exosip2ctypes Documentation

Release 1.2.4.post4+g911613c

Liu Xue Yan

Apr 19, 2018, 7:18:28 AM

Contents

1 Summaries **3**
- 1.1 exosip2ctypes.call . . . 3
- 1.2 exosip2ctypes.context . . . 4
- 1.3 exosip2ctypes.error . . . 4
- 1.4 exosip2ctypes.event . . . 5
- 1.5 exosip2ctypes.message . . . 5
- 1.6 exosip2ctypes.register . . . 5
- 1.7 exosip2ctypes.sdp . . . 5
- 1.8 exosip2ctypes.utils . . . 6
- 1.9 exosip2ctypes.version . . . 6

2 Contents **7**
- 2.1 exosip2ctypes . . . 7

3 Indices and tables **25**

Python Module Index **27**

Build Time Apr 19, 2018, 7:18:28 AM

Version 1.2

Release 1.2.4.post4+g911613c

CHAPTER 1

Summaries

exosip2ctypes.call	eXosip call API
exosip2ctypes.context	eXosip2 context API
exosip2ctypes.error	Error definitions
exosip2ctypes.event	eXosip2 event API
exosip2ctypes.message	osip and eXosip messages
exosip2ctypes.register	eXosip2 REGISTER and Registration Management
exosip2ctypes.sdp	eXosip2 SDP helper API.
exosip2ctypes.utils	Some helper functions
exosip2ctypes.version	version infomation

1.1 exosip2ctypes.call

eXosip call API

This file provide the API needed to control calls. You can use it to:

- build initial invite.
- send initial invite.
- build request within the call.
- send request within the call.

This API can be used to build the following messages: INVITE, INFO, OPTIONS, REFER, UPDATE, NOTIFY

Classes

Ack(context, did)	default ACK for a 200ok received.
Answer(context, tid, status)	default Answer for request.

Continued on next page

Table 1.2 – continued from previous page

InitInvite(context, to_url, from_url[, ...])	default INVITE message for a create call.

1.2 exosip2ctypes.context

eXosip2 context API

Classes

BaseContext	
Context([event_callback])	Allocate and Initiate an eXosip context.
ContextLock(context)	A helper class for eXosip Context lock

1.3 exosip2ctypes.error

Error definitions

see *osip/include/osipparser2/osip_port.h*

Functions

raise_if_osip_error(error_code[, message])	raise an *OsipError* exception if *error_code* is not OSIP_SUCCESS

Exceptions

MallocError	Failed to allocate an *eXosip* context.
OsipApiNotInitialized	
OsipBadParameter	
OsipDiskFull	
OsipError	Base *Osip* error Exception Class
OsipFileNotExists	
OsipNoCommonCodec	
OsipNoMem	
OsipNoNetwork	
OsipNoRights	
OsipNotFound	
OsipPortBusy	
OsipSyntaxError	
OsipTimeout	
OsipTooMuchCall	
OsipUndefinedError	
OsipUnknownError	Osip error, but don't known the error code
OsipUnknownHost	
OsipWrongFormat	

Continued on next page

Table 1.5 – continued from previous page

OsipWrongState	

1.4 exosip2ctypes.event

eXosip2 event API

see: http://www.antisip.com/doc/exosip2/group__eXosip2__event.html

Classes

Event(ptr, context)	Class for event description
EventType	Enumeration of event types

1.5 exosip2ctypes.message

osip and eXosip messages

Message classes of osip and eXosip

Classes

ExosipMessage(ptr, context)	class for eXosip2 message API
OsipMessage(ptr)	class for osip2 message API

1.6 exosip2ctypes.register

eXosip2 REGISTER and Registration Management

Classes

InitialRegister(context, from_, proxy[, ...])	initial REGISTER request.
Register(context, rid[, expires])	REGISTER request for an existing registration.

1.7 exosip2ctypes.sdp

eXosip2 SDP helper API.

Classes

SdpMessage(ptr)	SDP body

1.8 exosip2ctypes.utils

Some helper functions

Functions

to_bytes(s[, encoding])	Convert to *bytes* string.
to_str(s[, encoding])	Convert to *str* string.
to_unicode(s[, encoding])	Convert to *unicode* string.

Classes

LoggerMixin	Mixin Class provide a `logger` property

1.9 exosip2ctypes.version

version infomation

Functions

get_library_version()	**return** eXosip library (C library) version string

CHAPTER 2

Contents

2.1 exosip2ctypes

2.1.1 exosip2ctypes package

Submodules

exosip2ctypes.call module

eXosip call API

This file provide the API needed to control calls. You can use it to:

- build initial invite.
- send initial invite.
- build request within the call.
- send request within the call.

