**Extends**

Class "[ZeroCopyInputStream](#)", directly.

**Methods**

See [ZeroCopyInputStream](#)

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The ArrayInputStream class from the protobuf C++ library. [http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero\\_copy\\_stream\\_impl\\_lite.html#ArrayInputStream](http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero_copy_stream_impl_lite.html#ArrayInputStream)

**See Also**

[ZeroCopyInputStream](#) for methods

---

ArrayInputStream-methods

*Creates an ArrayInputStream*

---

### Description

Constructor for [ArrayInputStream](#) objects

### Methods

signature(payload = "raw", block\_size = "missing" ) Creates a [ArrayInputStream](#) using the raw vector as the payload of the stream

signature(payload = "raw", block\_size = "integer" ) Creates a [ArrayInputStream](#) ... same with block size.

signature(payload = "raw", block\_size = "numeric" ) Creates a [ArrayInputStream](#) ... same with block size.

---

ArrayOutputStream-class

*Class "ArrayOutputStream"*

---

### Description

A [ZeroCopyOutputStream](#) backed by an in-memory array of bytes

### Objects from the Class

Objects can be created by the [ArrayOutputStream](#) function

### Slots

pointer: External pointer to the google::protobuf::io::ArrayOutputStream C++ object

### Extends

Class "[ZeroCopyOutputStream](#)", directly.

### Methods

See [ZeroCopyOutputStream](#)

### Author(s)

Romain Francois <francoisromain@free.fr>

**References**

The ArrayOutputStream class from the protobuf C++ library. [http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero\\_copy\\_stream\\_impl\\_lite.html#ArrayOutputStream](http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero_copy_stream_impl_lite.html#ArrayOutputStream)

**See Also**

[ZeroCopyOutputStream](#) for methods

---

ArrayOutputStream-methods

*Creates an ArrayOutputStream*

---

**Description**

Constructor for [ArrayOutputStream](#) objects

**Methods**

signature(size = "integer", block\_size = "missing" ) Creates a [ArrayOutputStream](#) using of the given size

signature(size = "integer", block\_size = "integer" ) Creates a [ArrayOutputStream](#) ... same with block size.

signature(size = "integer", block\_size = "numeric" ) Creates a [ArrayOutputStream](#) ... same with block size.

signature(size = "numeric", block\_size = "missing" ) Creates a [ArrayOutputStream](#) using of the given size

signature(size = "numeric", block\_size = "integer" ) Creates a [ArrayOutputStream](#) ... same with block size.

signature(size = "numeric", block\_size = "numeric" ) Creates a [ArrayOutputStream](#) ... same with block size.

---

as.list.Message

*Grab the protocol buffer message as an R list*

---

**Description**

Utility to grab the protocol buffer message as an R list, with one item per field.

**Usage**

```
## S3 method for class 'Message':
as.list(x, ...)
## S3 method for class 'Descriptor':
as.list(x, ...)
## S3 method for class 'EnumDescriptor':
as.list(x, ...)
## S3 method for class 'FileDescriptor':
as.list(x, ...)
## S3 method for class 'ServiceDescriptor':
as.list(x, ...)
```

**Arguments**

x	A protocol buffer message, instance of <a href="#">Message</a> , or a protocol message descriptor, instance of <a href="#">Descriptor</a>
...	ignored

**Value**

For messages, a list of the content of the fields is returned.

For message type descriptors, a list containing nested type descriptors ([Descriptor](#) objects), enum type descriptors ([EnumDescriptor](#) objects), or field descriptors ([FieldDescriptor](#) objects)

For enum descriptors, ...

For file descriptors, ...

For service descriptors, ...

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )
romain <- new( Person, email = "francoisromain@free.fr", id = 1 )
as.list( romain )
as.list( Person )
as.list( Person$PhoneType)

## End(Not run)
```

---

asMessage	<i>coerce an object to a protobuf message</i>
-----------	---

---

### Description

coerce an object to the [Message](#) class. This is a short-hand to the [as](#) method with the `Class` argument set to "Message"

### Usage

```
asMessage(x, ...)
```

### Arguments

x	object to coerce to a protobuf message
...	Passed to <a href="#">as</a>

### Value

a [Message](#) object

### Author(s)

Romain Francois <francoisromain@free.fr>

### Examples

```
# coerce a message type descriptor to a message
asMessage( tutorial.Person )

# coerce a enum descriptor
asMessage( tutorial.Person.PhoneType )

# coerce a field descriptor
asMessage( tutorial.Person$email )

# coerce a file descriptor
asMessage( fileDescriptor( tutorial.Person ) )
```

---

BackUp-methods      *Backs up a number of bytes from a stream*

---

**Description**

Backs up a number of bytes from a stream

**See Also**

[ZeroCopyInputStream](#) implements BackUp.

---

ByteCount-methods      *The number of bytes read/written since the object was created*

---

**Description**

The number of bytes read/written since the object was created

**See Also**

[ZeroCopyInputStream](#) implements ByteCount.

---

bytesize-methods      *The number of bytes taken by a message*

---

**Description**

The number of bytes taken by a [Message](#)

**Methods**

`signature(object = "Message")` The number of bytes the message would take when serialized

**Examples**

```
message <- new( tutorial.Person, name = "dddd", email = "eeeeeee", id = 1 )
bytesize( message )
```

---

clear-methods	<i>Clear a field or all fields of the message and set them to their default values</i>
---------------	--

---

### Description

Clear one field or all fields of the message and set them to their default values

### Methods

`signature(object = "Message", field = "missing")` Clear all fields of the message and set them to their default values

`signature(object = "Message", field = "character")` Clear the field identified by its name

`signature(object = "Message", field = "integer")` Clear the field identified by its tag number

`signature(object = "Message", field = "numeric")` Clear the field identified by its tag number

`signature(object = "Message", field = "raw")` Clear the field identified by its tag number

### Examples

```
message <- new( tutorial.Person, name = "dddd", email = "eeeeeee", id = 1 )
writeLines( as.character( message ) )
clear( message )
# clear works also as a pseudo method :
message$clear()

writeLines( as.character( message ) )

# clear single fields
message <- new( tutorial.Person, name = "dddd", email = "eeeeeee", id = 1 )
message$clear( "name" )
writeLines( as.character( message ) )
```

---

clone-methods	<i>Clone protocolo buffer messages</i>
---------------	--

---

### Description

Generic "clone" function and associated method for [Message](#) objects

**Methods**

```
signature(object = "Message") clone the message
```

**Examples**

```
# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )

# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )

# creating a prototype message from the descriptor
sheep <- new( Person, email = "francoisromain@free.fr", id = 2 )

# cloning the sheep
newsheep <- clone( sheep )

# clone and update at once
newsheep <- clone( sheep, id = 3 )

# this can also be used as a pseudo method
sheep$clone()
sheep$clone( id = 3 )
```

---

 completion

---

*Completion support for protocol buffer messages and descriptors*


---

**Description**

These functions support completion of protocol buffer messages and descriptors.

**Usage**

```
## S3 method for class 'Message':
.DollarNames(x, pattern = "")
## S3 method for class 'Descriptor':
.DollarNames(x, pattern = "")
## S3 method for class 'EnumDescriptor':
.DollarNames(x, pattern = "")
## S3 method for class 'FieldDescriptor':
.DollarNames(x, pattern = "")
## S3 method for class 'FileDescriptor':
.DollarNames(x, pattern = "")
## S3 method for class 'ServiceDescriptor':
.DollarNames(x, pattern = "")
## S3 method for class 'MethodDescriptor':
.DollarNames(x, pattern = "")
```

```
## S3 method for class 'ZeroCopyInputStream':
.DollarNames(x, pattern = "")
## S3 method for class 'ZeroCopyOutputStream':
.DollarNames(x, pattern = "")
```

### Arguments

x                    message ([Message](#)) or descriptor ([Descriptor](#))  
 pattern            filter

### Value

Character vector containing potential completions.

For [Message](#) objects, completions are the fields of the message and a set of pseudo methods ("has")

For [EnumDescriptor](#) objects, completions are the names of the possible constants

For [Descriptor](#) objects, completions are the names of the fields, enum types and nested message types defined in the associated message type.

For [FileDescriptor](#) objects, completions are the names of the top-level descriptors (message, enum or service) contained in the associated file, or pseudo methods.

### Author(s)

Romain Francois <francoisromain@free.fr>

### Examples

