Package 'RProtoBuf'

January 21, 2024

Version 0.4.22 **Date** 2024-01-20

Author Romain Francois, Dirk Eddelbuettel, Murray Stokely and Jeroen Ooms

Maintainer Dirk Eddelbuettel <edd@debian.org>

Title R Interface to the 'Protocol Buffers' 'API' (Version 2 or 3)

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. Additional documentation is available in two included vignettes one of which corresponds to our 'JSS' paper (2016, <doi:10.18637/jss.v071.i02>. A sufficiently recent version of 'Protocol Buffers' library is required; currently version 3.3.0 from 2017 is the stated minimum.

Depends R (>= 3.0.0), methods **Imports** utils, stats, tools, Rcpp

LinkingTo Rcpp Suggests tinytest

SystemRequirements ProtoBuf libraries and compiler version 3.3.0 or later; On Debian/Ubuntu these can be installed as libprotoc-dev, libprotobuf-dev and protobuf-compiler, while on Fedora/CentOS protobuf-devel and protobuf-compiler are needed. A C++17 compiler is required as well.

BugReports https://github.com/eddelbuettel/rprotobuf/issues

URL https://github.com/eddelbuettel/rprotobuf,

https://dirk.eddelbuettel.com/code/rprotobuf.html

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-01-21 13:42:45 UTC

${\sf R}$ topics documented:

RProtoBuf-package
add-methods
ArrayInputStream-class
ArrayInputStream-methods
ArrayOutputStream-class
ArrayOutputStream-methods
as.list.Message
asMessage
BackUp-methods
ByteCount-methods
bytesize-methods
clear-methods
clone-methods
completion
ConnectionInputStream-class
ConnectionInputStream-methods
ConnectionOutputStream-class
ConnectionOutputStream-methods
containing_type-methods
Descriptor-class
descriptor-methods
EnumDescriptor-class
EnumValueDescriptor-class
enum_type-methods
enum_type_count-methods
fetch-methods
field-methods
FieldDescriptor-class
field_count-methods
FileDescriptor-class
fileDescriptor-methods
FileInputStream-class
FileInputStream-methods
FileOutputStream-class
FileOutputStream-methods
GetErrno-methods
has-methods
isInitialized-methods
is_extension-methods
label-methods
merge-methods
Message-class
MethodDescriptor-class
name
nested_type-methods
nested_type_count-methods

RProtoBuf-package	
KI 1010Bu1-package	

	Next-methods	37
	number-methods	37
	P	38
	read-methods	38
	readASCII-methods	39
	readJSON-methods	40
	readProtoFiles	41
	serialize_pb	42
	ServiceDescriptor-class	43
	set-methods	
	SetCloseOnDelete-methods	44
	size-methods	
	sizegets	
	Skip-methods	
	swap-methods	
	type-methods	
	with.Message	
	ZeroCopyInputStream-class	48
	ZeroCopyOutputStream-class	
Index		51
		_
RPro	toBuf-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

Author(s)

Romain Francois, Dirk Eddelbuettel, Murray Stokely and Jeroen Ooms.

References

https://github.com/eddelbuettel/rprotobuf

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

Examples

```
unitest.proto.file <- system.file("tinytest", "data", "unittest.proto",
  package = "RProtoBuf" )
readProtoFiles(file = unitest.proto.file)

test <- new(protobuf_unittest.TestAllTypes)
test$add("repeated_int32", 1)
test$add("repeated_int32", 2:10)
test$repeated_int32</pre>
```

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 François <françoisromain@free.fr>

References

The ArrayInputStream class from the protobuf C++ library.

See Also

ZeroCopyInputStream for methods

Examples

```
stream <- ArrayInputStream(as.raw(0:10))
stream$ReadRaw(5)

stringsstream <- ArrayInputStream(as.raw(c(0x74, 0x65, 0x73, 0x74, 0x69, 0x6e, 0x67)))
stringsstream$ReadString(7)

intstream <- ArrayInputStream(as.raw(c(0x9e, 0xa7, 0x05)))
intstream$ReadVarint32()</pre>
```

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 François <françoisromain@free.fr>

References

The ArrayOutputStream class from the protobuf C++ library.

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, ...)
```

8 asMessage

Arguments

A protocol buffer message, instance of Message, or a protocol message descriptor, instance of Descriptor
 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), then field descriptors (FieldDescriptor objects) in that order.

For enum descriptors, a named list of the enumerated values.

For file descriptors, a named list of descriptors defined in the specified file descriptor.

For service descriptors, ...

Author(s)

Romain Francois <francoisromain@free.fr>

Examples

```
Person <- P( "tutorial.Person" )
romain <- new( Person, email = "francoisromain@free.fr", id = 1 )
as.list( romain )
as.list( Person )
as.list( Person$PhoneType)</pre>
```

asMessage

coerce an object to a protobuf message

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 as

Value

```
a Message object
```

BackUp-methods 9

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.

10 clear-methods

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

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

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

Examples

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

clone-methods 11

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

clone-methods

Clone protocol buffer messages

Description

Generic "clone" function and associated method for Message objects

Methods

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

Examples

```
## Not run:
# 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 )

## End(Not run)

# 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 )</pre>
```

12 completion

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 François <françoisromain@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

 ${\tt ConnectionInputStream-methods}$

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

 ${\tt ConnectionOutputStream-methods}$

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

Examples

```
# Containing type of a field is the message descriptor
tutorial.Person$id$containing_type()
```

No containing type for the top-level message descriptor.
tutorial.Person\$containing_type()

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.

Descriptor-class

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 of 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 of 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

[[signature(x = "Descriptor"): extracts a field identified by its name or declared tag number names signature(x = "Descriptor"): extracts names of this descriptor length signature(x = "Descriptor"): extracts length of this descriptor

Author(s)

Romain Francois <francoisromain@free.fr>

descriptor-methods 17

See Also

the P function creates "Descriptor" messages.

Examples

```
## Not run:
# 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 )

