exoSip 开发者手册

——本手册指导开发者利用exoSip 栈开发用户代理

原文标题：exoSIP User Manual

原文作者：

联系方法：

项目网站：http://www.antisip.com/documentation/eXosip2/index.html

制作作者：周芒芝

联系方法：xing_1314@126.com

1 The eXtented eXosip stack

1.1 How-To initialize libeXosip2.

When using eXosip, your first task is to initialize both eXosip context and libosip library (parser and state machines). This must be done prior to any use of libeXosip2.

```
include <eXosip2/eXosip.h>
int i;
TRACE_INITIALIZE (6, stdout);
i=eXosip_init();
if (i!=0)
  return -1;
i = eXosip_listen_addr (IPPROTO_UDP, NULL, port, AF_INET, 0);
if (i!=0)
  {
    eXosip_quit();
    fprintf (stderr, "could not initialize transport layer\n");
    return -1;
  }
... then you have to send messages and wait for eXosip events...
```

In the previous code, you've learned how to:

- Initialize the osip trace (compile this code with -DENABLE_TRACE)
- Initialize eXosip (and osip) stack
- Open a socket for signalling (only UDP with initial eXosip2 version)

Now you have to handle eXosip events. Here is some code to get **eXosip_event** from the eXosip2 stack.

```
eXosip_event_t *je;
for (;;)
  {
    je = eXosip_event_wait (0, 50);
    eXosip_lock();
    eXosip_automatic_action ();
    eXosip_unlock();
    if (je == NULL)
      break;
    if (je->type == EXOSIP_CALL_NEW)
      {
```

```
      ....
      ....
      }
    else if (je->type == EXOSIP_CALL_ACK)
      {
      ....
      ....
      }
    else if (je->type == EXOSIP_CALL_ANSWERED)
      {
      ....
      ....
      }
    else .....
    ....
    ....
    eXosip_event_free(je);
  }
```

You will receive one event for each SIP message sent. Each event contains the original request of the affected transaction and the last response that triggers the event when available.

You can access all headers from those messages and store them in your own context for other actions or graphic displays.

For example, when you receive a REFER request for a call transfer, you'll typically retreive the "refer-To" header:

```
  osip_header_t *referto_head = NULL;
  i = osip_message_header_get_byname (evt->sip, "refer-to", 0,
&referto_head);
  if (referto_head == NULL || referto_head->hvalue == NULL)
```

The **eXosip_event** also contains identifiers for calls, registrations, incoming subscriptions or outgoing subscriptions when applicable. Those identifiers are used in API to control calls, registrations, incoming or outgoing subscriptions. These API will build default messages with usual SIP headers (From, To, Call-ID, CSeq, Route, Record-Route, Max-Forward...) and send thoses messages for you.

1.2 How-To initiate, modify or terminate calls.

eXosip2 offers a flexible API to help you controling calls.

1.2.1 Initiate a call

To start an outgoing call, you typically need a few headers which will be used by eXosip2 to build a default SIP INVITE request. The code below is used to start a call:

```
  osip_message_t *invite;
  int i;
  i = eXosip_call_build_initial_invite (&invite,
"<sip:to@antisip.com>",
                                        "<sip:from@antisip.com>",
                                        NULL, // optionnal route header
                                        "This is a call for a
conversation");
  if (i != 0)
    {
      return -1;
    }
  osip_message_set_supported (invite, "100rel");
  {
    char tmp[4096];
    char localip[128];
    eXosip_guess_localip (AF_INET, localip, 128);
    snprintf (tmp, 4096,
              "v=0\r\n"
              "o=josua 0 0 IN IP4 %s\r\n"
              "s=conversation\r\n"
              "c=IN IP4 %s\r\n"
              "t=0 0\r\n"
              "m=audio %s RTP/AVP 0 8 101\r\n"
              "a=rtpmap:0 PCMU/8000\r\n"
              "a=rtpmap:8 PCMA/8000\r\n"
              "a=rtpmap:101 telephone-event/8000\r\n"
              "a=fmtp:101 0-11\r\n", localip, localip, port);
    osip_message_set_body (invite, tmp, strlen (tmp));
    osip_message_set_content_type (invite, "application/sdp");
  }
  eXosip_lock ();
  i = eXosip_call_send_initial_invite (invite);
  if (i > 0)
    {
      eXosip_call_set_reference (i, reference);
    }
  eXosip_unlock ();
```

```
return i;
```

The above code is using eXosip_call_build_initial_invite to build a default SIP INVITE request for a new call. You have to insert a SDP body announcing your audio parameter for the RTP stream.

The above code also show the flexibility of the eXosip2 API which allow you to insert additionnal headers such as "Supported: 100rel" (announcing support for a SIP extension). Thus you can enterely control the creation of SIP requests.

The returned element of eXosip_call_send_initial_invite is the call identifier that you can use to send a CANCEL. In future events other than 100 Trying, you'll also get the dialog identifier that will also be needed to control established calls.

eXosip_call_set_reference is also a mean to attach one of your own context to a call so that you'll get your pointer back in **eXosip_event**.

1.2.2 Answer a call

The code below is another example that teach you how to answer an incoming call.