This API can be used to build the following messages: INVITE, INFO, OPTIONS, REFER, UPDATE, NOTIFY

class exosip2ctypes.call.**InitInvite**(*context*, *to_url*, *from_url*, *route=None*, *subject=None*)
Bases: *exosip2ctypes.message.ExosipMessage*

default INVITE message for a create call.

Build a default INVITE message for a create call.

Parameters

- **Context context** (Context) – Context object contains the *eXosip_t* instance.
- **to_url** (*str*) – SIP url for callee.
- **from_url** (*str*) – SIP url for caller.

- **route** (*str*) – Route header for INVITE. (optional)
- **subject** (*str*) – Subject for the call.

from_url
SIP url for caller.

Return type str

route
Route header for INVITE. (optional)

Return type str

subject
Subject for the call.

Return type str

to_url
SIP url for callee.

Return type str

class exosip2ctypes.call.**Ack**(*context*, *did*)
Bases: *exosip2ctypes.message.ExosipMessage*

default ACK for a 200ok received.

Build a default ACK for a 200ok received.

Parameters

- **context** (Context) – eXosip instance.
- **did** (*int*) – dialog id of call.

did

Returns dialog id of call.

Return type int

send()
Send the ACK for the 200ok received.

class exosip2ctypes.call.**Answer**(*context*, *tid*, *status*)
Bases: *exosip2ctypes.message.ExosipMessage*

default Answer for request.

Build default Answer for request.

Parameters

- **context** (Context) – eXosip instance.
- **tid** (*int*) – id of transaction to answer.
- **status** (*int*) – Status code to use.

send()
Send Answer for invite.

status

Returns Status code to use.

Return type int

tid

Returns id of transaction to answer.

Return type int

exosip2ctypes.context module

eXosip2 context API

class exosip2ctypes.context.**Context**(*event_callback=None*)
Bases: exosip2ctypes.context.BaseContext, *exosip2ctypes.utils.LoggerMixin*

Allocate and Initiate an eXosip context.

Parameters **event_callback** (*callable*) – Event callback.

It's a callback function or any other callable object. You can avoid the parameter in constructor, and set *event_callback* later.

The callback is like:

```
def event_callback(context, event):
    # do some thing...
    pass
```

It has two parameters:

- *Context* : eXosip context on which the event happened.
- Event : The event happened.

add_authentication_info(*user_name*, *user_id*, *password*, *realm*)
Add authentication credentials.

Parameters

- **user_name** (*str*) – username
- **user_id** (*str*) – login (usually equals the username)
- **password** (*str*) – password
- **realm** (*str*) – realm within which credentials apply, or *None* to apply credentials to unrecognized realms

These are used when an outgoing request comes back with an authorization required response.

automatic_action()
Initiate some automatic actions:

- Retry with credentials upon reception of 401/407.
- Retry with higher Sessions-Expires upon reception of 422.
- Refresh REGISTER and SUBSCRIBE before the expiration delay.
- Retry with Contact header upon reception of 3xx request.
- Send automatic UPDATE for session-timer feature.

build_message(*message_class*, **args*, ***kwargs*)
Build a default message for a eXosip transaction/dialog/call

Parameters

- **message_class** (type *(ExosipMessage)*) – Message Class
- **args** – Sequent arguments passed to the class constructor
- **kwargs** – Named arguments passed to the class constructor

Returns New built message object

Return type *ExosipMessage*

call_send_ack(*did=None*, *ack=None*)
Send the ACK for the 200ok received.

Parameters

- **did** (*int*) – dialog id of call.
- **ack** (call.Ack) – SIP ACK message to send. *did* won't be used if specified.

call_send_answer(*tid=None*, *status=None*, *answer=None*)
Send Answer for invite.

Parameters

- **tid** (*int*) – id of transaction to answer.
- **status** (*int*) – response status if *answer* is NULL. (not allowed for 2XX)
- **answer** (call.Answer) – The sip answer to send. *tid* and *status* won't be used if specified.

call_send_init_invite(*invite*)
Initiate a call.

Parameters **invite** (call.InitInvite) – SIP INVITE message to send.

Returns CID - unique id for SIP calls (but multiple dialogs!)

Return type int

> **Attention:** returned *call id* is an integer, which different from SIP message's *Call-Id* header

call_terminate(*cid*, *did=0*)
Terminate a call. send CANCEL, BYE or 603 Decline.

Parameters

- **cid** (*int*) – call id of call.
- **did** (*int*) – dialog id of call.

event_callback
Set event callback

Return type callable

> **Attention:** Event callback is invoked in a concurrent.futures.Executor

event_wait(*s*, *ms*)
Wait for an eXosip event.

Parameters

- **s** (*int*) – timeout value (seconds).
- **ms** (*int*) – timeout value (seconds).