```
# creating a prototype message from the descriptor
p <- new( tutorial.Person )

.DollarNames( p )
.DollarNames( tutorial.Person )
# but this is usually used with the <TAB> expansion on the command line
# <TAB> means "press the TAB key"
# p$<TAB>
# Person$<TAB>
```

---

```
ConnectionInputStream-class
      Class "ConnectionInputStream"
```

---

### Description

A [ZeroCopyInputStream](#) reading from a binary R connection

### Objects from the Class

Objects can be created by the [ConnectionInputStream](#) function

**Slots**

`pointer`: External pointer to the `rprotobuf::ConnectionInputStream` C++ object

**Extends**

Class "[ZeroCopyInputStream](#)", directly.

**Methods**

See [ZeroCopyInputStream](#)

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The internal C++ class `ConnectionInputStream`

**See Also**

[ZeroCopyInputStream](#) for methods

---

ConnectionInputStream-methods

*Creates an ConnectionInputStream*

---

**Description**

Constructor for [ConnectionInputStream](#) objects

**Methods**

`signature(object="connection")` Creates a [ConnectionInputStream](#) reading from the given R binary connection.

---

ConnectionOutputStream-class  
Class "ConnectionOutputStream"

---

**Description**

A [ZeroCopyOutputStream](#) writing to a binary R connection

**Objects from the Class**

Objects can be created by the [ConnectionOutputStream](#) function

**Slots**

pointer: External pointer to the `rprotobuf::ConnectionOutputStream` C++ object

**Extends**

Class "[ZeroCopyOutputStream](#)", directly.

**Methods**

See [ZeroCopyOutputStream](#)

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The internal C++ class `ConnectionOutputStream`

**See Also**

[ZeroCopyOutputStream](#) for methods

---

ConnectionOutputStream-methods  
*Creates an ConnectionOutputStream*

---

**Description**

Constructor for [ConnectionOutputStream](#) objects

**Methods**

`signature(object="connection")` Creates a [ConnectionOutputStream](#) writing to the given R binary connection.

---

containing\_type-methods

*Gets the message type descriptor that contains a descriptor*

---

### Description

Gets a [Descriptor](#) describing the message type that contains the descriptor.

### See Also

The method is implemented for these classes : [Descriptor](#), [EnumDescriptor](#), [FieldDescriptor](#)

---

Descriptor-class    *Class "Descriptor"*

---

### Description

full descriptive information about a protocol buffer message type. This is a thin wrapper around the C++ class `Descriptor`

### Objects from the Class

Objects are usually created by calls to the [P](#) function.

### Slots

`pointer`: external pointer holding a `Descriptor` object

`type`: full name of the corresponding message type

### Methods

**as.character** signature(`x = "Descriptor"`): returns the debug string of the descriptor.  
This is retrieved by a call to the `DebugString` method of the `Descriptor` object.

**toString** signature(`x = "Descriptor"`): same as `as.character`

**\$** signature(`x = "Descriptor"`): retrieves a descriptor for a member of the message type. This can either be another "Descriptor" instance describing a nested type, or a [EnumDescriptor](#) object describing an enum type, or a [FieldDescriptor](#) object describing a field of the message

**new** signature(`Class = "Descriptor"`): creates a prototype message ([Message](#)) of this descriptor

**show** signature(`object = "Descriptor"`): simple information

**containing\_type** signature(`object = "Descriptor"`): returns a descriptor of the message type that contains this message descriptor, or NULL if this is a top-level message type.

- field\_count** signature(object = "Descriptor") : The number of fields of this message type.
- nested\_type\_count** signature(object = "Descriptor") : The number of nested types of this message type.
- enum\_type\_count** signature(object = "Descriptor") : The number of enum types of this message type.
- field** signature(object = "Descriptor") : extract a field descriptor from a descriptor. Exactly one argument of index, number or name has to be used. If index is used, the field descriptor is retrieved by position, using the `field` method of the `google::protobuf::Descriptor` C++ class. If number is used, the field descriptor is retrieved using the tag number, with the `FindFieldByNumber` C++ method. If name is used, the field descriptor is retrieved by name using the `FindFieldByName`
- nested\_type** signature(object = "Descriptor") : extracts a message type descriptor that is nested in this descriptor. Exactly one argument of index or name has to be used. If index is used, the nested type will be retrieved using its position with the `nested_type` method of the `google::protobuf::Descriptor` C++ class. If name is used, the nested type will be retrieved using its name, with the `FindNestedTypeByName` C++ method
- enum\_type** signature(object = "Descriptor") : extracts an enum type descriptor that is contained in this descriptor. Exactly one argument of index or name has to be used. If index is used, the enum type will be retrieved using its position with the `enum_type` method of the `google::protobuf::Descriptor` C++ class. If name is used, the enum type will be retrieved using its name, with the `FindEnumTypeByName` C++ method

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The Descriptor c++ class. <http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.descriptor.html#Descriptor>

**See Also**

the `P` function creates "Descriptor" messages.

**Examples**

```
# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )

# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )

# enum type
Person$PhoneType
```

```

# nested type
Person$PhoneNumber

# field
Person$email

# use this descriptor to create a message
new( Person )

```

---

descriptor-methods *Get the descriptor of a message*

---

### Description

Get the [Descriptor](#) associated with a [Message](#)

### Methods

signature(object = "Message") Get the descriptor of the message, as a [Descriptor](#) instance

---

```
EnumDescriptor-class
```

*Class "EnumDescriptor"*

---

### Description

R representation of an enum descriptor. This is a thin wrapper around the `EnumDescriptor` C++ class.

### Objects from the Class

Objects of this class are typically retrieved as members of [Descriptor](#) objects

### Slots

**pointer:** external pointer to the `EnumDescriptor` instance  
**name:** simple name of the enum  
**full\_name:** fully qualified name  
**type:** fully qualified name of the type that contains this enumeration

**Methods**

**show** signature(object = "EnumDescriptor"): small information

**as.character** signature(x = "EnumDescriptor"): returns the debug string of the enum descriptor. This is retrieved by a call to the `DebugString` method of the `EnumDescriptor` object.

**toString** signature(x = "EnumDescriptor"): same as `as.character`

**\$** signature(x = "EnumDescriptor"): get the number associated with the name

**containing\_type** signature(object = "EnumDescriptor"): returns a [Descriptor](#) of the message type that contains this enum descriptor, or NULL if this is a top level enum descriptor.

**length** signature(x = "EnumDescriptor"): number of constants in this enum.

**value\_count** signature(object = "EnumDescriptor"): number of constants in this enum.

**value** signature(object = "EnumDescriptor"): extracts an [EnumValueDescriptor](#). Exactly one argument of `index`, `number` or `name` has to be used. If `index` is used, the enum value descriptor is retrieved by position, using the `value` method of the C++ class. If `number` is used, the enum value descriptor is retrieved using the value of the constant, using the `FindValueByNumber` C++ method. If `name` is used, the enum value descriptor is retrieved using the name of the constant, using the `FindValueByName` C++ method.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The `EnumDescriptor` C++ class

**See Also**

The [Descriptor](#) class

**Examples**

```
# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )

# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )

# enum type
Person$PhoneType
```

---

```
EnumValueDescriptor-class
    Class "EnumValueDescriptor"
```

---

### Description

R representation of an enum value descriptor. This is a thin wrapper around the `EnumValueDescriptor` C++ class.

### Objects from the Class

Objects of this class are typically retrieved with the `value` method of the `EnumDescriptor` class

### Slots

`pointer`: external pointer to the `EnumValueDescriptor` instance

### Methods

**show** signature (object = "EnumValueDescriptor"): small information  
**as.character** signature (x = "EnumValueDescriptor"): returns the debug string of the enum descriptor. This is retrieved by a call to the `DebugString` method of the `EnumDescriptor` object.  
**toString** signature (x = "EnumValueDescriptor"): same as `as.character`  
**\$** signature (x = "EnumValueDescriptor"): invoke pseudo methods  
**enum\_type** signature (object = "EnumDescriptor") : retrieves the `EnumDescriptor` related to this value descriptor.

### Author(s)

Romain Francois <francoisromain@free.fr>

### References

The `EnumValueDescriptor` C++ class. <http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.descriptor.html#EnumValueDescriptor>

---