## End(Not run)

# enum type
Person$PhoneType

# nested type
Person$PhoneNumber

# field
Person$email

# use this descriptor to create a message
new( Person )</pre>
```

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 descrip-
     tor. 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
has signature(object = "EnumDescriptor"): indicate if the given name is a constant present in
     this enum.
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.

[[signature(x = "EnumDescriptor"): extracts field identified by its name or declared tag num-

```
names signature(x = "EnumDescriptor") : extracts names of this enum
```

Author(s)

Romain François <françoisromain@free.fr>

References

The EnumDescriptor C++ class

See Also

The Descriptor class

Examples

```
## Not run:
# 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 )

## End(Not run)

# enum type
Person$PhoneType, "MOBILE")
has(Person$PhoneType, "HOME")
has(Person$PhoneType, "WORK")

has(Person$PhoneType, "FOOBAR")
length(Person$PhoneType)</pre>

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
name: simple name of the enum
full_name: fully qualified name
```

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
```

20 enum_type-methods

```
name signature(object = "EnumValueDescriptor", full = "logical"): return the name of
    this enum constant.
```

number signature(object = "EnumValueDescriptor"): return the numeric value of this enum
constant.

enum_type signature(object = "EnumDescriptor") : retrieves the EnumDescriptor related to
this value descriptor.

Author(s)

Romain François <françoisromain@free.fr>

Examples

```
## Not run:
# 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 )

## End(Not run)

# enum type
Person$PhoneType

# enum value type
value(Person$PhoneType, 1)

name(value(Person$PhoneType, 1))
name(value(Person$PhoneType, 1), TRUE)

number(value(Person$PhoneType, number=1))
enum_type(value(Person$PhoneType, number=1))</pre>
```

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

22 FieldDescriptor-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 descrip-
     tor. 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 mes-
     sage 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 de-
     fault value.
default_value signature(object = "FieldDescriptor"): the default value of this field.
message_type signature(object = "FieldDescriptor"): the Descriptor for the associated mes-
     sage 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
```

FieldDescriptor-class 23

Author(s)

Romain François <françoisromain@free.fr>

References

The FieldDescriptor C++ class

See Also

Descriptor

Examples

```
## Not run:
# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )</pre>
# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )</pre>
## End(Not run)
# field descriptor object
Person$email
# debug string
as.character( Person$email )
# or as a pseudo method
Person$email$as.character()
Person$email$is_required()
Person$email$is_optional()
Person$email$is_repeated()
Person$email$has_default_value()
Person$email$default_value()
Person$email$is_extension()
# Get the default values
has_default_value(Person$id)
has_default_value(Person$email)
has_default_value(Person$phone)
default_value(Person$id)
default_value(Person$email)
default_value(Person$phone)
# Get the types of field descriptors
type(Person$id)
type(Person$id, as.string=TRUE)
```

24 field_count-methods

```
cpp_type(Person$email)
cpp_type(Person$email, TRUE)
# Get the label of a field descriptor
label(Person$id)
label(Person$email)
label(Person$phone)
label(Person$id, TRUE)
label(Person$email, TRUE)
label(Person$phone, TRUE)
LABEL_OPTIONAL
LABEL_REQUIRED
LABEL_REPEATED
# Test if a field is optional
is_optional(Person$id)
is_optional(Person$email)
is_optional(Person$phone)
# Test if a field is repeated
is_repeated(Person$id)
is_repeated(Person$email)
is_repeated(Person$phone)
# Test if a field is required
is_required(Person$id)
is_required(Person$email)
is_required(Person$phone)
# Return the class of a message field
message_type(Person$phone)
```

field_count-methods The

The number of fields

Description

The number of fields

See Also

The method is implemented for the Descriptor class

FileDescriptor-class 25

```
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 package: the package name defined in the file, e.g. 'tutorial'. filename: the filename of this FileDescriptor
```

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 François <françoisromain@free.fr>

See Also

Descriptor

Examples

```
# example proto file supplied with this package
desc <- P("tutorial.Person")
person <- new(desc)

person$fileDescriptor()
name(person$fileDescriptor())
# [1] "addressbook.proto"
as.character(person$fileDescriptor())</pre>
```

26 FileInputStream-class

```
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 error 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 François <françoisromain@free.fr>

References

The FileInputStream class from the protobuf C++ library.

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 error 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 François <françoisromain@free.fr>

References

The FileOutputStream class from the protobuf C++ library.

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 an object has the given field set

Description

This generic method, currently implemented for Message and EnumDescriptor indicates if the message or enum descriptor has the given field set.

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

For messages and 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.

NULL is returned if the descriptor for the message does not contain the given field at all.

For EnumDescriptors, a boolean value indicates if the given name is present in the enum definition.

Methods

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

has signature(object = "EnumDescriptor"): Indicates if the EnumDescriptor has a given named element.

30 isInitialized-methods

Examples

```
unitest.proto.file <- system.file("tinytest", "data", "unittest.proto",
    package = "RProtoBuf" )
readProtoFiles(file = unitest.proto.file)

test <- new(protobuf_unittest.TestAllTypes)
test$has("optional_int32")
# FALSE
test$add("repeated_int32", 1:10)
test$has("repeated_int32")
# TRUE
test$has("nonexistant")
# NULL
has(protobuf_unittest.TestAllTypes$NestedEnum, "FOO")
has(protobuf_unittest.TestAllTypes$NestedEnum, "BAR")
has(protobuf_unittest.TestAllTypes$NestedEnum, "XXX")</pre>
```

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</pre>
```

is_extension-methods 31

Description

Indicates if a field descriptor is an extension

See Also

The method is implemented for the FieldDescriptor class

Examples

```
Person <- P( "tutorial.Person" )
is_extension(Person$id)</pre>
```

label-methods

Gets the label of a field

Description

Gets the label of a field (optional, required, or repeated).

Arguments

object A FieldDescriptor object.

as.string If true, print a string representation of the type.

See Also

The method is implemented for the FieldDescriptor class

Examples

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )

## End(Not run)

label(Person$id)
label(Person$email)
label(Person$phone)
label(Person$phone)
label(Person$email, TRUE)
label(Person$phone, TRUE)</pre>
```