Returns event or *None* if no event

Return type *Event*

get_event_callback()
Set event callback

Return type callable

Attention: Event callback is invoked in a `concurrent.futures.Executor`

is_running
Is the context's main loop running

Return type bool

- It will be *True* after calling *start()* or *run()*
- It will be *False* after calling *stop()*

listen_on_address(*address=None*, *transport=17*, *port=5060*, *family=<AddressFamily.AF_INET: 2>*, *secure=False*)
Listen on a specified socket.

Parameters

- **address** (*str*) – the address to bind (*NULL* for all interface)
- **transport** (*int*) – `socket.IPPROTO_UDP` for udp. (soon to come: TCP/TLS?)
- **port** (*int*) – the listening port. (0 for random port)
- **family** (*int*) – the IP family (`socket.AF_INET` or `socket.AF_INET6`).
- **secure** (*bool*) – *False* for UDP or TCP, *True* for TLS (with TCP).

lock
eXosip Context lock.

Return type *ContextLock*

lock_acquire()
Lock the eXtented oSIP library.

lock_release()
UnLock the eXtented oSIP library.

locked
True if the context has been acquired by some thread, False if not.

Return type bool

masquerade_contact(*public_address=None*, *port=0*)
This method is used to replace contact address with the public address of your NAT. The ip address should be retrieved manually (fixed IP address) or with STUN. This address will only be used when the remote correspondent appears to be on an DIFFERENT LAN.

If set to *None*, then the local ip address will be guessed automatically (returns to default mode).

Parameters

- **public_address** (*str*) – the ip address.
- **port** (*int*) – the port for masquerading.

Note: It's advised to call the method after *listen_on_address()*

---

ptr
C Pointer to the context's *eXosip_t* C structure

Return type ctypes.c_void_p

quit()
Release resource used by the eXtented oSIP library.

remove_authentication_info(*user_name*, *realm*)
Remove authentication credentials.

Parameters

- **user_name** (*str*) – user name
- **realm** (*str*) – realm must be exact same arg as for *add_authentication_info()*

run(*s=0*, *ms=50*, *event_executor=None*, *timeout=None*)
Start the main loop for the context in a create thread, and then wait until the thread terminates.

Parameters

- **s** (*int*) – timeout value (seconds). Passed to *event_wait()* in the main loop.
- **ms** (*int*) – timeout value (seconds). Passed to *event_wait()* in the main loop.
- **event_executor** (*concurrent.futures.Executor*) – see the same parameter in *start()*.
- **timeout** (*float*) – When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof)

This method **blocks**, it equals:

```
trd = context.start(s, ms)
trd.join(timeout)
```

set_event_callback(*val*)
Set event callback

Parameters **val** (*callable*) – callback function

Attention: Event callback is invoked in a concurrent.futures.Executor

start(*s=0*, *ms=50*, *event_executor=None*)
Start the main loop for the context in a create thread, and then return.

Parameters

- **s** (*int*) – timeout value (seconds). Passed to *event_wait()* in the main loop.
- **ms** (*int*) – timeout value (seconds). Passed to *event_wait()* in the main loop.

- **event_executor** (*concurrent.futures.Executor*) – Event executor instance. Events will be fired in it. Default is a concurrent.futures.ThreadPoolExecutor instance

Returns New created event loop thread.

Return type threading.Thread

This method returns soon after the main loop thread started, so it **does not block**.

Equal to set *is_running* to *True*

stop()
Stop the context's main loop thread and return after the thread stopped.

Equal to set *is_running* to *False*

user_agent
Context's user agent string

Return type str

class exosip2ctypes.context.**ContextLock**(*context*)
Bases: object

A helper class for eXosip Context lock

This class wraps eXosip's native context lock function, which is invoked in *Context.lock_acquire()* and *Context.lock_release()*. You can call theses methods directly, or use *Context.lock*.

with statement is supported.

eg:

```
context.lock.acquire()
try:
    do_something()
    # ...
finally:
    context.lock.release()
```

or:

```
with context.lock:
    do_something()
    # ...
```

Danger: Do **NOT** construct the class yourself, use *Context.lock*.

Parameters **context** (Context) – Context which the lock is for

acquire()
lock the context

locked()
Return the status of the lock: True if it has been acquired by some thread, False if not.