```
enum_type-methods  Extract an enum type descriptor for a nested type
```

---

### Description

Extract a `EnumDescriptor` contained in a `Descriptor`

### See Also

The method is implemented for the `Descriptor` class

---

enum\_type\_count-methods

*The number of enum types*

---

### **Description**

The number of enum types

### **See Also**

The method is implemented for the [Descriptor](#) class

---

fetch-methods

*Fetch content of a repeated field*

---

### **Description**

Fetch content of a repeated field of a message

### **Methods**

signature(object = "Message") Fetch content of a message repeated field

---

field-methods

*Extract a field descriptor*

---

### **Description**

Extract a [FieldDescriptor](#) from a [Descriptor](#)

### **See Also**

The method is implemented for the [Descriptor](#) class

---

```
FieldDescriptor-class
      Class "FieldDescriptor"
```

---

### Description

R representation of message type field descriptor. This is a thin wrapper around the C++ class `FieldDescriptor`

### Objects from the Class

Objects typically are retrieved from [FieldDescriptor](#)

### Slots

**pointer:** external pointer to the `FieldDescriptor` c++ object

**name:** name of the field within the message type

**full\_name:** Fully qualified name of the field

**type:** Fully qualified name of the type that contains this field

### Methods

**show** signature(object = "FieldDescriptor"): small description

**as.character** signature(x = "FieldDescriptor"): returns the debug string of the field descriptor. This is retrieved by a call to the `DebugString` method of the `FieldDescriptor` object.

**toString** signature(x = "FieldDescriptor"): same as `as.character`

**\$** signature(x = "FieldDescriptor"): used to invoke pseudo methods

**containing\_type** signature(object = "FieldDescriptor"): returns a [Descriptor](#) of the message type that contains this field descriptor.

**is\_extension** signature(object = "FieldDescriptor"): indicates if this is an extension.

**number** signature(object = "FieldDescriptor"): gets the declared tag number of this field.

**type** signature(object = "FieldDescriptor"): type of this field.

**cpp\_type** signature(object = "FieldDescriptor"): c++ type of this field.

**label** signature(object = "FieldDescriptor"): label of this field.

**is\_required** signature(object = "FieldDescriptor"): is this field required.

**is\_optional** signature(object = "FieldDescriptor"): is this field optional.

**is\_repeated** signature(object = "FieldDescriptor"): is this field repeated.

**has\_default\_value** signature(object = "FieldDescriptor"): indicates if this field has a default value.

**default\_value** signature(object = "FieldDescriptor") : the default value of this field.

**message\_type** signature(object = "FieldDescriptor") : the [Descriptor](#) for the associated message type. Generates an error if this field is not a message type field.

**enum\_type** signature(object = "FieldDescriptor") : the [EnumDescriptor](#) for the associated enum type. Generates an error if this field is not an enum type field

### Author(s)

Romain Francois <francoisromain@free.fr>

### References

The `FieldDescriptor` C++ class

### See Also

[Descriptor](#)

### Examples

```
# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )

# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )

# field descriptor object
Person$email

# debug string
as.character( Person$email )

# or as a pseudo method
Person$email$as.character()
```

---

field\_count-methods

*The number of fields*

---

### Description

The number of fields

### See Also

The method is implemented for the [Descriptor](#) class

---

FileDescriptor-class  
*Class "FileDescriptor"*

---

### Description

Class "FileDescriptor"

### Objects from the Class

Objects are usually created using the `fileDescriptor` method

### Slots

`pointer`: external pointer to a `google::protobuf::FileDescriptor` C++ object

### Methods

`$ signature(x = "FileDescriptor")`: used to invoke a pseudo method of the file descriptor or get a top level message, enum or service descriptor

`toString signature(x = "FileDescriptor" )`: gets the debug string

`as.character signature(x = "FileDescriptor" )`: gets the debug string

`show signature(x = "FileDescriptor" )`: prints small text

`name signature(object = "FileDescriptor" )`: name of the file

### Author(s)

Romain Francois <francoisromain@free.fr>

### References

The <http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.descriptor.html#FileDescriptor>

---

fileDescriptor-methods  
*gets the file descriptor of an object*

---

### Description

Gets the file descriptor of an object

**Methods**

signature(object = "Descriptor") retrieves the file descriptor associated with this descriptor

signature(object = "Message") retrieves the file descriptor associated with the descriptor of this message

signature(object = "EnumDescriptor") retrieves the file descriptor associated with the enum descriptor

signature(object = "FieldDescriptor") retrieves the file descriptor associated with the field descriptor

signature(object = "ServiceDescriptor") retrieves the file descriptor associated with the service descriptor

signature(object = "MethodDescriptor") retrieves the file descriptor associated with the method descriptor

---

FileInputStream-class

*Class "FileInputStream"*

---

**Description**

A [ZeroCopyInputStream](#) reading from a file

**Objects from the Class**

Objects can be created by the [FileInputStream](#) function

**Slots**

**pointer:** External pointer to the `google::protobuf::io::FileInputStream` C++ object

**Extends**

Class "[ZeroCopyInputStream](#)", directly.

**Methods**

**close** signature (con="FileInputStream"): Flushes any buffers and closes the underlying file. Returns false if an error occurs during the process; use `GetErrno` to examine the error

**GetErrno** signature (object="FileInputStream"): If an I/O error has occurred on this file descriptor, this is the `errno` from that error. Otherwise, this is zero. Once an error occurs, the stream is broken and all subsequent operations will fail.

**SetCloseOnDelete** signature (object="FileInputStream"): set the close on delete behavior.

See [ZeroCopyInputStream](#) for inherited methods

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The `FileInputStream` class from the protobuf C++ library. [http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero\\_copy\\_stream\\_impl\\_lite.html#FileInputStream](http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero_copy_stream_impl_lite.html#FileInputStream)

**See Also**

[ZeroCopyInputStream](#) for methods

---

`FileInputStream-methods`

*Creates an `FileInputStream`*

---

**Description**

Constructor for [FileInputStream](#) objects

**Methods**

`signature(filename = "character", block_size = "logical", close.on.delete = "logical")`  
 Creates a [FileInputStream](#) reading from the given file.

---

`FileOutputStream-class`

*Class "`FileOutputStream`"*

---

**Description**

A [ZeroCopyOutputStream](#) reading from a file

**Objects from the Class**

Objects can be created by the [FileOutputStream](#) function

**Slots**

`pointer`: External pointer to the `google::protobuf::io::FileOutputStream` C++ object

**Extends**

Class "[ZeroCopyOutputStream](#)", directly.

## Methods

**close** signature (con="FileOutputStream"): Flushes any buffers and closes the underlying file. Returns false if an error occurs during the process; use `GetErrno` to examine the error

**flush** signature (con="FileOutputStream"): Flushes `FileOutputStream`'s buffers but does not close the underlying file

**GetErrno** signature (object="FileInputStream"): If an I/O error has occurred on this file descriptor, this is the `errno` from that error. Otherwise, this is zero. Once an error occurs, the stream is broken and all subsequent operations will fail.

**SetCloseOnDelete** signature (object="FileOutputStream"): set the close on delete behavior.

See [ZeroCopyOutputStream](#) for inherited methods

## Author(s)

Romain Francois <francoisromain@free.fr>

## References

The `FileOutputStream` class from the `protobuf C++` library. [http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero\\_copy\\_stream\\_impl\\_lite.html#FileOutputStream](http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero_copy_stream_impl_lite.html#FileOutputStream)

## See Also

[ZeroCopyOutputStream](#) for methods

---

FileOutputStream-methods

*Creates an `FileOutputStream`*

---

## Description

Constructor for [FileOutputStream](#) objects

## Methods

signature(filename = "character", block\_size = "logical", close.on.delete = "logical")  
Creates a [FileOutputStream](#) writing to the given file.

---

GetErrno-methods     *Get the error number for an I/O error*

---

### Description

If an I/O error has occurred on this file descriptor, this is the errno from that error

### Methods

See classes [FileInputStream](#) and [FileOutputStream](#) for implementations.

---

has-methods     *Indicates if a message has the given field set*

---

### Description

This generic method, currently only implemented for [Message](#) indicates if the message has the given field set.

For non repeated fields, a call to the `HasField` method of the corresponding `Message` is issued.

For repeated fields, a call to the `FieldSize` method is issued, and the message is declared to have the field if the size is greater than 0.

### Methods

**has** signature(object = "Message"): Indicates if the message has a given field.

---

invoke-methods     *invoke a protobuf rpc method*

---

### Description

invoke a protobuf rpc method

### Methods

signature(method = "MethodDescriptor", message = "Message") invoke a protobuf rpc method locally.

signature(method = "MethodDescriptor", message = "Message", protocol = "RpcHTTP" )  
invoke a protobuf rpc method over http.

---

`isInitialized-methods`*Indicates if a protocol buffer message is initialized*

---

**Description**

Indicates if a [Message](#) is initialized. A message is initialized if all its required fields are set.

**Methods**

`signature(object = "Message")` is the message initialized

**Examples**

