# How-To initiate, modify or terminate calls.



eXosip2 offers a flexible API to help you controling calls.

## 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 cid;
int i;

i = eXosip_call_build_initial_invite (ctx, &invite, "<sip:to@antisip.com>",
                                      "<sip:from@antisip.com>",
                                      NULL, // optional 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 (ctx, AF_INET, localip, 128);
  snprintf (tmp, 4096,
            "v=0\r\n"
            "o=jack 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 (ctx);
cid = eXosip_call_send_initial_invite (ctx, invite);
if (cid > 0)
  {
    eXosip_call_set_reference (ctx, i, reference);
  }
eXosip_unlock (ctx);
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 cid (call identifier) that you can use to send a CANCEL. In future events other than 100 Trying, you'll also get the did (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**.

## 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 (ctx);
eXosip_call_send_answer (ctx, evt->tid, 180, NULL);
eXosip_unlock (ctx);
```

**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 (ctx);
i = eXosip_call_build_answer (ctx, evt->tid, 200, &answer);
if (i != 0)
{
   eXosip_call_send_answer (ctx, evt->tid, 400, NULL);
}
else
{
   i = sdp_complete_200ok (evt->did, answer);
   if (i != 0)
   {
      osip_message_free (answer);
      eXosip_call_send_answer (ctx, evt->tid, 415, NULL);
   }
   else
      eXosip_call_send_answer (ctx, evt->tid, 200, answer);
}
eXosip_unlock (ctx);
```

**Note**: In the above code, you can note that to send a response to a request, you have to use the tid (transaction identifier) and not a cid (call identifier) or a did (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. This is left to you! Once you have created the SDP, you add it in the answer using the following code:

```
osip_message_set_body (answer, tmp, strlen (tmp));
osip_message_set_content_type (answer, "application/sdp");
```

## Terminate a Call

Simple API, no much to say about it! You can use it when you want: it will either send a CANCEL, a negative answer or a BYE depending on the call state.

```
eXosip_lock (ctx);
eXosip_call_terminate (ctx, cid, did);
eXosip_unlock (ctx);
```

**Note**: You can't stop a call where no 100 Trying has been received. In that case, you need to wait before sending a CANCEL or a BYE... This is per rfc3261.

## Sending INFO, REFER, UPDATE, NOTIFY, OPTIONS request

The call control API allows you to send and receive REFER, UPDATE, INFO, OPTIONS, NOTIFY and INVITEs whitin calls.

Here you have a code sample to send an INFO requests used to send an out of band dtmf within the signalling layer. (not standard, but still used on some system!)

```
osip_message_t *info;
char dtmf_body[1000];
int i;

eXosip_lock (ctx);
i = eXosip_call_build_info (ctx, evt->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 (ctx, evt->did, info);
}
eXosip_unlock (ctx);
```

## Sending any other request, with any header

You can in fact, send any kind of other request using eXosip2 API.

You will find many other API to build any kind of sip message. Using osip API, you can add any header or body in those message. eXosip2 will always prepare the minimal and technical stuff you need.

```
osip_message_t *message;
char body[1000];
int i;

eXosip_lock (ctx);
i = eXosip_call_build_request (ctx, evt->did, "PRIVATECOMMAND", &message);
if (i == 0)
{
   snprintf (body, 999, "room=1;light=on\r\nroom=2;light=off\r\n");
   osip_message_set_content_type (message, "application/antisip-domotic");
   osip_message_set_body (message, body, strlen (body));

   osip_message_set_header (invite, "P-MyCommand", "option=value");

   i = eXosip_call_send_request (ctx, evt->did, message);
}
eXosip_unlock (ctx);
```