Return type bool

release()
unlock the context

exosip2ctypes.error module

Error definitions

see *osip/include/osipparser2/osip_port.h*

exosip2ctypes.error.**raise_if_osip_error**(*error_code*, *message=None*)
raise an *OsipError* exception if *error_code* is not OSIP_SUCCESS

Parameters

- **error_code** (*int*) – *osip* and/or *exosip* C functions return code
- **message** (*str*) – Exception message passed to the raised *OsipError*

Raises ***OsipError*** – Throw exception if *error_code* is smaller than zero(OSIP_SUCCESS)

Can be used to check osip2/eXosip2 API function integer return value

> **Attention:** All error codes in *osip* are smaller than 0

exception exosip2ctypes.error.**MallocError**
Bases: RuntimeError

Failed to allocate an *eXosip* context.

exception exosip2ctypes.error.**OsipError**
Bases: RuntimeError

Base *Osip* error Exception Class

exception exosip2ctypes.error.**OsipUnknownError**
Bases: *exosip2ctypes.error.OsipError*

Osip error, but don't known the error code

exception exosip2ctypes.error.**OsipUndefinedError**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipBadParameter**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipWrongState**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipNoMem**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipSyntaxError**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipNotFound**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipApiNotInitialized**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipNoNetwork**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipPortBusy**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipUnknownHost**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipDiskFull**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipNoRights**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipFileNotExists**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipTimeout**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipTooMuchCall**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipWrongFormat**
Bases: *exosip2ctypes.error.OsipError*

exception exosip2ctypes.error.**OsipNoCommonCodec**
Bases: *exosip2ctypes.error.OsipError*

exosip2ctypes.event module

eXosip2 event API

see: http://www.antisip.com/doc/exosip2/group__eXosip2__event.html

class exosip2ctypes.event.**Event**(*ptr*, *context*)
Bases: object

Class for event description

Parameters

- **ptr** (*ctypes.c_void_p*) – *struct eXosip_event_t *ptr*
- **context** (Context) – eXosip context

ack

Returns ack within current transaction

Return type *OsipMessage*

cid

Returns unique id for SIP calls (but multiple dialogs!)

Return type int

did

Returns unique id for SIP dialogs

Return type int

dispose()
Free resource in an eXosip event.

> **Danger:** After called this method, *struct eXosip_event_t* of the object is freed, following attributes will be *None*:
>
> - `request`
> - `response`
> - `ack`
>
> It is invoked in class destructor. Don't call it yourself, let Python runtime's GC dispose the structure.

disposed
Is resource in an eXosip event disposed

Return type bool

nid

Returns unique id for incoming subscriptions

Return type int

request

Returns request within current transaction

Return type *ExosipMessage*

response

Returns last response within current transaction

Return type *ExosipMessage*

rid

Returns unique id for registration

Return type int

sid
unique id for outgoing subscriptions

Return type int

ss_reason

Returns current Reason status for subscription

Return type int

ss_status

Returns current Subscription-State for subscription

Return type int

textinfo

Returns text description of event

Return type str

tid

Returns unique id for transactions (to be used for answers)

Return type int

type
type of the event

Return type *EventType*

class exosip2ctypes.event.**EventType**
Bases: enum.IntEnum

Enumeration of event types

call_ack = 12

call_answered = 7

call_cancelled = 13

call_closed = 21

call_globalfailure = 11

call_invite = 2

call_message_answered = 16

call_message_globalfailure = 20

call_message_new = 14

call_message_proceeding = 15

call_message_redirected = 17

call_message_requestfailure = 18

call_message_serverfailure = 19

call_noanswer = 4

call_proceeding = 5

call_redirected = 8

call_reinvite = 3

call_released = 22

call_requestfailure = 9

call_ringing = 6

call_serverfailure = 10

in_subscription_new = 38

message_answered = 25

message_globalfailure = 29

message_new = 23

message_proceeding = 24

message_redirected = 26

message_requestfailure = 27

message_serverfailure = 28

notification_answered = 41

notification_globalfailure = 45

```
notification_noanswer = 39
notification_proceeding = 40
notification_redirected = 42
notification_requestfailure = 43
notification_serverfailure = 44
registration_failure = 1
registration_success = 0
subscription_answered = 32
subscription_globalfailure = 36
subscription_noanswer = 30
subscription_notify = 37
subscription_proceeding = 31
subscription_redirected = 33
subscription_requestfailure = 34
subscription_serverfailure = 35
```

exosip2ctypes.message module

osip and eXosip messages

Message classes of osip and eXosip

class exosip2ctypes.message.**OsipMessage**(*ptr*)

Bases: object

class for osip2 message API

Parameters **ptr** (*ctypes.c_void_p*) – Pointer to the *osip_message_t* structure in C library

add_allow(*val*)

Set the Allow header.

Parameters **val** (*str*) – The string describing the element.

> **Attention:** This method will **ADD** a create *ALLOW* header

add_body(*val*)

Fill the body of message.

Parameters **val** (*str*) – Body string.

> **Attention:** This method will **ADD** a create body

add_contact(*val*)

Set the Contact header.

Parameters **val** (*str*) – The string describing the element.