```
message <- new( tutorial.Person, name = "" )
isInitialized( message ) # FALSE (id is not set)
message$isInitialized() # FALSE

message <- new( tutorial.Person, name = "", id = 2 )
isInitialized( message ) # TRUE
message$isInitialized() # TRUE
```

---

`is_extension-methods`*Indicates if a field descriptor is an extension*

---

**Description**

Indicates if a field descriptor is an extension

**See Also**

The method is implemented for the [FieldDescriptor](#) class

---

`label-methods`*Gets the label of a field*

---

**Description**

Gets the label of a field

**See Also**

The method is implemented for the [FieldDescriptor](#) class

---

merge-methods	<i>Merge two messages of the same type</i>
---------------	--

---

**Description**

Merge two [Message](#) objects of the same type.

**Methods**

signature(x = "Message", y = "Message") merge two messages of the same type

**Errors**

An error of class "IncompatibleType" is thrown if the two messages are not of the same message type.

**Examples**

```
m1 <- new( tutorial.Person, email = "francoisromain@free.fr" )
m2 <- new( tutorial.Person, id = 5 )
m3 <- merge( m1, m2 )
writeLines( as.character( m1 ) )
writeLines( as.character( m2 ) )
writeLines( as.character( m3 ) )
```

---

Message-class	<i>Class "Message"</i>
---------------	------------------------

---

**Description**

R representation of protocol buffer messages. This is a thin wrapper around the `Message` c++ class that holds the actual message as an external pointer.

**Objects from the Class**

Objects are typically created by the `new` function invoked on a [Descriptor](#) object.

**Slots**

pointer: external pointer to the c++ `Message` object

type: fully qualified name of the message type

**Methods**

**as.character** signature (x = "Message"): returns the debug string of the message. This is built from a call to the `DebugString` message of the `Message` object

**toString** signature (x = "Message"): same as `as.character`

**\$<-** signature (x = "Message"): set the value of a field of the message.

**\$** signature (x = "Message"): gets the value of a field. Primitive types are brought back to R as R objects of the closest matching R type. Messages are brought back as instances of the `Message` class.

**[[** signature (x = "Message"): extracts a field identified by its name or declared tag number

**[[<-** signature (x = "Message"): replace the value of a field identified by its name or declared tag number

**serialize** signature (object = "Message"): serialize a message. If the "connection" argument is `NULL`, the payload of the message is returned as a raw vector, if the "connection" argument is a binary writable connection, the payload is written into the connection. If "connection" is a character vector, the message is sent to the file (in binary format).

**show** signature (object = "Message"): displays a short text about the message

**update** signature (object = "Message"): set several fields of the message at once

**length** signature (x = "Message"): The number of fields actually contained in the message. A field counts in these two situations: the field is repeated and the field size is greater than 0, the field is not repeated and the message has the field.

**str** signature (object = "Message"): displays the structure of the message

**identical** signature (x = "Message", y = "Message"): Test if two messages are exactly identical

**==** signature (e1 = "Message", e2 = "Message"): Same as `identical`

**!=** signature (e1 = "Message", e2 = "Message"): Negation of `identical`

**all.equal** signature (e1 = "Message", e2 = "Message"): Test near equality

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The `Message` class from the C++ proto library. <http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.message.html>

**See Also**

`P` creates objects of class `Descriptor` that can be used to create messages.

**Examples**

```

# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )

# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )
PhoneNumber <- P( "tutorial.Person.PhoneNumber" )

# creating a prototype message from the descriptor
p <- new( Person )
p$email # not set, returns default value
p$id    # not set, returns default value
as.character( p ) # empty
has( p, "email" ) # is the "email" field set
has( p, "phone" ) # is the "email" field set
length( p )      # number of fields actually set

# update several fields at once
romain <- update( new( Person ),
  email = "francoisromain@free.fr",
  id = 1,
  name = "Romain Francois",
  phone = new( PhoneNumber , number = "+33(0)...", type = "MOBILE" )
)

# supply parameters to the constructor
dirk <- new( Person,
  email = "edd@debian.org",
  id = 2,
  name = "Dirk Eddelbuettel" )
# update the phone repeated field with a list of PhoneNumber messages
dirk$phone <- list(
  new( PhoneNumber , number = "+01...", type = "MOBILE" ),
  new( PhoneNumber , number = "+01...", type = "HOME" ) )

# with/within style
saptarshi <- within( new(Person), {
  id <- 3
  name <- "Saptarshi Guha"
  email <- "saptarshi.guha@gmail.com"
} )

# make an addressbook
book <- new( tutorial.AddressBook, person = list( romain, dirk, saptarshi ) )

# serialize the message to a file
tf <- tempfile( )
serialize( book, tf )

# the payload of the message
serialize( book, NULL )

```

```
# read the file into a new message
m <- tutorial.AddressBook$read( tf )
writeLines( as.character( m ) )
sapply( m$person, function(p) p$name )
```

---

```
MethodDescriptor-class
      Class "MethodDescriptor"
```

---

### Description

R representation of Service Descriptors

### Objects from the Class

TODO

### Slots

**pointer:** External pointer to a `google::protobuf::MethodDescriptor` C++ object  
**name:** fully qualified name of the method  
**service:** fully qualified name of the service that defines this method

### Methods

**as.character** signature(x = "MethodDescriptor"): debug string of the method  
**toString** signature(x = "MethodDescriptor"): debug string of the method  
**\$** signature(x = "MethodDescriptor"): ...  
**\$<-** signature(x = "MethodDescriptor"): ...  
**input\_type** signature(object = "MethodDescriptor"): the [Descriptor](#) of the input type of the method  
**output\_type** signature(object = "MethodDescriptor"): the [Descriptor](#) of the output type of the method

### Author(s)

Romain Francois <francoisromain@free.fr>

---

name	<i>Name or full name of a descriptor</i>
------	--

---

**Description**

name or full name of a descriptor

**Methods**

```
signature(object = "Descriptor") ...
signature(object = "FieldDescriptor") ...
signature(object = "EnumDescriptor") ...
signature(object = "ServiceDescriptor") ...
signature(object = "MethodDescriptor") ...
```

---

nested\_type-methods

*Extract a message type descriptor for a nested type*

---

**Description**

Extract a [Descriptor](#) nested in another [Descriptor](#)

**See Also**

The method is implemented for the [Descriptor](#) class

---

nested\_type\_count-methods

*The number of fields*

---

**Description**

The number of fields

**See Also**

The method is implemented for the [Descriptor](#) class

---

Next-methods      *Obtains a chunk of data from the stream*

---

**Description**

Obtains a chunk of data from the stream

**See Also**

[ZeroCopyInputStream](#) implements Next.

---

number-methods      *Gets the declared tag number of a field*

---

**Description**

Gets the declared tag number of a field

**See Also**

The method is implemented for the [FieldDescriptor](#) class

---

P      *Protocol Buffer descriptor importer*

---

**Description**

The P function searches for a protocol message descriptor in the descriptor pool.

**Usage**

P(type, file)

**Arguments**

type	Fully qualified type name of the protocol buffer
file	optional proto file. If given, the definition contained in the file is first registered with the pool of message descriptors

**Value**

An object of class [Descriptor](#). An error is generated otherwise.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )
as.character( Person )
```

---

read-methods

*Read a protocol buffer message from a connection*

---

**Description**

Read a [Message](#) from a connection using its associated [Descriptor](#)

**Methods**

signature(descriptor = "Descriptor", input = "character") Read the message from a file

signature(descriptor = "Descriptor") Read from a binary connection.

signature(descriptor = "Descriptor", input = "raw") Read the message from a raw vector

**Examples**

```
# example file that contains a "tutorial.AddressBook" message
book <- system.file( "examples", "addressbook.pb", package = "RProtoBuf" )

# read the message
message <- read( tutorial.AddressBook, book )

# or using the pseudo method
message <- tutorial.AddressBook$read( book )

# write its debug string
writeLines( as.character( message ) )

# grab the name of each person
sapply( message$person, function(p) p$name )

# read from a binary file connection
f <- file( book, open = "rb" )
message2 <- read( tutorial.AddressBook, f )
close( f )

# read from a message payload (raw vector)
payload <- readBin( book, raw(0), 5000 )
```

```
message3 <- tutorial.AddressBook$read( payload )
```

---

readASCII-methods    *read a message in ASCII format*

---

### Description

Method to read a Message in ASCII format

### Methods

signature(descriptor = "Descriptor", input = "ANY") Read the message from a connection (file, etc ...)

signature(descriptor = "Descriptor", input = "character") Read the message directly from the character string

---

readProtoFiles        *protocol buffer descriptor importer*

---

### Description

Imports proto files into the descriptor pool that is then used by the P function to resolve message type names.