32 Message-class

```
LABEL_OPTIONAL
LABEL_REQUIRED
LABEL_REPEATED
```

merge-methods

Merge two messages of the same type

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 ) )</pre>
```

Message-class

Class "Message"

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
```

Message-class 33

Methods

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

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

toTextFormat signature(x = "Message"): returns the TextFormat of the message. This is built from a call to TextFormat::PrintToString with the Message object

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

- toJSON signature(x = "Message"): returns the JSON representation of the message. This is built from a call to the google::protobuf::util::MessageToJsonString method and accepts two arguments preserve_proto_field_names - if FALSE (the default) convert field names to camelCase always_print_primitive_fields - whether to return the default value for missing primitive fields (default false)
- <- 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.

setExtension signature(object = "Message"): set an extension field of the Message.

getExtension signature(object = "Message"): get the value of an extension field of the Message.

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

names signature(x = "Message"): extracts the names of the message.

Author(s)

Romain Francois <francoisromain@free.fr>

34 Message-class

References

The Message class from the C++ proto library.

See Also

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

Examples

```
## Not run:
# example proto file supplied with this package
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )</pre>
# reading a proto file and creating the descriptor
Person <- P( "tutorial.Person", file = proto.file )</pre>
## End(Not run)
PhoneNumber <- P( "tutorial.Person.PhoneNumber" )</pre>
# creating a prototype message from the descriptor
p <- new( Person )</pre>
p$email # not set, returns default value
      # not set, returns default value
as.character( p ) # empty
has( p, "email" ) # is the "email" field set
has( p, "phone" ) # is the "email" field set
                # number of fields actually set
length( p )
# update several fields at once
romain <- update( new( Person ),</pre>
email = "francoisromain@free.fr",
name = "Romain Francois",
phone = new( PhoneNumber , number = "+33(0)...", type = "MOBILE" )
)
# supply parameters to the constructor
dirk <- new( Person,
email = "edd@debian.org",
id = 2,
name = "Dirk Eddelbuettel" )
# update the phone repeated field with a list of PhoneNumber messages
dirk$phone <- list(</pre>
new( PhoneNumber , number = "+01...", type = "MOBILE" ),
new( PhoneNumber , number = "+01...", type = "HOME" ) )
# with/within style
saptarshi <- within( new(Person), {</pre>
id <- 3
name <- "Saptarshi Guha"
```

MethodDescriptor-class

```
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 )</pre>
```

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</pre>
```

Author(s)

Romain François <françoisromain@free.fr>

name

Name or full name of a descriptor

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 37

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 FieldDescriptor and EnumValueDescriptor classes.

Examples

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )

## End(Not run)

number(Person$id)
number(Person$email)
as.character(Person)

number(value(tutorial.Person$PhoneType, name="HOME"))</pre>
```

38 read-methods

Ρ

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 or extension

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 for message types or FieldDescriptor for extensions. An error is generated otherwise.

Author(s)

Romain François <françoisromain@free.fr>

Examples

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )
## End(Not run)

cat(as.character( Person ))</pre>
```

read-methods

Read a protocol buffer message from a connection

Description

Read a Message from a connection using its associated Descriptor

readASCII-methods 39

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" )</pre>
# read the message
message <- read( tutorial.AddressBook, book )</pre>
# or using the pseudo method
message <- tutorial.AddressBook$read( book )</pre>
# write its debug string
writeLines( as.character( message ) )
# grab the name of each person
sapply( message$person, function(p) p$name )
# read from a binary file connection
f <- file( book, open = "rb" )</pre>
message2 <- read( tutorial.AddressBook, f )</pre>
close( f )
# read from a message payload (raw vector)
payload <- readBin( book, raw(0), 5000 )</pre>
message3 <- tutorial.AddressBook$read( payload )</pre>
```

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
```

40 readJSON-methods

Examples

```
## Not run:
# example file that contains a "tutorial.AddressBook" message
book <- system.file( "examples", "addressbook.pb", package = "RProtoBuf" )

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

# Output in text format to a temporary file
out.file <- tempfile()
writeLines( as.character(message), file(out.file))

# Verify that we can read back in the message from a text file.
message2 <- readASCII( tutorial.AddressBook, file(out.file, "rb"))

# Verify that we can read back in the message from an unopened file.
message3 <- readASCII( tutorial.AddressBook, file(out.file))

\dontshow{
stopifnot( identical( message, message2) )
}

## End(Not run)</pre>
```

readJSON-methods

read a message in JSON format

Description

Method to read a Message in JSON 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
```

Examples

```
## Not run:
# example file that contains a "tutorial.AddressBook" message
book <- system.file( "examples", "addressbook.pb", package = "RProtoBuf" )
# read the message
message <- read( tutorial.AddressBook, book )
# Output in text format to a temporary file
out.file <- tempfile()</pre>
```

readProtoFiles 41

```
writeLines( message$toJSON(), file(out.file))

# Verify that we can read back in the message from a text file.
message2 <- readJSON( tutorial.AddressBook, file(out.file, "rb"))

# Verify that we can read back in the message from an unopened file.
message3 <- readJSON( tutorial.AddressBook, file(out.file))

\dontshow{
stopifnot( identical( message, message2) )
}

## End(Not run)</pre>
```

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", pattern="\\.proto$", lib.loc=NULL)
readProtoFiles2(files, dir=".", pattern="\\.proto$", recursive=FALSE, protoPath=getwd())
resetDescriptorPool()
```

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.
pattern	A filename pattern to match proto files when using dir.
recursive	Whether to descend recursively into dir.
lib.loc	Library location.
protoPath	Search path for proto file imports.

Details

readProtoFiles2 is different from readProtoFiles to be consistent with the behavior of protoc command line tool in being explicit about the search path for proto import statements. In addition, we also require that both files and dir arguments are interpreted relative to protoPath, so that there is consistency in future imports of the same files through import statements of other proto files.

resetDescriptorPool clears all imported proto definitions.

42 serialize_pb

Value