> **Attention:** This method will **ADD** a create *Contact* header

add_header(*name*, *value*)
Allocate and Add an "unknown" header (not defined in oSIP).

Parameters

- **name** (*str*) – The token name.
- **value** (*str*) – The token value.

> **Attention:** This method will **ADD** a create header

allows
Get Allow header list.

Return type list

bodies
Get body header list.

Return type list

call_id
Call-id header.

Return type str

contacts
Get Contact header list.

Return type list

content_length
Content-length header.

Return type int

content_type
Content Type string of the SIP message

Return type str

from_
From header

Return type str

get_headers(*name*)
Find "unknown" header's list. (not defined in oSIP)

Parameters **name** (*str*) – The name of the header to find.

Returns Header's value string list.

Return type list

ptr
Pointer to the *osip_message_t* C Structure

Return type ctypes.c_void_p

to
To header.

Return type str

class exosip2ctypes.message.**ExosipMessage**(*ptr*, *context*)
Bases: *exosip2ctypes.message.OsipMessage*

class for eXosip2 message API

Parameters

- **ptr** (*ctypes.c_void_p*) – Pointer to the *osip_message_t* structure in C library
- **context** (Context) – eXosip context

Danger: Do **NOT** con/destruct the class yourself unless you known what you are doing.

Attention: In eXosip2, messages are managed inside the library, so we should **NOT** free *OsipMessage* object manually.

context
eXosip context of the message

Return type *Context*

send()

exosip2ctypes.register module

eXosip2 REGISTER and Registration Management

class exosip2ctypes.register.**InitialRegister**(*context*, *from_*, *proxy*, *contact=None*, *expires=3600*)
Bases: *exosip2ctypes.message.ExosipMessage*

initial REGISTER request.

To start a registration, you need to build a default REGISTER request by providing several mandatory headers.

You can start as many registration you want even in one eXosip_t context.

The created instance has a registration identifier (*rid*) that you can use to update your registration. In future events about this registration, you'll see that registration identifier when applicable.

Note: You should let eXosip2 manage contact headers alone. The setup you have provided will generate various behavior, all being controlled by eXosip2. See the "NAT and Contact header" section in setup documentation.

Build initial REGISTER request.

Parameters

- **context** – eXosip_t instance.
- **from** (*str*) – SIP url for caller.
- **proxy** (*str*) – Proxy used for registration.

- **contact** (*str*) – Contact address. (optional)
- **expires** (*int*) – The expires value for registration.

contract

Returns Contact address. (optional)

Return type str

expires

Returns The expires value for registration.

Return type int

proxy
:return:Proxy used for registration. :rtype: str

remove()
Remove existing registration without sending REGISTER.

If you need to delete a context without sending a REGISTER with expires 0, you can use this method to release memory.

rid

Returns registration identifier

Return type int

send()
Send this REGISTER request

class exosip2ctypes.register.**Register**(*context*, *rid*, *expires=3600*)
Bases: *exosip2ctypes.message.ExosipMessage*

REGISTER request for an existing registration.

Use this class if you need to update a registration, You just need to reuse the registration identifier (*InitialRegister.rid*)

Note: An UAC should delete its registration on the SIP server when terminating. To do so, you have to send a REGISTER request with the expires header set to value "0".

Build a new REGISTER request for an existing registration.

Parameters

- **context** (Context) – eXosip_t instance.
- **rid** (*int*) – A unique identifier for the registration context
- **expires** (*int*) – The expires value for registration.

expires

Returns The expires value for registration.

Return type int

remove()
Remove existing registration without sending REGISTER.

If you need to delete a context without sending a REGISTER with expires 0, you can use this method to release memory.

rid

Returns A unique identifier for the registration context

Return type int

send()
Send this REGISTER request

exosip2ctypes.sdp module

eXosip2 SDP helper API.

class exosip2ctypes.sdp.**SdpMessage**(*ptr*)
Bases: object

SDP body

Parameters **ptr** (*ctypes.c_void_p*) – *sdp_message_t**

ptr

exosip2ctypes.utils module

Some helper functions

exosip2ctypes.utils.**to_bytes**(*s*, *encoding='utf-8'*)
Convert to *bytes* string.

Parameters

- **s** – String to convert.
- **encoding** (*str*) – Encoding codec.

Returns *bytes* string, it's *bytes* or *str* in Python 2.x, *bytes* in Python 3.x.

Return type bytes

- In Python 2, convert *s* to *bytes* if it's *unicode*.
- In Python 2, return original *s* if it's not *unicode*.
- In Python 2, it equals to *to_str()*.
- In Python 3, convert *s* to *bytes* if it's *unicode* or *str*.
- In Python 3, return original *s* if it's neither *unicode* nor *str*.

exosip2ctypes.utils.**to_str**(*s*, *encoding='utf-8'*)
Convert to *str* string.