You'll usually need to send a "180 Ringing" SIP answer when receiving a SIP INVITE:

```
eXosip_lock ();
eXosip_call_send_answer (ca->tid, 180, NULL);
eXosip_unlock ();
```

Note: The above code also shows that the stack is sometimes able to build and send a default SIP messages with only one API call

Then, when the user wants to answer the call, you'll need to send a 200 ok and insert a SDP body in your SIP answer:

```
osip_message_t *answer = NULL;
eXosip_lock ();
i = eXosip_call_build_answer (ca->tid, 200, &answer);
if (i != 0)
{
   eXosip_call_send_answer (ca->tid, 400, NULL);
}
else
{
   i = sdp_complete_200ok (ca->did, answer);
   if (i != 0)
```

```
    {
       osip_message_free (answer);
       eXosip_call_send_answer (ca->tid, 415, NULL);
    }
    else
       eXosip_call_send_answer (ca->tid, 200, answer);
  }
  eXosip_unlock ();
```

Note: In the above code, you can note that to send a response to a request, you have to use the transaction identifier (and not a call identifier or a dialog identifier!)

Note2: For sending a 200ok, you'll usually need to insert a SDP body in the answer and before this, to negotiate the parameters and codecs that you want to support. In the test tool, provided by eXosip2 (josua application), you'll find a very basic implementation of the SDP negotiation.

1.2.3 Sending other request

The call control API allows you to send and receive REFER, UPDATE, INFO, OPTIONS, NOTIFY and INVITEs whitin calls. A few limitations still exist for answering other requests within calls, but it should be already possible to send any kind of request.

Here you have a code sample to send an INFO requests used to send an out of band dtmf within the signalling layer.

```
  osip_message_t *info;
  char dtmf_body[1000];
  int i;
  eXosip_lock ();
  i = eXosip_call_build_info (ca->did, &info);
  if (i == 0)
  {
     snprintf (dtmf_body, 999, "Signal=%c\r\nDuration=250\r\n", c);
     osip_message_set_content_type (info, "application/dtmf-relay");
     osip_message_set_body (info, dtmf_body, strlen (dtmf_body));
     i = eXosip_call_send_request (ca->did, info);
  }
  eXosip_unlock ();
```

1.3 How-To send or update registrations.

eXosip2 offers a flexible API to help you to register one or several identities.

1.3.1 Initiate a registration

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

```
  osip_message_t *reg = NULL;
  int id;
  int i;
  eXosip_lock ();
  id = eXosip_register_build_initial_register (identity, registrar,
NULL,
                                               1800, &reg);
  if (id < 0)
    {
      eXosip_unlock ();
      return -1;
    }
  osip_message_set_supported (reg, "100rel");
  osip_message_set_supported(reg, "path");
  i = eXosip_register_send_register (id, reg);
  eXosip_unlock ();
  return i;
```

The returned element of eXosip_register_build_initial_register is the registration identifier that you can use to update your registration. In future events about this registration, you'll see that registration identifier when applicable.

1.3.2 Update a registration

You just need to reuse the registration identifier:

```
  int i;
  eXosip_lock ();
  i = eXosip_register_build_register (id, 1800, &reg);
  if (i < 0)
    {
      eXosip_unlock ();
      return -1;
```

```
  }
eXosip_register_send_register (id, reg);
eXosip_unlock ();
```

Note: The above code also shows that the stack is sometimes able to build and send a default SIP messages with only one API call

1.3.3 Closing the registration

A softphone 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".

The same code as for updating a registration is used with 0 instead of 1800 for the expiration delay.

2 General purpose API

2.1 eXosip2 configuration API

2.1.1 Functions

int	**eXosip_init** (void)
void	**eXosip_quit** (void)
int	**eXosip_set_option** (eXosip_option opt, const void *value)
int	**eXosip_lock** (void)
int	**eXosip_unlock** (void)
int	**eXosip_listen_addr** (int transport, const char *addr, int port, int family, int secure)
int	**eXosip_set_socket** (int transport, int socket, int port)
void	**eXosip_set_user_agent** (const char *user_agent)
void	**eXosip_enable_ipv6** (int ipv6_enable)
void	**eXosip_masquerade_contact** (const char *public_address, int port)

2.1.2 Function Documentation

int eXosip_init(void)

Initiate the eXtented oSIP library.

void eXosip_quit(void)

Release ressource used by the eXtented oSIP library.

int eXosip_set_option (eXosip_option *opt*, const void * *value*)

Set eXosip options. See eXosip_option for available options.

Parameters:

opt option to configure.

value value for options.

int eXosip_lock(void)

Lock the eXtented oSIP library.

int eXosip_unlock(void)

UnLock the eXtented oSIP library.

```
int eXosip_listen_addr (int transport,
                        const char * addr,
                        int port,
                        int family,
                        int secure)
```

Listen on a specified socket.

Parameters:

transport IPPROTO_UDP for udp. (soon to come: TCP/TLS?)

addr the address to bind (NULL for all interface)

port the listening port. (0 for random port)

family the IP family (AF_INET or AF_INET6).

secure 0 for UDP or TCP, 1 for TLS (with TCP).

```
int eXosip_set_socket (int transport,
                       int socket,
                       int port)
```

Listen on a specified socket.

Parameters:

transport IPPROTO_UDP for udp. (soon to come: TCP/TLS?)

socket socket to use for listening to UDP sip messages.

port the listening port for masquerading.

void eXosip_set_user_agent(const char * *user_agent*)

Set the SIP User-Agent: header string.

Parameters:

user_agent the User-Agent header to insert in messages.

void eXosip_enable_ipv6(int *ipv6_enable*)

Use IPv6 instead of IPv4.