### Usage

```
readProtoFiles(files, dir, package = "RProtoBuf")
```

### Arguments

files	Proto files
dir	Directory. If files is not specified, files with the "proto" extension in the dir directory are imported
package	R package name. If files and dir are missing, "proto" files in the "proto" directory of the package tree are imported.

### Value

NULL, invisibly.

### Author(s)

Romain Francois <francoisromain@free.fr>

**See Also**

[P](#)

**Examples**

```
# from a package
readProtoFiles( package = "RProtoBuf" )

# from a directory
proto.dir <- system.file( "proto", package = "RProtoBuf" )
readProtoFiles( dir = proto.dir )

# set of files
proto.files <- list.files( proto.dir, full.names = TRUE )
readProtoFiles( proto.files )
```

---

RpcHTTP-class

*Class "RpcHTTP"*

---

**Description**

Support for protobuf rpc over HTTP

**Objects from the Class**

Objects can be created by calls of the form `new("RpcHTTP", host = "somehost", port = port.number, root = "" )`

**Slots**

`host`: Host name  
`port`: port number  
`root`: root directory of the protobuf http server

**Author(s)**

Romain Francois <francoisromain@free.fr>

**See Also**

[invoke](#) uses objects of this class to perform a method invocation over http.

---

```
ServiceDescriptor-class
    Class "ServiceDescriptor"
```

---

**Description**

R representation of Service Descriptors

**Objects from the Class**

TODO

**Slots**

**pointer:** External pointer to a `google::protobuf::ServiceDescriptor` C++ object  
**name:** fully qualified name of the service

**Methods**

**as.character** signature (x = "ServiceDescriptor"): debug string of the service  
**toString** signature (x = "ServiceDescriptor"): debug string of the service  
**show** signature (x = "ServiceDescriptor"): ...  
**\$** signature (x = "ServiceDescriptor"): invoke pseudo methods or retrieve method descriptors contained in this service descriptor.  
**[[** signature (x = "ServiceDescriptor"): extracts methods descriptors contained in this service descriptor  
**length** signature (x = "ServiceDescriptor"): number of [MethodDescriptor](#)  
**method\_count** signature (x = "ServiceDescriptor"): number of [MethodDescriptor](#)  
**method** signature (x = "ServiceDescriptor"): retrieves a [MethodDescriptor](#)

**Author(s)**

Romain Francois <francoisromain@free.fr>

---

```
set-methods          set a subset of values of a repeated field of a message
```

---

**Description**

set a subset of values of a repeated field of a message

**Methods**

signature(object = "Message") set a subset of values of a repeated field of a message

---

SetCloseOnDelete-methods  
*set the close on delete behavior*

---

**Description**

By default, the file descriptor is not closed when a stream is destroyed, use `SetCloseOnDelete ( stream, TRUE )` to change that.

**Methods**

See classes [FileInputStream](#) and [FileOutputStream](#) for implementations.

---

size-methods      *Size of a message field*

---

**Description**

The number of object currently in a given field of a protocol buffer message.  
For non repeated fields, the size is 1 if the message has the field, 0 otherwise.  
For repeated fields, the size is the number of objects in the array.

**Methods**

`signature(object = "Message")` Number of objects in a message field

---

`sizegets`      *Set the size of a field*

---

**Description**

Sets the size of a repeated field.

**Methods**

`signature(object = "Message")` sets the size of a message field

---

Skip-methods      *Skips a number of bytes*

---

**Description**

Skips a number of bytes

---

swap-methods	<i>swap elements of a repeated field of a message</i>
--------------	---

---

**Description**

swap elements of a repeated field of a message.

**Methods**

signature(object = "Message") swap elements of a repeated field of a message

**References**

See the SwapElements of the Reflection class, part of the protobuf library. <http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.message.html>

---

type-methods	<i>Gets the type or the c++ type of a field</i>
--------------	---

---

**Description**

Gets the type or the c++ type of a field

**See Also**

The method is implemented for the [FieldDescriptor](#) class

---

with.Message	<i>with and within methods for protocol buffer messages</i>
--------------	---

---

**Description**

Convenience wrapper that allow getting and setting fields of protocol buffer messages from within the object

**Usage**

```
## S3 method for class 'Message':  
with(data, expr, ...)  
## S3 method for class 'Message':  
within(data, expr, ...)
```

**Arguments**

<code>data</code>	A protocol buffer message, instance of <a href="#">Message</a>
<code>expr</code>	R expression to evaluate
<code>...</code>	ignored

**Details**

The expression is evaluated in an environment that allows to set and get fields of the message

The fields of the message are mapped to active bindings (see [makeActiveBinding](#)) so that they can be accessed and modified from within the environment.

**Value**

`with` returns the value of the expression and `within` returns the `data` argument.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )
romain <- within( new( Person ), {
  email <- "francoisromain@free.fr"
  id <- 10L
} )