```
NULL, invisibly.
```

Author(s)

Romain Francois <francoisromain@free.fr>

See Also

Р

Examples

```
## Not run:
# 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)

## End(Not run)</pre>
```

serialize_pb

Serialize R object to Protocol Buffer Message.

Description

Serializes R objects to a general purpose protobul message using the same rexp.proto descriptor and mapping between R objects and protobul messages as RHIPE.

Usage

```
serialize_pb(object, connection, ...)
```

Arguments

```
object R object to serialize

connection passed on to serialize

... additional arguments passed on to serialize
```

Details

Clients need both the message and the rexp.proto descriptor to parse serialized R objects. The latter is included in the the package installation proto directory: system.file(package="RProtoBuf", "proto/rexp.proto")

The following storage types are natively supported by the descriptor: character, raw, double, complex, integer, list, and NULL. Objects with other storage types, such as functions, environments, S4 classes, etc, are serialized using base R serialize and stored in the proto native type. Missing values, attributes and numeric precision will be preserved.

Examples

```
msg <- tempfile();
serialize_pb(iris, msg);
obj <- unserialize_pb(msg);
identical(iris, obj);</pre>
```

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
```

method signature(x = "ServiceDescriptor"): retrieves a MethodDescriptor

Methods

44 size-methods

Author(s)

Romain François <françoisromain@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.

For repeated fields, the size can also be assigned to in order to shrink or grow the vector. Numeric types are given a default value of 0 when the new size is greater than the existing size. Character types are given a default value of "". Growing a repeated field in this way is not supported for message, group, and enum types.

Methods

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

sizegets 45

Examples

```
unitest.proto.file <- system.file("tinytest", "data", "unittest.proto",
   package = "RProtoBuf" )
readProtoFiles(file = unitest.proto.file)

test <- new(protobuf_unittest.TestAllTypes)
test$size("optional_int32")

test$add("repeated_int32", 1:10)
test$size("repeated_int32")
test$repeated_int32
size(test, "repeated_int32") <- 5
test$repeated_int32</pre>
size(test, "repeated_int32") <- 15
test$repeated_int32
```

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

46 type-methods

swap-methods

swap elements of a repeated field of a message

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.

type-methods

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

Description

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

Arguments

object A FieldDescriptor object.

as.string If true, print a string representation of the type.

See Also

The method is implemented for the FieldDescriptor class

Examples

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )

## End(Not run)

type(Person$id)
type(Person$id, as.string=TRUE)
cpp_type(Person$email)
cpp_type(Person$email, TRUE)</pre>
```

with.Message 47

with.Message

with and within methods for protocol buffer messages

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

data A protocol buffer message, instance of Message expr R expression to evaluate 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 François <françoisromain@free.fr>

Examples

```
## Not run:
proto.file <- system.file( "proto", "addressbook.proto", package = "RProtoBuf" )
Person <- P( "tutorial.Person", file = proto.file )