Parameters:

ipv6_enable This paramter should be set to 1 to enable IPv6 mode.

```
void                    (const
eXosip_masquerade_       char *    public_address,
contact
                         int       port
                        )
```

This method is used to replace contact address with the public address of your NAT. The ip address should be retreived manually (fixed IP address) or with STUN. This address will only be used when the remote correspondant appears to be on an DIFFERENT LAN.

Parameters:

public_address the ip address.

port the port for masquerading.

If set to NULL, then the local ip address will be guessed automatically (returns to default mode).

2.2 eXosip2 network API

2.2.2 Functions

int	**eXosip_transport_set** (osip_message_t *msg, const char *transport)
int	**eXosip_guess_localip** (int family, char *address, int size)

2.2.3 Function Documentation

```
int eXosip_transport_set (osip_message_t * msg,
                          const char * transport
                         )
```

Modify the transport protocol used to send SIP message.

Parameters:

msg The SIP message to modify

transport transport protocol to use ("UDP", "TCP" or "TLS")

```
int eXosip_guess_localip (int family,
                          char * address,
                          int size
                         )
```

Find the current localip (interface with default route).

Parameters:

family	AF_INET or AF_INET6
address	a string containing the local IP address.
size	The size of the string

2.3 eXosip2 event API

2.3.1 Data Structures

struct	**eXosip_event**
struct	**eXosip_event**

2.3.2 Enumerations

enum	**eXosip_event_type** { **EXOSIP_REGISTRATION_NEW**, **EXOSIP_REGISTRATION_SUCCESS**, **EXOSIP_REGISTRATION_FAILURE**, **EXOSIP_REGISTRATION_REFRESHED**, **EXOSIP_REGISTRATION_TERMINATED**, **EXOSIP_CALL_INVITE**, **EXOSIP_CALL_REINVITE**, **EXOSIP_CALL_NOANSWER**, **EXOSIP_CALL_PROCEEDING**, **EXOSIP_CALL_RINGING**, **EXOSIP_CALL_ANSWERED**, **EXOSIP_CALL_REDIRECTED**, **EXOSIP_CALL_REQUESTFAILURE**, **EXOSIP_CALL_SERVERFAILURE**, **EXOSIP_CALL_GLOBALFAILURE**, **EXOSIP_CALL_ACK**, **EXOSIP_CALL_CANCELLED**, **EXOSIP_CALL_TIMEOUT**, **EXOSIP_CALL_MESSAGE_NEW**, **EXOSIP_CALL_MESSAGE_PROCEEDING**, **EXOSIP_CALL_MESSAGE_ANSWERED**, **EXOSIP_CALL_MESSAGE_REDIRECTED**, **EXOSIP_CALL_MESSAGE_REQUESTFAILURE**, **EXOSIP_CALL_MESSAGE_SERVERFAILURE**, **EXOSIP_CALL_MESSAGE_GLOBALFAILURE**, **EXOSIP_CALL_CLOSED**, **EXOSIP_CALL_RELEASED**, **EXOSIP_MESSAGE_NEW**,

	EXOSIP_MESSAGE_PROCEEDING, **EXOSIP_MESSAGE_ANSWERED**, **EXOSIP_MESSAGE_REDIRECTED**, **EXOSIP_MESSAGE_REQUESTFAILURE**, **EXOSIP_MESSAGE_SERVERFAILURE**, **EXOSIP_MESSAGE_GLOBALFAILURE**, **EXOSIP_SUBSCRIPTION_UPDATE**, **EXOSIP_SUBSCRIPTION_CLOSED**, **EXOSIP_SUBSCRIPTION_NOANSWER**, **EXOSIP_SUBSCRIPTION_PROCEEDING**, **EXOSIP_SUBSCRIPTION_ANSWERED**, **EXOSIP_SUBSCRIPTION_REDIRECTED**, **EXOSIP_SUBSCRIPTION_REQUESTFAILURE**, **EXOSIP_SUBSCRIPTION_SERVERFAILURE**, **EXOSIP_SUBSCRIPTION_GLOBALFAILURE**, **EXOSIP_SUBSCRIPTION_NOTIFY**, **EXOSIP_SUBSCRIPTION_RELEASED**, **EXOSIP_IN_SUBSCRIPTION_NEW**, **EXOSIP_IN_SUBSCRIPTION_RELEASED**, **EXOSIP_NOTIFICATION_NOANSWER**, **EXOSIP_NOTIFICATION_PROCEEDING**, **EXOSIP_NOTIFICATION_ANSWERED**, **EXOSIP_NOTIFICATION_REDIRECTED**, **EXOSIP_NOTIFICATION_REQUESTFAILURE**, **EXOSIP_NOTIFICATION_SERVERFAILURE**, **EXOSIP_NOTIFICATION_GLOBALFAILURE**, **EXOSIP_EVENT_COUNT** }

2.3.3 Functions

void	**eXosip_event_free** (**eXosip_event_t** *je)
eXosip_event_t *	**eXosip_event_wait** (int tv_s, int tv_ms)
eXosip_event_t *	**eXosip_event_get** (void)

2.3.4 Enumeration Type Documentation

enum eXosip_event_type

Structure for event type description

Enumerator:

EXOSIP_REGISTRATION_NEW	announce new registration.
EXOSIP_REGISTRATION_SUCCESS	user is successfully registred.
EXOSIP_REGISTRATION_FAILURE	user is not registred.
EXOSIP_REGISTRATION_REFRESHED	registration has been refreshed.
EXOSIP_REGISTRATION_TERMINATED	UA is not registred any more.
EXOSIP_CALL_INVITE	announce a new call
EXOSIP_CALL_REINVITE	announce a new INVITE within call
EXOSIP_CALL_NOANSWER	announce no answer within the timeout
EXOSIP_CALL_PROCEEDING	announce processing by a remote app
EXOSIP_CALL_RINGING	announce ringback
EXOSIP_CALL_ANSWERED	announce start of call
EXOSIP_CALL_REDIRECTED	announce a redirection
EXOSIP_CALL_REQUESTFAILURE	announce a request failure
EXOSIP_CALL_SERVERFAILURE	announce a server failure
EXOSIP_CALL_GLOBALFAILURE	announce a global failure
EXOSIP_CALL_ACK	ACK received for 200ok to INVITE
EXOSIP_CALL_CANCELLED	announce that call has been cancelled
EXOSIP_CALL_TIMEOUT	announce that call has failed
EXOSIP_CALL_MESSAGE_NEW	announce new incoming request.
EXOSIP_CALL_MESSAGE_PROCEEDING	announce a 1xx for request.
EXOSIP_CALL_MESSAGE_ANSWERED	announce a 200ok
EXOSIP_CALL_MESSAGE_REDIRECTED	announce a failure.
EXOSIP_CALL_MESSAGE_REQUESTFAILURE	announce a failure.
EXOSIP_CALL_MESSAGE_SERVERFAILURE	announce a failure.
EXOSIP_CALL_MESSAGE_GLOBALFAILURE	announce a failure.
EXOSIP_CALL_CLOSED	a BYE was received for this call
EXOSIP_CALL_RELEASED	call context is cleared.
EXOSIP_MESSAGE_NEW	announce new incoming request.
EXOSIP_MESSAGE_PROCEEDING	announce a 1xx for request.
EXOSIP_MESSAGE_ANSWERED	announce a 200ok
EXOSIP_MESSAGE_REDIRECTED	announce a failure.
EXOSIP_MESSAGE_REQUESTFAILURE	announce a failure.
EXOSIP_MESSAGE_SERVERFAILURE	announce a failure.
EXOSIP_MESSAGE_GLOBALFAILURE	announce a failure.

EXOSIP_SUBSCRIPTION_UPDATE	announce incoming SUBSCRIBE.
EXOSIP_SUBSCRIPTION_CLOSED	announce end of subscription.
EXOSIP_SUBSCRIPTION_NOANSWER	announce no answer
EXOSIP_SUBSCRIPTION_PROCEEDING	announce a 1xx
EXOSIP_SUBSCRIPTION_ANSWERED	announce a 200ok
EXOSIP_SUBSCRIPTION_REDIRECTED	announce a redirection
EXOSIP_SUBSCRIPTION_REQUESTFAILURE	announce a request failure
EXOSIP_SUBSCRIPTION_SERVERFAILURE	announce a server failure
EXOSIP_SUBSCRIPTION_GLOBALFAILURE	announce a global failure
EXOSIP_SUBSCRIPTION_NOTIFY	announce new NOTIFY request
EXOSIP_SUBSCRIPTION_RELEASED	call context is cleared.
EXOSIP_IN_SUBSCRIPTION_NEW	announce new incoming SUBSCRIBE.
EXOSIP_IN_SUBSCRIPTION_RELEASED	announce end of subscription.
EXOSIP_NOTIFICATION_NOANSWER	announce no answer
EXOSIP_NOTIFICATION_PROCEEDING	announce a 1xx
EXOSIP_NOTIFICATION_ANSWERED	announce a 200ok
EXOSIP_NOTIFICATION_REDIRECTED	announce a redirection
EXOSIP_NOTIFICATION_REQUESTFAILURE	announce a request failure
EXOSIP_NOTIFICATION_SERVERFAILURE	announce a server failure
EXOSIP_NOTIFICATION_GLOBALFAILURE	announce a global failure
EXOSIP_EVENT_COUNT	MAX number of events

2.3.5 Function Documentation

void eXosip_event_free(eXosip_event_t * *je*)

Free ressource in an eXosip event.

Parameters:

je event to work on.

```
eXosip_event_t*
eXosip_event_wait (int tv_s,
                   int tv_ms
                  )
```

Wait for an eXosip event.

Parameters:

tv_s	timeout value (seconds).
tv_ms	timeout value (mseconds).

eXosip_event_t* eXosip_event_get(void)

Wait for next eXosip event.