## End(Not run)
```

---

ZeroCopyInputStream-class

*Virtual Class "ZeroCopyInputStream"*

---

**Description**

R wrapper for the ZeroCopyInputStream c++ class

**Objects from the Class**

This is a virtual class

**Slots**

`pointer`: external pointer to the `google::protobuf::io::ZeroCopyInputStream` object

**Methods**

- \$** signature (x="ZeroCopyInputStream"): invokes a method
- Next** signature (object="ZeroCopyInputStream"): Get a number of bytes from the stream as a raw vector.
- Skip** signature (object="ZeroCopyInputStream"): skip a number of bytes
- BackUp** signature (object="ZeroCopyInputStream"): Backs up a number of bytes, so that the next call to Next returns data again that was already returned by the last call to Next.
- ByteCount** signature (object="ZeroCopyInputStream"): Returns the total number of bytes read since this object was created.
- ReadRaw** signature (object="ZeroCopyInputStream", size = "integer"): read raw bytes from the stream
- ReadString** signature (object="ZeroCopyInputStream", size = "integer"): same as ReadRaw but formats the result as a string
- ReadVarint32** signature (object="ZeroCopyInputStream"): Read an unsigned integer with Varint encoding, truncating to 32 bits.
- ReadLittleEndian32** signature (object="ZeroCopyInputStream"): Read a 32-bit little-endian integer.
- ReadLittleEndian64** signature (object="ZeroCopyInputStream"): Read a 64-bit little-endian integer. In R the value is stored as a double which loses some precision (no other way)
- ReadVarint64** signature (object="ZeroCopyInputStream"): Read a 64-bit integer with varint encoding. In R the value is stored as a double which loses some precision (no other way)

**Author(s)**

Romain Francois <francoisromain@free.fr>

**References**

The google::protobuf::io::ZeroCopyInputStream C++ class. [http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero\\_copy\\_stream.html#ZeroCopyInputStream](http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero_copy_stream.html#ZeroCopyInputStream)

**See Also**

TODO: add classes that extend

---

ZeroCopyOutputStream-class  
*Virtual Class "ZeroCopyOutputStream"*

---

### Description

R wrapper for the ZeroCopyOutputStream c++ class

### Objects from the Class

This is a virtual class

### Slots

**pointer:** external pointer to the `google::protobuf::io::ZeroCopyOutputStream` object

### Methods

**\$ signature** (x="ZeroCopyOutputStream"): invokes a method

**Next signature** (object="ZeroCopyOutputStream", payload = "raw" ): push the raw vector into the stream. Returns the number of bytes actually written.

**BackUp signature** (object="ZeroCopyOutputStream"): Backs up a number of bytes, so that the end of the last buffer returned by Next is not actually written.

**ByteCount signature** (object="ZeroCopyOutputStream"): Returns the total number of bytes written since this object was created.

**WriteRaw signature** (object="ZeroCopyOutputStream", payload = "raw": write the raw bytes to the stream

### Author(s)

Romain Francois <francoisromain@free.fr>

### References

The `google::protobuf::io::ZeroCopyOutputStream` C++ class. [http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero\\_copy\\_stream.html#ZeroCopyOutputStream](http://code.google.com/apis/protocolbuffers/docs/reference/cpp/google.protobuf.io.zero_copy_stream.html#ZeroCopyOutputStream)

### See Also

TODO: add classes that extend

# Index

- !=, Message, Message-method  
(Message-class), 28
- \*Topic **classes**
  - ArrayInputStream-class, 3
  - ArrayOutputStream-class, 4
  - ConnectionInputStream-class,  
11
  - ConnectionOutputStream-class,  
13
  - Descriptor-class, 14
  - EnumDescriptor-class, 16
  - EnumValueDescriptor-class, 18
  - FieldDescriptor-class, 20
  - FileDescriptor-class, 22
  - FileInputStream-class, 23
  - FileOutputStream-class, 24
  - Message-class, 28
  - MethodDescriptor-class, 31
  - RpcHTTP-class, 36
  - ServiceDescriptor-class, 37
  - with.Message, 39
  - ZeroCopyInputStream-class, 40
  - ZeroCopyOutputStream-class,  
42
- \*Topic **interface**
  - P, 33
- \*Topic **methods**
  - add-methods, 3
  - ArrayInputStream-methods, 4
  - ArrayOutputStream-methods, 5
  - BackUp-methods, 8
  - ByteCount-methods, 8
  - bytesize-methods, 8
  - clear-methods, 9
  - clone-methods, 9
  - ConnectionInputStream-methods,  
12
  - ConnectionOutputStream-methods,  
13
  - containing\_type-methods, 14
  - descriptor-methods, 16
  - enum\_type-methods, 18
  - enum\_type\_count-methods, 19
  - fetch-methods, 19
  - field-methods, 19
  - field\_count-methods, 21
  - fileDescriptor-methods, 22
  - FileInputStream-methods, 24
  - FileOutputStream-methods, 25
  - GetErrno-methods, 26
  - has-methods, 26
  - invoke-methods, 26
  - is\_extension-methods, 27
  - isInitialized-methods, 27
  - label-methods, 27
  - merge-methods, 28
  - name, 32
  - nested\_type-methods, 32
  - nested\_type\_count-methods, 32
  - Next-methods, 33
  - number-methods, 33
  - read-methods, 34
  - readASCII-methods, 35
  - set-methods, 37
  - SetCloseOnDelete-methods, 38
  - size-methods, 38
  - sizegets, 38
  - Skip-methods, 38
  - swap-methods, 39
  - type-methods, 39
- \*Topic **package**
  - RProtoBuf-package, 2
- \*Topic **programming**
  - as.list.Message, 6
  - asMessage, 7
  - completion, 10
  - readProtoFiles, 35
  - .DollarNames.Descriptor

- (*completion*), 10
- .DollarNames.EnumDescriptor  
(*completion*), 10
- .DollarNames.FieldDescriptor  
(*completion*), 10
- .DollarNames.FileDescriptor  
(*completion*), 10
- .DollarNames.Message  
(*completion*), 10
- .DollarNames.MethodDescriptor  
(*completion*), 10
- .DollarNames.ServiceDescriptor  
(*completion*), 10
- .DollarNames.ZeroCopyInputStream  
(*completion*), 10
- .DollarNames.ZeroCopyOutputStream  
(*completion*), 10
- ==, Message, Message-method  
(*Message-class*), 28
- [[], Message-method  
(*Message-class*), 28
- [[], ServiceDescriptor-method  
(*ServiceDescriptor-class*),  
37
- [<-, Message-method  
(*Message-class*), 28
- \$, Descriptor-method  
(*Descriptor-class*), 14
- \$, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- \$, EnumValueDescriptor-method  
(*EnumValueDescriptor-class*),  
18
- \$, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- \$, FileDescriptor-method  
(*FileDescriptor-class*), 22
- \$, Message-method (*Message-class*),  
28
- \$, MethodDescriptor-method  
(*MethodDescriptor-class*),  
31
- \$, ServiceDescriptor-method  
(*ServiceDescriptor-class*),  
37
- \$, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- \$, ZeroCopyOutputStream-method  
(*ZeroCopyOutputStream-class*),  
42
- \$<-, Descriptor-method  
(*Descriptor-class*), 14
- \$<-, Message-method  
(*Message-class*), 28
- \$<-, MethodDescriptor-method  
(*MethodDescriptor-class*),  
31
- add (*add-methods*), 3
- add, Message-method (*add-methods*),  
3
- add-methods, 3
- all.equal, Message, Message-method  
(*Message-class*), 28
- ArrayInputStream, 3, 4
- ArrayInputStream  
(*ArrayInputStream-methods*),  
4
- ArrayInputStream, raw, integer-method  
(*ArrayInputStream-methods*),  
4
- ArrayInputStream, raw, missing-method  
(*ArrayInputStream-methods*),  
4
- ArrayInputStream, raw, numeric-method  
(*ArrayInputStream-methods*),  
4
- ArrayInputStream-class, 3
- ArrayInputStream-methods, 4
- ArrayOutputStream, 4, 5
- ArrayOutputStream  
(*ArrayOutputStream-methods*),  
5
- ArrayOutputStream, integer, integer-method  
(*ArrayOutputStream-methods*),  
5
- ArrayOutputStream, integer, missing-method  
(*ArrayOutputStream-methods*),  
5
- ArrayOutputStream, integer, numeric-method  
(*ArrayOutputStream-methods*),  
5
- ArrayOutputStream, numeric, integer-method  
(*ArrayOutputStream-methods*),  
5

- ArrayOutputStream, numeric, missing-method  
(*ArrayOutputStream-methods*),  
5
- ArrayOutputStream, numeric, numeric-method  
(*ArrayOutputStream-methods*),  
5
- ArrayOutputStream-class, 4
- ArrayOutputStream-methods, 5
- as, 7
- as.character, Descriptor-method  
(*Descriptor-class*), 14
- as.character, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- as.character, EnumValueDescriptor-method  
(*EnumValueDescriptor-class*),  
18
- as.character, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- as.character, FileDescriptor-method  