## End(Not run)

romain <- within( new( Person ), {
email <- "francoisromain@free.fr"
id <- 10L
} )</pre>
```

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
- ReadRaw signature(object="ZeroCopyInputStream", size = "numeric"): read raw bytes
 from the stream
- **ReadString** signature(object="ZeroCopyInputStream", size = "integer"): same as ReadRaw but formats the result as a string
- **ReadString** signature(object="ZeroCopyInputStream", size = "numeric"): 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 looses 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 looses some precision (no other way)

Author(s)

Romain François <françoisromain@free.fr>

References

The google::protobuf::io::ZeroCopyInputStream C++ class.

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="ZeroCopyOuputStream"), payload = "raw": write the raw bytes
to the stream

Author(s)

Romain François <françoisromain@free.fr>

References

The google::protobuf::io::ZeroCopyOutputStream C++ class.

See Also

TODO: add classes that extend

Index

!=,Message,Message-method	field_count-methods, 24
(Message-class), 32	fileDescriptor-methods, 26
* classes	FileInputStream-methods, 27
ArrayInputStream-class,4	FileOutputStream-methods, 29
ArrayOutputStream-class, 6	GetErrno-methods, 29
ConnectionInputStream-class, 13	has-methods, 29
ConnectionOutputStream-class, 14	is_extension-methods, 31
Descriptor-class, 15	isInitialized-methods, 30
EnumDescriptor-class, 17	label-methods, 31
EnumValueDescriptor-class, 19	merge-methods, 32
FieldDescriptor-class, 22	name, 36
FileDescriptor-class, 25	<pre>nested_type-methods, 36</pre>
FileInputStream-class, 26	<pre>nested_type_count-methods, 36</pre>
FileOutputStream-class, 28	Next-methods, 37
Message-class, 32	number-methods, 37
MethodDescriptor-class, 35	read-methods, 38
ServiceDescriptor-class, 43	readASCII-methods, 39
with.Message, 47	readJSON-methods, 40
ZeroCopyInputStream-class, 48	set-methods, 44
ZeroCopyOutputStream-class, 49	SetCloseOnDelete-methods, 44
* interface	size-methods, 44
P, 38	sizegets, 45
* methods	Skip-methods, 45
add-methods, 4	swap-methods, 46
ArrayInputStream-methods, 5	type-methods, 46
ArrayOutputStream-methods, 7	* package
BackUp-methods, 9	RProtoBuf-package, 3
ByteCount-methods, 9	* programming
bytesize-methods, 10	as.list.Message,7
clear-methods, 10	asMessage, 8
clone-methods, 11	completion, 12
ConnectionInputStream-methods, 14	readProtoFiles,41
ConnectionOutputStream-methods, 15	.DollarNames.Descriptor(completion), 12
<pre>containing_type-methods, 15</pre>	.DollarNames.EnumDescriptor
descriptor-methods, 17	(completion), 12
$\operatorname{enum_type-methods}, 20$.DollarNames.FieldDescriptor
<pre>enum_type_count-methods, 21</pre>	(completion), 12
fetch-methods, 21	.DollarNames.FileDescriptor
field-methods, 21	(completion), 12

.DollarNames.Message(completion), 12	all.equal,Message,Message-method
.DollarNames.MethodDescriptor	(Message-class), 32
(completion), 12	ArrayInputStream, 5 , 6
.DollarNames.ServiceDescriptor	ArrayInputStream
(completion), 12	(ArrayInputStream-methods), 5
.DollarNames.ZeroCopyInputStream	ArrayInputStream,raw,integer-method
(completion), 12	(ArrayInputStream-methods), 5
.DollarNames.ZeroCopyOutputStream	ArrayInputStream,raw,missing-method
(completion), 12	(ArrayInputStream-methods), 5
==,Message,Message-method	ArrayInputStream,raw,numeric-method
(Message-class), 32	(ArrayInputStream-methods), 5
[[,Descriptor-method	ArrayInputStream-class, 4
(Descriptor-class), 15	ArrayInputStream-methods, 5
[[,EnumDescriptor-method	ArrayOutputStream, 6, 7
(EnumDescriptor-class), 17	ArrayOutputStream
[[,Message-method(Message-class), 32	(ArrayOutputStream-methods), 7
[[,ServiceDescriptor-method	ArrayOutputStream,integer,integer-method
(ServiceDescriptor-class), 43	(ArrayOutputStream-methods), 7
[[<-,Message-method(Message-class), 32	ArrayOutputStream,integer,missing-method
\$,Descriptor-method (Descriptor-class),	(ArrayOutputStream-methods), 7
15	ArrayOutputStream,integer,numeric-method
\$,EnumDescriptor-method	(ArrayOutputStream-methods), 7
(EnumDescriptor-class), 17	ArrayOutputStream, numeric, integer-method
\$,EnumValueDescriptor-method	(ArrayOutputStream-methods), 7
(EnumValueDescriptor-class), 19	ArrayOutputStream, numeric, missing-method
\$,FieldDescriptor-method	(ArrayOutputStream-methods), 7
(FieldDescriptor-class), 22	
\$,FileDescriptor-method	ArrayOutputStream, numeric, numeric-method
(FileDescriptor-class), 25	(ArrayOutputStream-methods), 7
\$, Message-method (Message-class), 32	ArrayOutputStream-class, 6
\$, MethodDescriptor-method	ArrayOutputStream-methods, 7
(MethodDescriptor-class), 35	as, 8
	as.character,Descriptor-method
<pre>\$,ServiceDescriptor-method</pre>	(Descriptor-class), 15
	as.character,EnumDescriptor-method
\$, ZeroCopyInputStream-method	(EnumDescriptor-class), 17
(ZeroCopyInputStream-class), 48	as.character,EnumValueDescriptor-method
\$, ZeroCopyOutputStream-method	(EnumValueDescriptor-class), 19
(ZeroCopyOutputStream-class),	as.character,FieldDescriptor-method
49	(FieldDescriptor-class), 22
\$<-,Descriptor-method	as.character,FileDescriptor-method
(Descriptor-class), 15	(FileDescriptor-class), 25
\$<-, Message-method (Message-class), 32	as.character,Message-method
\$<-,MethodDescriptor-method	(Message-class), 32
(MethodDescriptor-class), 35	as.character,MethodDescriptor-method
	(MethodDescriptor-class), 35
add (add-methods), 4	as.character,ServiceDescriptor-method
add, Message-method (add-methods), 4	(ServiceDescriptor-class), 43
add-methods, 4	as.list.Descriptor(as.list.Message),7

as.list.EnumDescriptor	ConnectionInputStream, 13, 14
(as.list.Message), 7	ConnectionInputStream
as.list.FileDescriptor	(ConnectionInputStream-methods),
(as.list.Message), 7	14
as.list.Message,7	ConnectionInputStream,connection-method
as.list.ServiceDescriptor	(ConnectionInputStream-methods),
(as.list.Message), 7	14
asMessage, 8	ConnectionInputStream-class, 13
	ConnectionInputStream-methods, 14
BackUp (BackUp-methods), 9	ConnectionOutputStream, 14, 15
BackUp,ZeroCopyInputStream-method	ConnectionOutputStream
(ZeroCopyInputStream-class), 48	(ConnectionOutputStream-methods),
BackUp,ZeroCopyOutputStream-method	15
<pre>(ZeroCopyOutputStream-class),</pre>	ConnectionOutputStream,connection-method
49	(ConnectionOutputStream-methods),
BackUp-methods, 9	15
ByteCount (ByteCount-methods), 9	ConnectionOutputStream-class, 14
ByteCount,ZeroCopyInputStream-method	ConnectionOutputStream-methods, 15
(ZeroCopyInputStream-class), 48	containing_type