3 SIP messages and call control API

3.1 eXosip2 INVITE and Call Management

3.1.1 Functions

int	**eXosip_call_set_reference** (int id, void *reference)
int	**eXosip_call_build_initial_invite** (osip_message_t **invite, const char *to, const char *from, const char *route, const char *subject)
int	**eXosip_call_send_initial_invite** (osip_message_t *invite)
int	**eXosip_call_build_request** (int did, const char *method, osip_message_t **request)
int	**eXosip_call_build_ack** (int did, osip_message_t **ack)
int	**eXosip_call_send_ack** (int did, osip_message_t *ack)
int	**eXosip_call_build_refer** (int did, const char *refer_to, osip_message_t **request)
int	**eXosip_call_build_info** (int did, osip_message_t **request)
int	**eXosip_call_build_options** (int did, osip_message_t **request)
int	**eXosip_call_build_update** (int did, osip_message_t **request)
int	**eXosip_call_build_notify** (int did, int subscription_status, osip_message_t **request)
int	**eXosip_call_send_request** (int did, osip_message_t *request)
int	**eXosip_call_build_answer** (int tid, int status, osip_message_t **answer)
int	**eXosip_call_send_answer** (int tid, int status, osip_message_t *answer)
int	**eXosip_call_terminate** (int cid, int did)
int	**eXosip_call_build_prack** (int tid, osip_message_t **prack)
int	**eXosip_call_send_prack** (int tid, osip_message_t *prack)
int	**eXosip_transfer_send_notify** (int did, int subscription_status, char *body)
int	**eXosip_call_get_referto** (int did, char *refer_to, size_t refer_to_len)

3.1.2 Function Documentation

```
int eXosip_call_set_reference (int id, void *reference)
```

Set a new application context for an existing call

Parameters:

id	call-id or dialog-id of call
reference	New application context.

```
int eXosip_call_build_initial_invite (osip_message_t **invite, const char *to, const char *from, const char *route, const char *subject)
```

Build a default INVITE message for a new call.

Parameters:

invite	Pointer for the SIP element to hold.
to	SIP url for callee.
from	SIP url for caller.
route	Route header for INVITE. (optionnal)
subject	Subject for the call.

```
int eXosip_call_send_initial_invite( osip_message_t * invite )
```

Initiate a call.

Parameters:

invite SIP INVITE message to send.

```
int eXosip_call_build_request (int did, const char *method, osip_message_t **request)
```

Build a default request within a call. (INVITE, OPTIONS, INFO, REFER)

Parameters:

did dialog id of call.

method request type to build.

request The sip request to build.

```
int eXosip_call_build_ack (int did, osip_message_t **ack)
```

Build a default ACK for a 200ok received.

Parameters:

did dialog id of call.

ack The sip request to build.

```
int eXosip_call_send_ack (int did, osip_message_t *ack)
```

Send the ACK for the 200ok received..

Parameters:

did dialog id of call.

ack SIP ACK message to send.

```
int eXosip_call_build_refer (int did, const char * refer_to, osip_message_t ** request)
```

Build a default REFER for a call transfer.

Parameters:

did dialog id of call.

refer_to url for call transfer (Refer-To header).

request The sip request to build.

```
int eXosip_call_build_info (int did, osip_message_t ** request)
```

Build a default INFO within a call.

Parameters:

did dialog id of call.

request The sip request to build.

```
int eXosip_call_build_options (int did, osip_message_t ** request)
```

Build a default OPTIONS within a call.

Parameters:

did dialog id of call.

request The sip request to build.

```
int                 (int
eXosip_call_build                 did,
_update
                     osip_messa
                     ge_t **      request
                    )
```

Build a default UPDATE within a call.

Parameters:

did	dialog id of call.
request	The sip request to build.

```
int                 (int
eXosip_call_build                 did,
_notify
                     int          subscription_status,
                     osip_messa
                     ge_t **      request
                    )
```

Build a default NOTIFY within a call.

Parameters:

did	dialog id of call.
subscription_status	Subscription status of the request.
request	The sip request to build.

```
int                 (int
eXosip_call_send_                 did,
request
                     osip_mess
                     age_t *      request
                    )
```

send the request within call. (INVITE, OPTIONS, INFO, REFER, UPDATE)

Parameters:

did	dialog id of call.
request	The sip request to send.

```
int                 (int          tid,
```

```
eXosip_call_build_answer
                        (int status,
                         osip_message_t ** answer
                        )
```

Build default Answer for request.

Parameters:

tid	id of transaction to answer.
status	Status code to use.
answer	The sip answer to build.

```
int eXosip_call_send_answer
                        (int tid,
                         int status,
                         osip_message_t * answer
                        )
```

Send Answer for invite.

Parameters:

tid	id of transaction to answer.
status	response status if answer is NULL. (not allowed for 2XX)
answer	The sip answer to send.

```
int eXosip_call_terminate
                        (int cid,
                         int did
                        )
```

Terminate a call. send CANCEL, BYE or 603 Decline.

Parameters:

cid	call id of call.
did	dialog id of call.

```
int                 (int
eXosip_call_build               tid,
_prack
                     osip_messa
                     ge_t **    prack
                    )
```

Build a PRACK for invite.

Parameters:

tid id of the invite transaction.

prack The sip prack to build.

```
int                 (int
eXosip_call_send                tid,
_prack
                     osip_messa
                     ge_t *     prack
                    )
```

Send a PRACK for invite.

Parameters:

tid id of the invite transaction.

prack The sip prack to send.

```
int                       (int
eXosip_transfer_send            did,
_notify
                           int  subscription_status,
                           char
                           *    body
                          )
```

Send a NOTIFY containing the information about a call transfer.

THIS METHOD WILL BE REPLACED or REMOVED, please use the new API to build NOTIFY.

Parameters:

did dialog id of call.

subscription_status the subscription status.

body the body to attach to NOTIFY.

```
int eXosip_call_get_referto (int did, char * refer_to, size_t refer_to_len)
```

Get Refer-To header with Replace parameter from dialog.

Parameters:

did	id of the dialog.
refer_to	buffer to be filled with refer-to info.
refer_to_len	size of refer_to buffer.

