

**NAME**

Net::DBus::Binding::Iterator – Reading and writing message parameters

**SYNOPSIS**

Creating a new message

```
my $msg = new Net::DBus::Binding::Message::Signal;
my $iterator = $msg->iterator;

$iterator->append_boolean(1);
$iterator->append_byte(123);
```

Reading from a message

```
my $msg = ...get it from somewhere...
my $iter = $msg->iterator();

my $i = 0;
while ($iter->has_next()) {
    $iter->next();
    $i++;
    if ($i == 1) {
        my $val = $iter->get_boolean();
    } elsif ($i == 2) {
        my $val = $iter->get_byte();
    }
}
```

**DESCRIPTION**

Provides an iterator for reading or writing message fields. This module provides a Perl API to access the `dbus_message_iter_XXX` methods in the C API. The array and dictionary types are not yet supported, and there are bugs in the Quad support (ie it always returns -1!).

**METHODS**

`$res = $iter->has_next()`

Determines if there are any more fields in the message iterator to be read. Returns a positive value if there are more fields, zero otherwise.

`$success = $iter->next()`

Skips the iterator onto the next field in the message. Returns a positive value if the current field pointer was successfully advanced, zero otherwise.

`my $val = $iter->get_boolean()`

`$iter->append_boolean($val);`

Read or write a boolean value from/to the message iterator

`my $val = $iter->get_byte()`

`$iter->append_byte($val);`

Read or write a single byte value from/to the message iterator.

`my $val = $iter->get_string()`

`$iter->append_string($val);`

Read or write a UTF-8 string value from/to the message iterator

`my $val = $iter->get_object_path()`

`$iter->append_object_path($val);`

Read or write a UTF-8 string value, whose contents is a valid object path, from/to the message iterator

`my $val = $iter->get_signature()`

`$iter->append_signature($val);`

Read or write a UTF-8 string, whose contents is a valid type signature, value from/to the message iterator

```

my $val = $iter->get_int16()
$iter->append_int16($val);
    Read or write a signed 16 bit value from/to the message iterator

my $val = $iter->get_uint16()
$iter->append_uint16($val);
    Read or write an unsigned 16 bit value from/to the message iterator

my $val = $iter->get_int32()
$iter->append_int32($val);
    Read or write a signed 32 bit value from/to the message iterator

my $val = $iter->get_uint32()
$iter->append_uint32($val);
    Read or write an unsigned 32 bit value from/to the message iterator

my $val = $iter->get_int64()
$iter->append_int64($val);
    Read or write a signed 64 bit value from/to the message iterator. An error will be raised if this build of
    Perl does not support 64 bit integers

my $val = $iter->get_uint64()
$iter->append_uint64($val);
    Read or write an unsigned 64 bit value from/to the message iterator. An error will be raised if this
    build of Perl does not support 64 bit integers

my $val = $iter->get_double()
$iter->append_double($val);
    Read or write a double precision floating point value from/to the message iterator

my $val = $iter->get_unix_fd()
$iter->append_unix_fd($val);
    Read or write a unix_fd value from/to the message iterator

my $value = $iter->get()
my $type = $iter->get($type);
    Get the current value pointed to by this iterator. If the optional $type parameter is supplied, the wire
    type will be compared with the desired type & a warning output if their differ. The $type value must
    be one of the Net::DBus::Binding::Message::TYPE* constants.

my $hashref = $iter->get_dict()
    If the iterator currently points to a dictionary value, unmarshalls and returns the value as a hash
    reference.

my $hashref = $iter->get_array()
    If the iterator currently points to an array value, unmarshalls and returns the value as a array reference.

my $hashref = $iter->get_variant()
    If the iterator currently points to a variant value, unmarshalls and returns the value contained in the
    variant.

my $hashref = $iter->get_struct()
    If the iterator currently points to an struct value, unmarshalls and returns the value as a array reference.
    The values in the array correspond to members of the struct.

$iter->append($value)
$iter->append($value, $type)
    Appends a value to the message associated with this iterator. The value is marshalled into wire format,
    according to the following rules.

    If the $value is an instance of Net::DBus::Binding::Value, the embedded data type is used.

    If the $type parameter is supplied, that is taken to represent the data type. The type must be one of

```

the `Net::DBus::Binding::Message::TYPE_*` constants.

Otherwise, the data type is chosen to be a string, dict or array according to the perl data types `SCALAR`, `HASH` or `ARRAY`.

`my $type = $iter->guess_type($value)`

Make a best guess at the on the wire data type to use for marshalling `$value`. If the value is a hash reference, the dictionary type is returned; if the value is an array reference the array type is returned; otherwise the string type is returned.

`my $sig = $iter->format_signature($type)`

Given a data type representation, construct a corresponding signature string

`$iter->append_array($value, $type)`

Append an array of values to the message. The `$value` parameter must be an array reference, whose elements all have the same data type specified by the `$type` parameter.

`$iter->append_struct($value, $type)`

Append a struct to the message. The `$value` parameter must be an array reference, whose elements correspond to members of the structure. The `$type` parameter encodes the type of each member of the struct.

`$iter->append_dict($value, $type)`

Append a dictionary to the message. The `$value` parameter must be an hash reference. The `$type` parameter encodes the type of the key and value of the hash.

`$iter->append_variant($value)`

Append a value to the message, encoded as a variant type. The `$value` can be of any type, however, the variant will be encoded as either a string, dictionary or array according to the rules of the `guess_type` method.

`my $type = $iter->get_arg_type`

Retrieves the type code of the value pointing to by this iterator. The returned code will correspond to one of the constants `Net::DBus::Binding::Message::TYPE_*`

`my $type = $iter->get_element_type`

If the iterator points to an array, retrieves the type code of array elements. The returned code will correspond to one of the constants `Net::DBus::Binding::Message::TYPE_*`

## AUTHOR

Daniel P. Berrange

## COPYRIGHT

Copyright (C) 2004–2011 Daniel P. Berrange

## SEE ALSO

`Net::DBus::Binding::Message`