ByteCount,ZeroCopyOutputStream-method	(containing_type-methods), 15
<pre>(ZeroCopyOutputStream-class),</pre>	<pre>containing_type,Descriptor-method</pre>
49	(Descriptor-class), 15
ByteCount-methods, 9	<pre>containing_type,EnumDescriptor-method</pre>
bytesize (bytesize-methods), 10	(EnumDescriptor-class), 17
bytesize, Message-method	<pre>containing_type,FieldDescriptor-method</pre>
(bytesize-methods), 10	(FieldDescriptor-class), 22
bytesize-methods, 10	<pre>containing_type-methods, 15</pre>
can_serialize_pb (serialize_pb), 42	<pre>cpp_type (type-methods), 46</pre>
clear (clear-methods), 10	<pre>cpp_type,FieldDescriptor-method</pre>
clear, Message, character-method	(FieldDescriptor-class), 22
(clear-methods), 10	cpp_type-methods(type-methods),46
clear, Message, integer-method	CPPTYPE_BOOL (type-methods), 46
(clear-methods), 10	CPPTYPE_DOUBLE (type-methods), 46
clear, Message, missing-method	CPPTYPE_ENUM (type-methods), 46
(clear-methods), 10	CPPTYPE_FLOAT (type-methods), 46
clear, Message, numeric-method	CPPTYPE_INT32 (type-methods), 46
(clear-methods), 10	CPPTYPE_INT64 (type-methods), 46
clear, Message, raw-method	CPPTYPE_MESSAGE (type-methods), 46
(clear-methods), 10	CPPTYPE_STRING(type-methods), 46
clear-methods, 10	CPPTYPE_UINT32 (type-methods), 46
clone (clone-methods), 11	CPPTYPE_UINT64 (type-methods), 46
clone, Message-method (clone-methods), 11	
clone-methods, 11	<pre>default_value (FieldDescriptor-class),</pre>
close,FileInputStream-method	22
(FileInputStream-class), 26	default_value,FieldDescriptor-method
close, FileOutputStream-method	(FieldDescriptor-class), 22
(FileOutputStream-class), 28	default_value-methods
completion, 12	(FieldDescriptor-class), 22

Descriptor, 8, 12, 15, 17, 18, 20–25, 32,	fileDescriptor,FieldDescriptor-method
34–36, 38	(fileDescriptor-methods), 26
descriptor (descriptor-methods), 17	fileDescriptor,Message-method
descriptor, Message-method	(fileDescriptor-methods), 26
(descriptor-methods), 17	fileDescriptor,MethodDescriptor-method
Descriptor-class, 15	(fileDescriptor-methods), 26
descriptor-methods, 17	fileDescriptor,ServiceDescriptor-method
	(fileDescriptor-methods), 26
<pre>enum_type (enum_type-methods), 20</pre>	FileDescriptor-class, 25
enum_type,Descriptor,ANY,ANY-method	fileDescriptor-methods, 26
(Descriptor-class), 15	FileInputStream, 26, 27, 29, 44
<pre>enum_type,EnumValueDescriptor,missing,miss</pre>	singEinethowoutStream
(EnumValueDescriptor-class), 19	(FileInputStream-methods), 27
enum_type,FieldDescriptor,missing,missing-	-metհենխeInputStream,character,integer,logical-method
(FieldDescriptor-class), 22	(FileInputStream-methods), 27
enum_type-methods, 20	FileInputStream-class, 26
enum_type_count	FileInputStream-methods, 27
(enum_type_count-methods), 21	FileOutputStream, 28, 29, 44
enum_type_count,Descriptor-method	FileOutputStream
(Descriptor-class), 15	(FileOutputStream-methods), 29
enum_type_count-methods, 21	FileOutputStream, character, integer, logical-method
EnumDescriptor, 8, 12, 15, 16, 19, 20, 22, 29	(FileOutputStream-methods), 29
EnumDescriptor-class, 17	FileOutputStream-class, 28
EnumValueDescriptor, 18, 37	FileOutputStream-methods, 29
EnumValueDescriptor-class, 19	flush,FileOutputStream-method
,	(FileOutputStream-class), 28
fetch (fetch-methods), 21	
fetch, Message-method (fetch-methods), 21	GetErrno (GetErrno-methods), 29
fetch-methods, 21	GetErrno,FileInputStream-method
field (field-methods), 21	(FileInputStream-class), 26
field, Descriptor-method	GetErrno,FileOutputStream-method
(Descriptor-class), 15	(FileOutputStream-class), 28
field-methods, 21	GetErrno-methods, 29
field_count (field_count-methods), 24	getExtension (Message-class), 32
field_count,Descriptor-method	getExtension, Message-method
(Descriptor-class), 15	(Message-class), 32
field_count-methods, 24	(
FieldDescriptor, 8, 15, 16, 21, 22, 31, 37,	has (has-methods), 29
38, 46	has,EnumDescriptor-method
FieldDescriptor-class, 22	(EnumDescriptor-class), 17
FileDescriptor, 12	has, Message-method (has-methods), 29
fileDescriptor, 25	has-methods, 29
fileDescriptor	has_default_value
(fileDescriptor-methods), 26	(FieldDescriptor-class), 22
fileDescriptor, Descriptor-method	has_default_value,FieldDescriptor-method
(fileDescriptor-methods), 26	(FieldDescriptor-class), 22
fileDescriptor, EnumDescriptor-method	has_default_value-methods
(fileDescriptor-methods). 26	(FieldDescriptor-class), 22

identical, Message, Message-method	merge, Message, Message-method
(Message-class), 32	(merge-methods), 32
<pre>input_type (MethodDescriptor-class), 35</pre>	merge-methods, 32
<pre>input_type,MethodDescriptor-method</pre>	Message, 3, 8, 10–12, 16, 17, 29, 30, 32, 38, 47
(MethodDescriptor-class), 35	Message-class, 32
<pre>input_type-methods</pre>	<pre>message_type (FieldDescriptor-class), 22</pre>
(MethodDescriptor-class), 35	<pre>message_type,FieldDescriptor-method</pre>
<pre>is_extension(is_extension-methods), 31</pre>	(FieldDescriptor-class), 22
is_extension,FieldDescriptor-method	<pre>message_type-methods</pre>
(FieldDescriptor-class), 22	(FieldDescriptor-class), 22
is_extension-methods, 31	<pre>method (ServiceDescriptor-class), 43</pre>
<pre>is_optional (FieldDescriptor-class), 22</pre>	method, ServiceDescriptor-method
is_optional,FieldDescriptor-method	(ServiceDescriptor-class), 43
(FieldDescriptor-class), 22	method-methods
is_optional-methods	(ServiceDescriptor-class), 43
(FieldDescriptor-class), 22	<pre>method_count (ServiceDescriptor-class),</pre>
<pre>is_repeated (FieldDescriptor-class), 22</pre>	43
<pre>is_repeated,FieldDescriptor-method</pre>	<pre>method_count,ServiceDescriptor-method</pre>
(FieldDescriptor-class), 22	(ServiceDescriptor-class), 43
is_repeated-methods	<pre>method_count-methods</pre>
(FieldDescriptor-class), 22	(ServiceDescriptor-class), 43
<pre>is_required (FieldDescriptor-class), 22</pre>	MethodDescriptor,43
is_required,FieldDescriptor-method	MethodDescriptor-class, 35
(FieldDescriptor-class), 22	
is_required-methods	name, 36
(FieldDescriptor-class), 22	name, Descriptor-method (name), 36
<pre>isInitialized(isInitialized-methods),</pre>	name, EnumDescriptor-method(name), 36
30	name, EnumValueDescriptor-method
isInitialized,Message-method	(EnumValueDescriptor-class), 19
(isInitialized-methods), 30	name, FieldDescriptor-method (name), 36
isInitialized-methods, 30	name,FileDescriptor-method
	(FileDescriptor-class), 25
label (label-methods), 31	name, MethodDescriptor-method (name), 36
label,FieldDescriptor-method	name, ServiceDescriptor-method (name), 36