Parameters

- **s** – String to convert.
- **encoding** (*str*) – Decoding codec.

Returns *str* string, it's *bytes* in Python 2.x, *unicode* or *str* in Python 3.x.

Return type str

- In Python 2, convert *s* to *str* if it's *unicode*.

- In Python 2, return original *s* if it's not *unicode*.
- In Python 2, it equals to *to_bytes()*.
- In Python 3, convert *s* to *str* if it's *bytes*.
- In Python 3, return original *s* if it's not *bytes*.
- In Python 3, it equals to *to_unicode()*.

exosip2ctypes.utils.**to_unicode**(*s*, *encoding='utf-8'*)
Convert to *unicode* string.

Parameters

- **s** – String to convert.
- **encoding** (*str*) – Encoding codec.

Returns *unicode* string, it's *unicode* in Python 2.x, *str* or *unicode* in Python 3.x.

Return type unicode

- In Python 2, convert *s* to *unicode* if it's *str* or *bytes*.
- In Python 2, return original *s* if it's neither *str* or *bytes*.
- In Python 3, convert *s* to *str* or *unicode* if it's *bytes*.
- In Python 3, return original *s* if it's not *bytes*.
- In Python 3, it equals to *to_str()*.

class exosip2ctypes.utils.**LoggerMixin**
Bases: object

Mixin Class provide a *logger* property

logger
logger instance.

Return type logging.Logger

logger name format is *ModuleName.ClassName*

exosip2ctypes.version module

version infomation

exosip2ctypes.version.**get_library_version**()

Returns eXosip library (C library) version string

Return type str

Module contents

eXosip API

author Liu Xue Yan

email realtanbro@gmail.com

eXosip is a high layer library for rfc3261: the SIP protocol. It offers a simple API to make it easy to use. eXosip2 offers great flexibility for implementing SIP endpoint like:

- SIP User-Agents
- SIP Voicemail or IVR
- SIP B2BUA
- any SIP server acting as an endpoint (music server...)

If you need to implement proxy or complex SIP applications, you should consider using osip instead.

Here are the eXosip capabilities:

- REGISTER to rpc_handler registration.
- INVITE/BYE to start/stop VoIP sessions.
- INFO to send DTMF within a VoIP sessions.
- OPTIONS to simulate VoIP sessions.
- re-INVITE to modify VoIP sessions
- REFER/NOTIFY to transfer calls.
- MESSAGE to send Instant Message.
- SUBSCRIBE/NOTIFY to rpc_handler presence capabilities.
- any other request to rpc_handler what you want!

Constants

`exosip2ctypes.`**`DLL_NAME`**
Default so/dll name, value is `eXosip2`

Functions

`exosip2ctypes.`**`initialize`**(*path: str=None*) → None:
Load *libeXosip2* into this Python library

Parameters **`path`** (*str*) – *libeXosip2* SO/DLL path, *default* is *None*. When *None* or empty string, the function will try to find and load so/dll by *DLL_NAME*

Raises **`RuntimeError`** – When failed loading so/dll

> **Attention:** You **MUST** call this function **FIRST** to initialize *libeXosip2*, before any other actions!

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

exosip2ctypes, 23
exosip2ctypes.call, 3
exosip2ctypes.context, 4
exosip2ctypes.error, 4
exosip2ctypes.event, 5
exosip2ctypes.message, 5
exosip2ctypes.register, 5
exosip2ctypes.sdp, 5
exosip2ctypes.utils, 6
exosip2ctypes.version, 6

Index

A

Ack (class in exosip2ctypes.call), 8
ack (exosip2ctypes.event.Event attribute), 15
acquire() (exosip2ctypes.context.ContextLock method), 13
add_allow() (exosip2ctypes.message.OsipMessage method), 18
add_authentication_info() (exosip2ctypes.context.Context method), 9
add_body() (exosip2ctypes.message.OsipMessage method), 18
add_contact() (exosip2ctypes.message.OsipMessage method), 18
add_header() (exosip2ctypes.message.OsipMessage method), 19
allows (exosip2ctypes.message.OsipMessage attribute), 19
Answer (class in exosip2ctypes.call), 8
automatic_action() (exosip2ctypes.context.Context method), 9

B

bodies (exosip2ctypes.message.OsipMessage attribute), 19
build_message() (exosip2ctypes.context.Context method), 9