3.2 eXosip2 request outside of dialog

3.2.1 Functions

int	**eXosip_message_build_request** (osip_message_t **message, const char *method, const char *to, const char *from, const char *route)
int	**eXosip_message_send_request** (osip_message_t *message)
int	**eXosip_message_build_answer** (int tid, int status, osip_message_t **answer)
int	**eXosip_message_send_answer** (int tid, int status, osip_message_t *answer)

3.2.2 Function Documentation

```
int                     (osip_mess
eXosip_message_bui      age_t **      message,
ld_request
                         const  char
                         *             method,
                         const  char
                         *             to,
                         const  char
                         *             from,
                         const  char
                         *             route
                        )
```

Build a default request message.

This method will be updated to send any message outside of dialog In this later case, you'll specify the method to use in the second argument.

Parameters:

message Pointer for the SIP request to build.

method request method. (like "MESSAGE" or "PING"...)

to SIP url for callee.

from SIP url for caller.

route Route header for request. (optionnal)

int eXosip_message_send_request(osip_message_t * *message*)

Send an request.

Parameters:

message SIP request to send.

```
int eXosip_message_build_answer (int tid, int status, osip_message_t ** answer)
```

Build answer for a request.

Parameters:

tid id of transaction.

status status for SIP answer to build.

answer The SIP answer to build.

```
int eXosip_message_send_answer (int tid, int status, osip_message_t * answer)
```

Send answer for a request.

Parameters:

tid id of transaction.

status status for SIP answer to send.

answer The SIP answer to send. (default will be sent if NULL)

3.3 eXosip2 OPTIONS and UA capabilities Management

3.3.1 Functions

int	**eXosip_options_build_request** (osip_message_t **options, const char *to, const char *from, const char *route)
int	**eXosip_options_send_request** (osip_message_t *options)
int	**eXosip_options_build_answer** (int tid, int status, osip_message_t **answer)
int	**eXosip_options_send_answer** (int tid, int status, osip_message_t *answer)

3.3.2 Function Documentation

```
int eXosip_options_build_request (osip_message_t ** options,
                                  const char * to,
                                  const char * from,
                                  const char * route
                                 )
```

Build a default OPTIONS message.

Parameters:

options	Pointer for the SIP request to build.
to	SIP url for callee.
from	SIP url for caller.
route	Route header for INVITE. (optionnal)

```
int eXosip_options_send_request( osip_message_t * options )
```

Send an OPTIONS request.

Parameters:

options SIP OPTIONS message to send.

```
int eXosip_options_build_answer (int tid, int status, osip_message_t ** answer)
```

Build answer for an OPTIONS request.

Parameters:

tid id of OPTIONS transaction.

status status for SIP answer to build.

answer The SIP answer to build.

```
int eXosip_options_send_answer (int tid, int status, osip_message_t * answer)
```

Send answer for an OPTIONS request.

Parameters:

tid id of OPTIONS transaction.

status status for SIP answer to send.

answer The SIP answer to send. (default will be sent if NULL)

3.4 eXosip2 Publication Management

3.4.1 Functions

int	**eXosip_build_publish** (osip_message_t **message, const char *to, const char *from, const char *route, const char *event, const char *expires, const char *ctype, const char *body)
int	**eXosip_publish** (osip_message_t *message, const char *sip_if_match)

3.4.2 Function Documentation

```
int                (osip_messa
eXosip_build_p      ge_t **       message,
ublish
                    const char * to,
                    const char * from,
                    const char * route,
                    const char * event,
                    const char * expires,
                    const char * ctype,
                    const char * body
                   )
```

build publication for a user. (PUBLISH request)

Parameters:

message	returned published request.
to	SIP url for callee.
from	SIP url for caller.
route	Route used for publication.
event	SIP Event header.
expires	SIP Expires header.
ctype	Content-Type of body.
body	body for publication.

```
int             (osip_messa message,
```

```
eXosip_publish *
                const char * sip_if_match
               )
```

Send an Publication Message (PUBLISH request).

Parameters:

message	is a ready to be sent publish message .
sip_if_match	is the SIP-If-Match header. (NULL for initial publication)

3.5 eXosip2 REFER and blind tranfer Management outside of calls

3.5.1 Functions

int	**eXosip_refer_build_request** (osip_message_t **refer, const char *refer_to, const char *from, const char *to, const char *route)
int	**eXosip_refer_send_request** (osip_message_t *refer)

3.5.2 Function Documentation

```
int eXosip_refer_build_request (osip_message_t ** refer,
                                const char * refer_to,
                                const char * from,
                                const char * to,
                                const char * route
                               )
```

Build a default REFER message for a blind transfer outside of any calls.

Parameters:

refer	Pointer for the SIP element to hold.
refer_to	SIP url for transfer.
from	SIP url for caller.
to	SIP url for callee.
route	Route header for REFER. (optionnal)

```
int eXosip_refer_send_request( osip_message_t * refer )
```

Initiate a blind tranfer outside of any call.

Parameters:

refer SIP REFER message to send.

3.6 eXosip2 REGISTER and Registration Management

3.6.1 Functions

int	**eXosip_register_build_initial_register** (const char *from, const char *proxy, const char *contact, int expires, osip_message_t **reg)
int	**eXosip_register_build_register** (int rid, int expires, osip_message_t **reg)
int	**eXosip_register_send_register** (int rid, osip_message_t *reg)
int	**eXosip_register_remove** (int rid)

3.6.2 Function Documentation

```
int eXosip_register_build_initial_register (const char * from,
                                            const char * proxy,
                                            const char * contact,
                                            int expires,
                                            osip_message_t ** reg
                                           )
```

Build initial REGISTER request.