(*FileDescriptor-class*), 22
- as.character, Message-method  
(*Message-class*), 28
- as.character, MethodDescriptor-method  
(*MethodDescriptor-class*),  
31
- as.character, ServiceDescriptor-method  
(*ServiceDescriptor-class*),  
37
- as.list.Descriptor  
(*as.list.Message*), 6
- as.list.EnumDescriptor  
(*as.list.Message*), 6
- as.list.FileDescriptor  
(*as.list.Message*), 6
- as.list.Message, 6
- as.list.ServiceDescriptor  
(*as.list.Message*), 6
- asMessage, 7
- BackUp (*BackUp-methods*), 8
- BackUp, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- BackUp, ZeroCopyOutputStream-method  
(*ZeroCopyOutputStream-class*),  
42
- BackUp-methods, 8
- ByteCount (*ByteCount-methods*), 8
- ByteCount, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- ByteCount, ZeroCopyOutputStream-method  
(*ZeroCopyOutputStream-class*),  
42
- ByteCount-methods, 8
- bytesize (*bytesize-methods*), 8
- bytesize, Message-method  
(*bytesize-methods*), 8
- bytesize-methods, 8
- clear (*clear-methods*), 9
- clear, Message, character-method  
(*clear-methods*), 9
- clear, Message, integer-method  
(*clear-methods*), 9
- clear, Message, missing-method  
(*clear-methods*), 9
- clear, Message, numeric-method  
(*clear-methods*), 9
- clear, Message, raw-method  
(*clear-methods*), 9
- clear-methods, 9
- clone (*clone-methods*), 9
- clone, Message-method  
(*clone-methods*), 9
- clone-methods, 9
- close, FileInputStream-method  
(*FileInputStream-class*), 23
- close, FileOutputStream-method  
(*FileOutputStream-class*),  
24
- completion, 10
- ConnectionInputStream, 11, 12
- ConnectionInputStream  
(*ConnectionInputStream-methods*),  
12
- ConnectionInputStream, connection-method  
(*ConnectionInputStream-methods*),  
12
- ConnectionInputStream-class, 11
- ConnectionInputStream-methods, 12
- ConnectionOutputStream, 13
- ConnectionOutputStream  
(*ConnectionOutputStream-methods*),  
13
- ConnectionOutputStream, connection-method  
(*ConnectionOutputStream-methods*),

- 13
- ConnectionOutputStream-class, 13
- ConnectionOutputStream-methods, 13
- containing\_type
  - (*containing\_type-methods*), 14
- containing\_type, Descriptor-method (*Descriptor-class*), 14
- containing\_type, EnumDescriptor-method (*EnumDescriptor-class*), 16
- containing\_type, FieldDescriptor-method (*FieldDescriptor-class*), 20
- containing\_type-methods, 14
- cpp\_type (*type-methods*), 39
- cpp\_type, FieldDescriptor-method (*FieldDescriptor-class*), 20
- cpp\_type-methods (*type-methods*), 39
- CPPTYPE\_BOOL (*type-methods*), 39
- CPPTYPE\_DOUBLE (*type-methods*), 39
- CPPTYPE\_ENUM (*type-methods*), 39
- CPPTYPE\_FLOAT (*type-methods*), 39
- CPPTYPE\_INT32 (*type-methods*), 39
- CPPTYPE\_INT64 (*type-methods*), 39
- CPPTYPE\_MESSAGE (*type-methods*), 39
- CPPTYPE\_STRING (*type-methods*), 39
- CPPTYPE\_UINT32 (*type-methods*), 39
- CPPTYPE\_UINT64 (*type-methods*), 39
- default\_value
  - (*FieldDescriptor-class*), 20
- default\_value, FieldDescriptor-method (*FieldDescriptor-class*), 20
- default\_value-methods (*FieldDescriptor-class*), 20
- Descriptor, 6, 11, 14, 16–21, 28, 29, 31–34
- descriptor (*descriptor-methods*), 16
- descriptor, Message-method (*descriptor-methods*), 16
- Descriptor-class, 14
- descriptor-methods, 16
- enum\_type (*enum\_type-methods*), 18
- enum\_type, Descriptor, ANY, ANY-method (*Descriptor-class*), 14
- enum\_type, EnumValueDescriptor, missing, missing-method (*EnumValueDescriptor-class*), 18
- enum\_type, FieldDescriptor, missing, missing-method (*FieldDescriptor-class*), 20
- enum\_type-methods, 18
- enum\_type\_count
  - (*enum\_type\_count-methods*), 19
- enum\_type\_count, Descriptor-method (*Descriptor-class*), 14
- enum\_type\_count-methods, 19
- EnumDescriptor, 6, 11, 14, 18, 21
- EnumDescriptor-class, 16
- EnumValueDescriptor, 17
- EnumValueDescriptor-class, 18
- fetch (*fetch-methods*), 19
- fetch, Message-method (*fetch-methods*), 19
- fetch-methods, 19
- field (*field-methods*), 19
- field, Descriptor-method (*Descriptor-class*), 14
- field-methods, 19
- field\_count
  - (*field\_count-methods*), 21
- field\_count, Descriptor-method (*Descriptor-class*), 14
- field\_count-methods, 21
- FieldDescriptor, 6, 14, 19, 20, 27, 33, 39
- FieldDescriptor-class, 20
- FileDescriptor, 11
- fileDescriptor, 22
- fileDescriptor
  - (*fileDescriptor-methods*), 22
- fileDescriptor, Descriptor-method (*fileDescriptor-methods*), 22
- fileDescriptor, EnumDescriptor-method (*fileDescriptor-methods*), 22
- fileDescriptor, FieldDescriptor-method (*fileDescriptor-methods*), 22
- fileDescriptor, Message-method (*fileDescriptor-methods*), 22
- fileDescriptor, MethodDescriptor-method (*fileDescriptor-methods*), 22

- 22
- fileDescriptor, ServiceDescriptor-method (MethodDescriptor-class), (fileDescriptor-methods), 22
- FileDescriptor-class, 22
- fileDescriptor-methods, 22
- FileInputStream, 23, 24, 26, 38
- FileInputStream (FileInputStream-methods), 24
- FileInputStream, character, integer, logical-method (FileInputStream-methods), 24
- FileInputStream-class, 23
- FileInputStream-methods, 24
- FileOutputStream, 24–26, 38
- FileOutputStream (FileOutputStream-methods), 25
- FileOutputStream, character, integer, logical-method (FileOutputStream-methods), 25
- FileOutputStream-class, 24
- FileOutputStream-methods, 25
- flush, FileOutputStream-method (FileOutputStream-class), 24
- GetErrno (GetErrno-methods), 26
- GetErrno, FileInputStream-method (FileInputStream-class), 23
- GetErrno, FileOutputStream-method (FileOutputStream-class), 24
- GetErrno-methods, 26
- has (has-methods), 26
- has, Message-method (has-methods), 26
- has-methods, 26
- has\_default\_value (FieldDescriptor-class), 20
- has\_default\_value, FieldDescriptor-method (FieldDescriptor-class), 20
- has\_default\_value-methods (FieldDescriptor-class), 20
- identical, Message, Message-method (Message-class), 28
- input\_type (MethodDescriptor-class), 31
- input\_type, MethodDescriptor-method (MethodDescriptor-class), 31
- input\_type-methods (MethodDescriptor-class), 31
- invoke, 36
- invoke, MethodDescriptor-method (invoke-methods), 26
- invoke, MethodDescriptor, Message, missing-method (invoke-methods), 26
- invoke, MethodDescriptor, Message, RpcHTTP-method (invoke-methods), 26
- invoke-methods, 26
- is\_extension (is\_extension-methods), 27
- is\_extension, FieldDescriptor-method (FieldDescriptor-class), 20
- is\_extension-methods, 27
- is\_optional (FieldDescriptor-class), 20
- is\_optional, FieldDescriptor-method (FieldDescriptor-class), 20
- is\_optional-methods (FieldDescriptor-class), 20
- is\_repeated (FieldDescriptor-class), 20
- is\_repeated, FieldDescriptor-method (FieldDescriptor-class), 20
- is\_repeated-methods (FieldDescriptor-class), 20
- is\_required (FieldDescriptor-class), 20
- is\_required, FieldDescriptor-method (FieldDescriptor-class), 20
- is\_required-methods (FieldDescriptor-class), 20
- isInitialized (isInitialized-methods), 27
- isInitialized, Message-method (isInitialized-methods), 27
- isInitialized-methods, 27
- label (label-methods), 27
- label, FieldDescriptor-method (FieldDescriptor-class), 20
- label-methods, 27

- LABEL\_OPTIONAL (*label-methods*), 27
- LABEL\_REPEATED (*label-methods*), 27
- LABEL\_REQUIRED (*label-methods*), 27
- length, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- length, Message-method  
(*Message-class*), 28
- length, ServiceDescriptor-method  
(*ServiceDescriptor-class*), 37
  
- makeActiveBinding, 40
- merge, Message, Message-method  
(*merge-methods*), 28
- merge-methods, 28
- Message, 2, 6–9, 11, 14, 16, 26–28, 34, 40
- Message-class, 28
- message\_type  
(*FieldDescriptor-class*), 20
- message\_type, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- message\_type-methods  
(*FieldDescriptor-class*), 20
- method (*ServiceDescriptor-class*), 37
- method, ServiceDescriptor-method  
(*ServiceDescriptor-class*), 37
- method-methods  
(*ServiceDescriptor-class*), 37
- method\_count  
(*ServiceDescriptor-class*), 37
- method\_count, ServiceDescriptor-method  
(*ServiceDescriptor-class*), 37
- method\_count-methods  
(*ServiceDescriptor-class*), 37
- MethodDescriptor, 37
- MethodDescriptor-class, 31
  