C

call_ack (exosip2ctypes.event.EventType attribute), 17
call_answered (exosip2ctypes.event.EventType attribute), 17
call_cancelled (exosip2ctypes.event.EventType attribute), 17
call_closed (exosip2ctypes.event.EventType attribute), 17
call_globalfailure (exosip2ctypes.event.EventType attribute), 17
call_id (exosip2ctypes.message.OsipMessage attribute), 19
call_invite (exosip2ctypes.event.EventType attribute), 17
call_message_answered (exosip2ctypes.event.EventType attribute), 17
call_message_globalfailure (exosip2ctypes.event.EventType attribute), 17
call_message_new (exosip2ctypes.event.EventType attribute), 17
call_message_proceeding (exosip2ctypes.event.EventType attribute), 17
call_message_redirected (exosip2ctypes.event.EventType attribute), 17
call_message_requestfailure (exosip2ctypes.event.EventType attribute), 17
call_message_serverfailure (exosip2ctypes.event.EventType attribute), 17
call_noanswer (exosip2ctypes.event.EventType attribute), 17
call_proceeding (exosip2ctypes.event.EventType attribute), 17
call_redirected (exosip2ctypes.event.EventType attribute), 17
call_reinvite (exosip2ctypes.event.EventType attribute), 17
call_released (exosip2ctypes.event.EventType attribute), 17
call_requestfailure (exosip2ctypes.event.EventType attribute), 17
call_ringing (exosip2ctypes.event.EventType attribute), 17
call_send_ack() (exosip2ctypes.context.Context method), 10
call_send_answer() (exosip2ctypes.context.Context method), 10
call_send_init_invite() (exosip2ctypes.context.Context method), 10
call_serverfailure (exosip2ctypes.event.EventType attribute), 17
call_terminate() (exosip2ctypes.context.Context method), 10
cid (exosip2ctypes.event.Event attribute), 15
contacts (exosip2ctypes.message.OsipMessage attribute),

19
content_length (exosip2ctypes.message.OsipMessage attribute), 19
content_type (exosip2ctypes.message.OsipMessage attribute), 19
Context (class in exosip2ctypes.context), 9
context (exosip2ctypes.message.ExosipMessage attribute), 20
ContextLock (class in exosip2ctypes.context), 13
contract (exosip2ctypes.register.InitialRegister attribute), 21

D

did (exosip2ctypes.call.Ack attribute), 8
did (exosip2ctypes.event.Event attribute), 15
dispose() (exosip2ctypes.event.Event method), 15
disposed (exosip2ctypes.event.Event attribute), 16
DLL_NAME (in module exosip2ctypes), 24

E

Event (class in exosip2ctypes.event), 15
event_callback (exosip2ctypes.context.Context attribute), 10
event_wait() (exosip2ctypes.context.Context method), 10
EventType (class in exosip2ctypes.event), 17
exosip2ctypes (module), 23
exosip2ctypes.call (module), 3, 7
exosip2ctypes.context (module), 4, 9
exosip2ctypes.error (module), 4, 14
exosip2ctypes.event (module), 5, 15
exosip2ctypes.message (module), 5, 18
exosip2ctypes.register (module), 5, 20
exosip2ctypes.sdp (module), 5, 22
exosip2ctypes.utils (module), 6, 22
exosip2ctypes.version (module), 6, 23
ExosipMessage (class in exosip2ctypes.message), 20
expires (exosip2ctypes.register.InitialRegister attribute), 21
expires (exosip2ctypes.register.Register attribute), 21

F

from_ (exosip2ctypes.message.OsipMessage attribute), 19
from_url (exosip2ctypes.call.InitInvite attribute), 8

G

get_event_callback() (exosip2ctypes.context.Context method), 11
get_headers() (exosip2ctypes.message.OsipMessage method), 19
get_library_version() (in module exosip2ctypes.version), 23

I

in_subscription_new (exosip2ctypes.event.EventType attribute), 17
initialize() (in module exosip2ctypes), 24
InitialRegister (class in exosip2ctypes.register), 20
InitInvite (class in exosip2ctypes.call), 7
is_running (exosip2ctypes.context.Context attribute), 11

L

listen_on_address() (exosip2ctypes.context.Context method), 11
lock (exosip2ctypes.context.Context attribute), 11
lock_acquire() (exosip2ctypes.context.Context method), 11
lock_release() (exosip2ctypes.context.Context method), 11
locked (exosip2ctypes.context.Context attribute), 11
locked() (exosip2ctypes.context.ContextLock method), 13
logger (exosip2ctypes.utils.LoggerMixin attribute), 23
LoggerMixin (class in exosip2ctypes.utils), 23

M

MallocError, 14
masquerade_contact() (exosip2ctypes.context.Context method), 11
message_answered (exosip2ctypes.event.EventType attribute), 17
message_globalfailure (exosip2ctypes.event.EventType attribute), 17
message_new (exosip2ctypes.event.EventType attribute), 17
message_proceeding (exosip2ctypes.event.EventType attribute), 17
message_redirected (exosip2ctypes.event.EventType attribute), 17
message_requestfailure (exosip2ctypes.event.EventType attribute), 17
message_serverfailure (exosip2ctypes.event.EventType attribute), 17