(FieldDescriptor-class), 22	name-methods (name), 36
label-methods, 31	names, Descriptor-method
LABEL_OPTIONAL (label-methods), 31	(Descriptor-class), 15
LABEL_REPEATED (label-methods), 31	names, EnumDescriptor-method
LABEL_REQUIRED (label-methods), 31	(EnumDescriptor-class), 17
length, Descriptor-method	names, Message-method (Message-class), 32
(Descriptor-class), 15	nested_type (nested_type-methods), 36
length, EnumDescriptor-method	nested_type,Descriptor-method
(EnumDescriptor-class), 17	(Descriptor-class), 15
<pre>length, Message-method (Message-class),</pre>	nested_type-methods, 36
32	nested_type_count
length, ServiceDescriptor-method	(nested_type_count-methods), 36
(ServiceDescriptor-class), 43	nested_type_count,Descriptor-method
1	(Descriptor-class), 15
makeActiveBinding, 47	<pre>nested_type_count-methods, 36</pre>

new, Descriptor-method	ReadLittleEndian32-methods
(Descriptor-class), 15	(ZeroCopyInputStream-class), 48
Next (Next-methods), 37	ReadLittleEndian64
Next, ZeroCopyInputStream, missing-method	(ZeroCopyInputStream-class), 48
(ZeroCopyInputStream-class), 48	ReadLittleEndian64,ZeroCopyInputStream-method
Next,ZeroCopyOutputStream,raw-method	(ZeroCopyInputStream-class), 48
(ZeroCopyOutputStream-class),	ReadLittleEndian64-methods
49	(ZeroCopyInputStream-class), 48
Next-methods, 37	readProtoFiles,41
number (number-methods), 37	<pre>readProtoFiles2 (readProtoFiles), 41</pre>
number,EnumValueDescriptor-method	ReadRaw(ZeroCopyInputStream-class), 48
(EnumValueDescriptor-class), 19	${\tt ReadRaw,ZeroCopyInputStream,integer-method}$
number,FieldDescriptor-method	(ZeroCopyInputStream-class), 48
(FieldDescriptor-class), 22	${\sf ReadRaw, ZeroCopyInputStream, numeric-method}$
number-methods, 37	(ZeroCopyInputStream-class), 48
	ReadRaw-methods
<pre>output_type (MethodDescriptor-class), 35</pre>	(ZeroCopyInputStream-class), 48
output_type, MethodDescriptor-method	<pre>ReadString (ZeroCopyInputStream-class),</pre>
(MethodDescriptor-class), 35	48
output_type-methods	ReadString, ZeroCopyInputStream, integer-method
(MethodDescriptor-class), 35	(ZeroCopyInputStream-class), 48
(Hethousescriptor Class), 33	ReadString,ZeroCopyInputStream,numeric-method
D 15 17 24 20 42	(ZeroCopyInputStream-class), 48
P, 15, 17, 34, 38, 42	ReadString-methods
	(ZeroCopyInputStream-class), 48
read (read-methods), 38	ReadVarint32
read, Descriptor, ANY-method	(ZeroCopyInputStream-class), 48
(read-methods), 38	ReadVarint32,ZeroCopyInputStream-method
read,Descriptor,character-method	(ZeroCopyInputStream-class), 48
(read-methods), 38	ReadVarint32-methods
read,Descriptor,raw-method	(ZeroCopyInputStream-class), 48
(read-methods), 38	ReadVarint64
read-methods, 38	(ZeroCopyInputStream-class), 48
readASCII (readASCII-methods), 39	ReadVarint64, ZeroCopyInputStream-method
readASCII, Descriptor, ANY-method	(ZeroCopyInputStream-class), 48
(readASCII-methods), 39	ReadVarint64-methods
readASCII, Descriptor, character-method	(ZeroCopyInputStream-class), 48
(readASCII-methods), 39	resetDescriptorPool (readProtoFiles), 41
readASCII-methods, 39	RProtoBuf (RProtoBuf-package), 3
readJSON (readJSON-methods), 40	RProtoBuf-package, 3
readJSON, Descriptor, ANY-method	RPTOLOBUT-package, 5
(readJSON-methods), 40	serialize, <i>42</i> , <i>43</i>
readJSON, Descriptor, character-method	serialize, 42, 43 serialize, Message-method
(readJSON-methods), 40	(Message-class), 32
readJSON-methods, 40	
ReadLittleEndian32	serialize_pb, 42
	ServiceDescriptor-class, 43
(ZeroCopyInputStream-class), 48	set (set-methods), 44
ReadLittleEndian32,ZeroCopyInputStream-met	
(ZeroCopyInputStream-class),48	set-methods.44

SetCloseOnDelete	toString,FieldDescriptor-method
(SetCloseOnDelete-methods), 44	(FieldDescriptor-class), 22
SetCloseOnDelete,FileInputStream-method	toString,FileDescriptor-method
(FileInputStream-class), 26	(FileDescriptor-class), 25
SetCloseOnDelete,FileOutputStream-method	toString,Message-method
(FileOutputStream-class), 28	(Message-class), 32
SetCloseOnDelete-methods, 44	toString,MethodDescriptor-method
setExtension (Message-class), 32	(MethodDescriptor-class), 35
setExtension, Message-method	toString,ServiceDescriptor-method
(Message-class), 32	(ServiceDescriptor-class), 43
show, Descriptor-method	type (type-methods), 46
(Descriptor-class), 15	type,FieldDescriptor-method
show, EnumDescriptor-method	(FieldDescriptor-class), 22
(EnumDescriptor-class), 17	type-methods, 46
show, EnumValueDescriptor-method	TYPE_BOOL (type-methods), 46
(EnumValueDescriptor-class), 19	TYPE_BYTES (type-methods), 46
show,FieldDescriptor-method	TYPE_DOUBLE (type-methods), 46
(FieldDescriptor-class), 22	TYPE_ENUM(type-methods), 46
show,FileDescriptor-method	TYPE_FIXED32 (type-methods), 46
(FileDescriptor-class), 25	TYPE_FIXED64 (type-methods), 46
show, Message-method (Message-class), 32	TYPE_FLOAT (type-methods), 46
show, ServiceDescriptor-method	TYPE_GROUP (type-methods), 46
(ServiceDescriptor-class), 43	TYPE_INT32 (type-methods), 46
size (size-methods), 44	TYPE_INT64 (type-methods), 46
size, Message-method (size-methods), 44	TYPE_MESSAGE (type-methods), 46
size-methods, 44	TYPE_SFIXED32 (type-methods), 46
size<- (sizegets), 45	TYPE_SFIXED64 (type-methods), 46
size<-, Message-method (sizegets), 45	TYPE_SINT32 (type-methods), 46
size <methods (sizegets),="" 45<="" td=""><td>TYPE_SINT64 (type-methods), 46</td></methods>	TYPE_SINT64 (type-methods), 46
sizegets, 45	TYPE_STRING(type-methods), 46
Skip (Skip-methods), 45	TYPE_UINT32 (type-methods), 46
Skip,ZeroCopyInputStream-method	TYPE_UINT64 (type-methods), 46
(ZeroCopyInputStream-class), 48	
Skip-methods, 45	unserialize_pb (serialize_pb), 42
str, Message-method (Message-class), 32	update, Message-method (Message-class),
swap (swap-methods), 46	32
swap, Message-method (swap-methods), 46	value (FaumDeseminter aless) 17
swap-methods, 46	value (EnumDescriptor-class), 17
•	value, EnumDescriptor-method
toJSON (Message-class), 32	(EnumDescriptor-class), 17
toJSON, Message-method (Message-class),	value-methods (EnumDescriptor-class), 17 value_count (EnumDescriptor-class), 17
32	•
toString,Descriptor-method	value_count,EnumDescriptor-method
(Descriptor-class), 15	(EnumDescriptor-class), 17 value_count-methods
toString,EnumDescriptor-method	
(EnumDescriptor-class), 17	(EnumDescriptor-class), 17
toString,EnumValueDescriptor-method	with.Message, 47
(EnumValueDescriptor-class), 19	within.Message, 47 within.Message (with.Message), 47
(Endinatachesei ipioi etass), 1)	"1 :://11 : 1 : 0 : 0 : 0 : 0 : 0 : 0 : 0 : 0 :