- name, 32
- name, Descriptor-method (*name*), 32
- name, EnumDescriptor-method  
(*name*), 32
- name, FieldDescriptor-method  
(*name*), 32
- name, FileDescriptor-method  
(*FileDescriptor-class*), 22
- name, MethodDescriptor-method  
(*name*), 32
- name, ServiceDescriptor-method  
(*name*), 32
- name-methods (*name*), 32
- nested\_type  
(*nested\_type-methods*), 32
- nested\_type, Descriptor-method  
(*Descriptor-class*), 14
- nested\_type-methods, 32
- nested\_type\_count  
(*nested\_type\_count-methods*), 32
- nested\_type\_count, Descriptor-method  
(*Descriptor-class*), 14
- nested\_type\_count-methods, 32
- new, Descriptor-method  
(*Descriptor-class*), 14
- Next (*Next-methods*), 33
- Next, ZeroCopyInputStream, missing-method  
(*ZeroCopyInputStream-class*), 40
- Next, ZeroCopyOutputStream, raw-method  
(*ZeroCopyOutputStream-class*), 42
- Next-methods, 33
- number (*number-methods*), 33
- number, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- number-methods, 33
  
- output\_type  
(*MethodDescriptor-class*), 31
- output\_type, MethodDescriptor-method  
(*MethodDescriptor-class*), 31
- output\_type-methods  
(*MethodDescriptor-class*), 31
  
- P, 14, 15, 29, 33, 36
  
- read (*read-methods*), 34
- read, Descriptor, ANY-method  
(*read-methods*), 34

- read, Descriptor, character-method  
(*read-methods*), 34
- read, Descriptor, raw-method  
(*read-methods*), 34
- read-methods, 34
- readASCII (*readASCII-methods*), 35
- readASCII, Descriptor, ANY-method  
(*readASCII-methods*), 35
- readASCII, Descriptor, character-method  
(*readASCII-methods*), 35
- readASCII-methods, 35
- ReadLittleEndian32  
(*ZeroCopyInputStream-class*),  
40
- ReadLittleEndian32, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- ReadLittleEndian32-methods  
(*ZeroCopyInputStream-class*),  
40
- ReadLittleEndian64  
(*ZeroCopyInputStream-class*),  
40
- ReadLittleEndian64, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- ReadLittleEndian64-methods  
(*ZeroCopyInputStream-class*),  
40
- readProtoFiles, 35
- ReadRaw  
(*ZeroCopyInputStream-class*),  
40
- ReadRaw, ZeroCopyInputStream, integer-method  
(*ZeroCopyInputStream-class*),  
40
- ReadRaw-methods  
(*ZeroCopyInputStream-class*),  
40
- ReadString  
(*ZeroCopyInputStream-class*),  
40
- ReadString, ZeroCopyInputStream, integer-method  
(*ZeroCopyInputStream-class*),  
40
- ReadString-methods  
(*ZeroCopyInputStream-class*),  
40
- ReadVarint32  
(*ZeroCopyInputStream-class*),  
40
- ReadVarint32, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- ReadVarint32-methods  
(*ZeroCopyInputStream-class*),  
40
- ReadVarint64  
(*ZeroCopyInputStream-class*),  
40
- ReadVarint64, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- ReadVarint64-methods  
(*ZeroCopyInputStream-class*),  
40
- RpcHTTP-class, 36
- RProtoBuf (*RProtoBuf-package*), 2
- RProtoBuf-package, 2
- serialize, Message-method  
(*Message-class*), 28
- ServiceDescriptor-class, 37
- set (*set-methods*), 37
- set, Message-method (*set-methods*),  
37
- set-methods, 37
- SetCloseOnDelete  
(*SetCloseOnDelete-methods*),  
38
- SetCloseOnDelete, FileInputStream-method  
(*FileInputStream-class*), 23
- SetCloseOnDelete, FileOutputStream-method  
(*FileOutputStream-class*),  
24
- SetCloseOnDelete-methods, 38
- show, Descriptor-method  
(*Descriptor-class*), 14
- show, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- show, EnumValueDescriptor-method  
(*EnumValueDescriptor-class*),  
18
- show, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- show, FileDescriptor-method  
(*FileDescriptor-class*), 22

- show, Message-method  
(*Message-class*), 28
- show, ServiceDescriptor-method  
(*ServiceDescriptor-class*),  
37
- size (*size-methods*), 38
- size, Message-method  
(*size-methods*), 38
- size-methods, 38
- size<- (*sizegets*), 38
- size<-, Message-method (*sizegets*),  
38
- size<-methods (*sizegets*), 38
- sizegets, 38
- Skip (*Skip-methods*), 38
- Skip, ZeroCopyInputStream-method  
(*ZeroCopyInputStream-class*),  
40
- Skip-methods, 38
- str, Message-method  
(*Message-class*), 28
- swap (*swap-methods*), 39
- swap, Message-method  
(*swap-methods*), 39
- swap-methods, 39
- toString, Descriptor-method  
(*Descriptor-class*), 14
- toString, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- toString, EnumValueDescriptor-method  
(*EnumValueDescriptor-class*),  
18
- toString, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- toString, FileDescriptor-method  
(*FileDescriptor-class*), 22
- toString, Message-method  
(*Message-class*), 28
- toString, MethodDescriptor-method  
(*MethodDescriptor-class*),  
31
- toString, ServiceDescriptor-method  
(*ServiceDescriptor-class*),  
37
- type (*type-methods*), 39
- type, FieldDescriptor-method  
(*FieldDescriptor-class*), 20
- type-methods, 39
- TYPE\_BOOL (*type-methods*), 39
- TYPE\_BYTES (*type-methods*), 39
- TYPE\_DOUBLE (*type-methods*), 39
- TYPE\_ENUM (*type-methods*), 39
- TYPE\_FIXED32 (*type-methods*), 39
- TYPE\_FIXED64 (*type-methods*), 39
- TYPE\_FLOAT (*type-methods*), 39
- TYPE\_GROUP (*type-methods*), 39
- TYPE\_INT32 (*type-methods*), 39
- TYPE\_INT64 (*type-methods*), 39
- TYPE\_MESSAGE (*type-methods*), 39
- TYPE\_SFIXED32 (*type-methods*), 39
- TYPE\_SFIXED64 (*type-methods*), 39
- TYPE\_SINT32 (*type-methods*), 39
- TYPE\_SINT64 (*type-methods*), 39
- TYPE\_STRING (*type-methods*), 39
- TYPE\_UINT32 (*type-methods*), 39
- TYPE\_UINT64 (*type-methods*), 39
- update, Message-method  
(*Message-class*), 28
- value (*EnumDescriptor-class*), 16
- value, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- value-methods  
(*EnumDescriptor-class*), 16
- value\_count  
(*EnumDescriptor-class*), 16
- value\_count, EnumDescriptor-method  
(*EnumDescriptor-class*), 16
- value\_count-methods  
(*EnumDescriptor-class*), 16
- with.Message, 39
- within.Message (*with.Message*), 39
- WriteLittleEndian32  
(*ZeroCopyOutputStream-class*),  
42
- WriteLittleEndian32, ZeroCopyOutputStream, integer  
(*ZeroCopyOutputStream-class*),  
42
- WriteLittleEndian32, ZeroCopyOutputStream, numeric  
(*ZeroCopyOutputStream-class*),  
42
- WriteLittleEndian32, ZeroCopyOutputStream, raw-met  
(*ZeroCopyOutputStream-class*),  
42

- WriteLittleEndian32-methods  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteLittleEndian64  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteLittleEndian64, ZeroCopyOutputStream, integer-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteLittleEndian64, ZeroCopyOutputStream, numeric-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteLittleEndian64, ZeroCopyOutputStream, raw-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteLittleEndian64-methods  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteRaw  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteRaw, ZeroCopyOutputStream, raw-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteRaw-methods  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteString  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteString, ZeroCopyOutputStream, character-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteString-methods  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint32  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint32, ZeroCopyOutputStream, integer-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint32, ZeroCopyOutputStream, numeric-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint32, ZeroCopyOutputStream, raw-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint32-methods  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint64  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint64, ZeroCopyOutputStream, integer-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint64, ZeroCopyOutputStream, numeric-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint64, ZeroCopyOutputStream, raw-method  
(ZeroCopyOutputStream-class),  
[42](#)
- WriteVarint64-methods  
(ZeroCopyOutputStream-class),  
[42](#)
- ZeroCopyInputStream, [3](#), [4](#), [8](#), [11](#), [12](#),  
[23](#), [24](#), [33](#)
- ZeroCopyInputStream-class, [40](#)
- ZeroCopyOutputStream, [4](#), [5](#), [13](#), [24](#), [25](#)
- ZeroCopyOutputStream-class, [42](#)