N

nid (exosip2ctypes.event.Event attribute), 16
notification_answered (exosip2ctypes.event.EventType attribute), 17
notification_globalfailure (exosip2ctypes.event.EventType attribute), 17
notification_noanswer (exosip2ctypes.event.EventType attribute), 17
notification_proceeding (exosip2ctypes.event.EventType attribute), 18
notification_redirected (exosip2ctypes.event.EventType attribute), 18

notification_requestfailure (exosip2ctypes.event.EventType attribute), 18
notification_serverfailure (exosip2ctypes.event.EventType attribute), 18

O

OsipApiNotInitialized, 14
OsipBadParameter, 14
OsipDiskFull, 15
OsipError, 14
OsipFileNotExists, 15
OsipMessage (class in exosip2ctypes.message), 18
OsipNoCommonCodec, 15
OsipNoMem, 14
OsipNoNetwork, 14
OsipNoRights, 15
OsipNotFound, 14
OsipPortBusy, 14
OsipSyntaxError, 14
OsipTimeout, 15
OsipTooMuchCall, 15
OsipUndefinedError, 14
OsipUnknownError, 14
OsipUnknownHost, 14
OsipWrongFormat, 15
OsipWrongState, 14

P

proxy (exosip2ctypes.register.InitialRegister attribute), 21
ptr (exosip2ctypes.context.Context attribute), 12
ptr (exosip2ctypes.message.OsipMessage attribute), 19
ptr (exosip2ctypes.sdp.SdpMessage attribute), 22

Q

quit() (exosip2ctypes.context.Context method), 12

R

raise_if_osip_error() (in module exosip2ctypes.error), 14
Register (class in exosip2ctypes.register), 21
registration_failure (exosip2ctypes.event.EventType attribute), 18
registration_success (exosip2ctypes.event.EventType attribute), 18
release() (exosip2ctypes.context.ContextLock method), 13
remove() (exosip2ctypes.register.InitialRegister method), 21
remove() (exosip2ctypes.register.Register method), 21
remove_authentication_info() (exosip2ctypes.context.Context method), 12
request (exosip2ctypes.event.Event attribute), 16
response (exosip2ctypes.event.Event attribute), 16
rid (exosip2ctypes.event.Event attribute), 16
rid (exosip2ctypes.register.InitialRegister attribute), 21
rid (exosip2ctypes.register.Register attribute), 21
route (exosip2ctypes.call.InitInvite attribute), 8
run() (exosip2ctypes.context.Context method), 12

S

SdpMessage (class in exosip2ctypes.sdp), 22
send() (exosip2ctypes.call.Ack method), 8
send() (exosip2ctypes.call.Answer method), 8
send() (exosip2ctypes.message.ExosipMessage method), 20
send() (exosip2ctypes.register.InitialRegister method), 21
send() (exosip2ctypes.register.Register method), 22
set_event_callback() (exosip2ctypes.context.Context method), 12
sid (exosip2ctypes.event.Event attribute), 16
ss_reason (exosip2ctypes.event.Event attribute), 16
ss_status (exosip2ctypes.event.Event attribute), 16
start() (exosip2ctypes.context.Context method), 12
status (exosip2ctypes.call.Answer attribute), 8
stop() (exosip2ctypes.context.Context method), 13
subject (exosip2ctypes.call.InitInvite attribute), 8
subscription_answered (exosip2ctypes.event.EventType attribute), 18
subscription_globalfailure (exosip2ctypes.event.EventType attribute), 18
subscription_noanswer (exosip2ctypes.event.EventType attribute), 18
subscription_notify (exosip2ctypes.event.EventType attribute), 18
subscription_proceeding (exosip2ctypes.event.EventType attribute), 18
subscription_redirected (exosip2ctypes.event.EventType attribute), 18
subscription_requestfailure (exosip2ctypes.event.EventType attribute), 18
subscription_serverfailure (exosip2ctypes.event.EventType attribute), 18

T

textinfo (exosip2ctypes.event.Event attribute), 16
tid (exosip2ctypes.call.Answer attribute), 8
tid (exosip2ctypes.event.Event attribute), 16
to (exosip2ctypes.message.OsipMessage attribute), 19
to_bytes() (in module exosip2ctypes.utils), 22
to_str() (in module exosip2ctypes.utils), 22
to_unicode() (in module exosip2ctypes.utils), 23
to_url (exosip2ctypes.call.InitInvite attribute), 8
type (exosip2ctypes.event.Event attribute), 16

U

user_agent (exosip2ctypes.context.Context attribute), 13