```
WriteLittleEndian32
                                                                                                                  (ZeroCopyOutputStream-class),
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint32, ZeroCopyOutputStream, integer-method
Write Little Endian 32, Zero Copy Output Stream, integer-meth (\ref{CopyOutput} Stream-class), \\
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint32, ZeroCopyOutputStream, numeric-method
Write Little Endian 32, Zero Copy Output Stream, numeric-meth \ref{CopyOutputStream-class}), \\
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint32, ZeroCopyOutputStream, raw-method
\label{lem:writeLittleEndian32,ZeroCopyOutputStream,raw-method} (ZeroCopyOutputStream-class),
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint32-methods
                 49
                                                                                                                  (ZeroCopyOutputStream-class),
WriteLittleEndian32-methods
                                                                                                                  49
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint64
                                                                                                                  (ZeroCopyOutputStream-class),
WriteLittleEndian64
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint64, ZeroCopyOutputStream, integer-method
\label{thm:procopy0utputStream} Write Little Endian 64, Zero Copy0utput Stream-class), where the description of the procopy 
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint64, ZeroCopyOutputStream, numeric-method
\label{thm:procopy0utputStream} Write Little Endian 64, Zero Copy Output Stream-class), \\
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint64, ZeroCopyOutputStream, raw-method
                 49
                                                                                                                 (ZeroCopyOutputStream-class),
WriteLittleEndian64,ZeroCopyOutputStream,raw-method
                                                                                                                  49
                 (ZeroCopyOutputStream-class),
                                                                                                 WriteVarint64-methods
                                                                                                                  (ZeroCopyOutputStream-class),
WriteLittleEndian64-methods
                 (ZeroCopyOutputStream-class),
                                                                                                 ZeroCopyInputStream, 4, 5, 9, 13, 26, 27, 37
WriteRaw (ZeroCopyOutputStream-class),
                                                                                                 ZeroCopyInputStream-class, 48
                                                                                                 ZeroCopyOutputStream, 6, 14, 28
WriteRaw, ZeroCopyOutputStream, raw-method
                                                                                                 ZeroCopyOutputStream-class, 49
                 (ZeroCopyOutputStream-class),
WriteRaw-methods
                 (ZeroCopyOutputStream-class),
                 49
WriteString
                 (ZeroCopyOutputStream-class),
WriteString, ZeroCopyOutputStream, character-method
                 (ZeroCopyOutputStream-class),
                 49
WriteString-methods
                 (ZeroCopyOutputStream-class),
WriteVarint32
```