Parameters:

from	SIP url for caller.
proxy	Proxy used for registration.
contact	Contact address. (optional)
expires	The expires value for registration.

reg The SIP request to build.

```
int eXosip_register_build_register (int rid, int expires, osip_message_t ** reg)
```

Build a new REGISTER request for an existing registration.

Parameters:

rid A unique identifier for the registration context

expires The expires value for registration.

reg The SIP request to build.

```
int eXosip_register_send_register (int rid, osip_message_t * reg)
```

Send a REGISTER request for an existing registration.

Parameters:

rid A unique identifier for the registration context

reg The SIP request to build. (NULL for default REGISTER)

```
int eXosip_register_remove( int rid )
```

Remove existing registration without sending REGISTER.

Parameters:

rid A unique identifier for the registration context

3.7 eXosip2 SUBSCRIBE and outgoing subscriptions

3.7.1 Enumerations

enum	**eXosip_ss** {
	EXOSIP_SUBCRSTATE_UNKNOWN,
	EXOSIP_SUBCRSTATE_PENDING,
	EXOSIP_SUBCRSTATE_ACTIVE,
	EXOSIP_SUBCRSTATE_TERMINATED
	}

enum	**eXosip_ss_reason** {
	DEACTIVATED,
	PROBATION,
	REJECTED,
	TIMEOUT,
	GIVEUP,
	NORESOURCE
	}

enum	**eXosip_ss_status** {
	EXOSIP_NOTIFY_UNKNOWN,
	EXOSIP_NOTIFY_PENDING,
	EXOSIP_NOTIFY_ONLINE,
	EXOSIP_NOTIFY_BUSY,
	EXOSIP_NOTIFY_BERIGHTBACK,
	EXOSIP_NOTIFY_AWAY,
	EXOSIP_NOTIFY_ONTHEPHONE,
	EXOSIP_NOTIFY_OUTTOLUNCH,
	EXOSIP_NOTIFY_CLOSED
	}

3.7.2 Functions

int	**eXosip_subscribe_build_initial_request** (osip_message_t **subscribe, const char *to, const char *from, const char *route, const char *event, int expires)

int	**eXosip_subscribe_send_initial_request** (osip_message_t *subscribe)
int	**eXosip_subscribe_build_refresh_request** (int did, osip_message_t **sub)
int	**eXosip_subscribe_send_refresh_request** (int did, osip_message_t *sub)

3.7.3 Enumeration Type Documentation

enum eXosip_ss

Enumerator:

EXOSIP_SUBCRSTATE_UNKNOWN	unknown subscription-state
EXOSIP_SUBCRSTATE_PENDING	pending subscription-state
EXOSIP_SUBCRSTATE_ACTIVE	active subscription-state
EXOSIP_SUBCRSTATE_TERMINATED	terminated subscription-state

enum eXosip_ss_reason

Enumerator:

DEACTIVATED	deactivated for subscription-state
PROBATION	probation for subscription-state
REJECTED	rejected for subscription-state
TIMEOUT	timeout for subscription-state
GIVEUP	giveup for subscription-state
NORESOURCE	noresource for subscription-state

enum eXosip_ss_status

Enumerator:

EXOSIP_NOTIFY_UNKNOWN	unknown state for subscription
EXOSIP_NOTIFY_PENDING	subscription not yet accepted
EXOSIP_NOTIFY_ONLINE	online status
EXOSIP_NOTIFY_BUSY	busy status
EXOSIP_NOTIFY_BERIGHTBACK	be right back status
EXOSIP_NOTIFY_AWAY	away status
EXOSIP_NOTIFY_ONTHEPHONE	on the phone status
EXOSIP_NOTIFY_OUTTOLUNCH	out to lunch status
EXOSIP_NOTIFY_CLOSED	closed status

3.7.4 Function Documentation

```
int                          (osip_mess
eXosip_subscribe_build        age_t **      subscribe,
_initial_request
                              const  char
                              *             to,
                              const  char
                              *             from,
                              const  char
                              *             route,
                              const  char
                              *             event,
                              int           expires
                             )
```

Build a default initial SUBSCRIBE request.

Parameters:

subscribe	Pointer for the SIP request to build.
to	SIP url for callee.
from	SIP url for caller.
route	Route header for SUBSCRIBE. (optionnal)
event	Event header for SUBSCRIBE.
expires	Expires header for SUBSCRIBE.

```
int eXosip_subscribe_send_initial_request( osip_message_t * subscribe )
```

Send an initial SUBSCRIBE request.

Parameters:

subscribe	SIP SUBSCRIBE message to send.

```
int                          (int
eXosip_subscribe_build_                   did,
refresh_request
                              osip_mess
                              age_t **    sub
                             )
```

Build a default new SUBSCRIBE message.

Parameters:

did identifier of the subscription.

sub Pointer for the SIP request to build.

```
int eXosip_subscribe_send_refresh_request (int did, osip_message_t * sub)
```

Send a new SUBSCRIBE request.

Parameters:

did identifier of the subscription.

sub SIP SUBSCRIBE message to send.

3.8 eXosip2 SUBSCRIBE and incoming subscriptions

3.8.1 Functions

int	**eXosip_insubscription_build_answer** (int tid, int status, osip_message_t **answer)
int	**eXosip_insubscription_send_answer** (int tid, int status, osip_message_t *answer)
int	**eXosip_insubscription_build_request** (int did, const char *method, osip_message_t **request)
int	**eXosip_insubscription_build_notify** (int did, int subscription_status, int subscription_reason, osip_message_t **request)
int	**eXosip_insubscription_send_request** (int did, osip_message_t *request)

3.8.2 Function Documentation

int eXosip_insubscription_build_answer (int *tid*, int *status*, osip_message_t ** *answer*)

Build answer for an SUBSCRIBE request.

Parameters:

tid	id of SUBSCRIBE transaction.
status	status for SIP answer to build.
answer	The SIP answer to build.

int eXosip_insubscription_send_answer (int *tid*,

```
                                int           status,
                                osip_mess
                                age_t *       answer
                               )
```

Send answer for an SUBSCRIBE request.

Parameters:

tid id of SUBSCRIBE transaction.

status status for SIP answer to send.

answer The SIP answer to send. (default will be sent if NULL)

```
int                            (int
eXosip_insubscription_                      did,
build_request
                                const  char
                                *             method,
                                osip_mess
                                age_t **      request
                               )
```

Build a request within subscription.

Parameters:

did id of incoming subscription.

method request method to build.

request The SIP request to build.

```
int                           (int
eXosip_insubscription                       did,
_build_notify
                                int           subscription_status,
                                int           subscription_reason,
                                osip_mess
                                age_t **      request
                               )
```

Build a NOTIFY request within subscription.

Parameters:

did id of incoming subscription.

subscription_status subscription status (pending, active, terminated)

subscription_reason subscription reason

request The SIP request to build.

```
int eXosip_insubscription_send_request (int did, osip_message_t * request)
```

Send a request within subscription.

Parameters:

did id of incoming subscription.

request The SIP request to send.

3.9 eXosip2 authentication API

3.9.1 Functions

int	**eXosip_add_authentication_info** (const char *username, const char *userid, const char *passwd, const char *ha1, const char *realm)
int	**eXosip_clear_authentication_info** (void)
int	**eXosip_default_action** (**eXosip_event_t** *je)
void	**eXosip_automatic_refresh** (void)
void	**eXosip_automatic_action** (void)
int	**eXosip_generate_random** (char *buf, int buf_size)

3.9.2 Function Documentation

```
int eXosip_add_authentication_info (const char * username,
                                    const char * userid,
                                    const char * passwd,
                                    const char * ha1,
                                    const char * realm
                                   )
```

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

Parameters:

username	username
userid	login (usually equals the username)
passwd	password

ha1 currently ignored

realm realm within which credentials apply, or NULL to apply credentials to unrecognized realms

int eXosip_clear_authentication_info(void)

Clear all authentication credentials stored in eXosip.

int eXosip_default_action(eXosip_event_t * *je*)

Initiate some default actions:

Retry with credentials upon reception of 401/407. Retry with Contact header upon reception of 3xx request.

void eXosip_automatic_refresh(void)

Refresh REGISTER and SUBSCRIBE before the expiration delay.

void eXosip_automatic_action(void)

Initiate some automatic actions:

Retry with credentials upon reception of 401/407. Refresh REGISTER and SUBSCRIBE before the expiration delay. Retry with Contact header upon reception of 3xx request.

```
int eXosip_generate_random (char * buf, int buf_size)
```

Generate random string:

Parameters:

buf destination buffer for random string.

buf_size size of destination buffer

3.10 Xosip2 SDP helper API.

3.10.1 Functions

sdp_message_t * **eXosip_get_remote_sdp** (int did)

sdp_message_t * **eXosip_get_local_sdp** (int did)

sdp_message_t ***eXosip_get_remote_sdp_from_tid** (int tid)

sdp_message_t ***eXosip_get_local_sdp_from_tid** (int tid)

sdp_message_t ***eXosip_get_sdp_info** (osip_message_t *message)

sdp_connection_t ***eXosip_get_audio_connection** (sdp_message_t *sdp)

sdp_media_t ***eXosip_get_audio_media** (sdp_message_t *sdp)

3.10.2 Function Documentation

sdp_message_t* eXosip_get_remote_sdp(int *did*)

Get remote SDP body for the latest INVITE of call.

Parameters:

did dialog id of call.

sdp_message_t* eXosip_get_local_sdp(int *did*)

Get local SDP body for the latest INVITE of call.

Parameters:

did dialog id of call.

sdp_message_t* eXosip_get_remote_sdp_from_tid(int *tid*)

Get remote SDP body for the latest INVITE of call.

Parameters:

tid transction id of transaction.

sdp_message_t* eXosip_get_local_sdp_from_tid(int *tid*)

Get local SDP body for the latest INVITE of call.

Parameters:

tid transction id of transaction.

sdp_message_t* eXosip_get_sdp_info(osip_message_t * *message*)

Get local SDP body for the given message.

Parameters:

message message containing the SDP.

sdp_connection_t* eXosip_get_audio_connection(sdp_message_t * *sdp*)

Get audio connection information for call.

Parameters:

sdp sdp information.

sdp_media_t* eXosip_get_audio_media(sdp_message_t * *sdp*)

Get audio media information for call.

Parameters:

sdp sdp information.