SDK接口说明文档

版本：V1.0

目录

[SDK接口说明文档 1](#_Toc405134018)

[1.初始化接口 4](#_Toc405134019)

[2.重置 4](#_Toc405134020)

[3.退出注销 4](#_Toc405134021)

[4.设置语音编码 4](#_Toc405134022)

[5.修改语言编码的比特率 5](#_Toc405134023)

[6.设置语言传输的初始端口 5](#_Toc405134024)

[7.设置服务器地址 5](#_Toc405134025)

[8.设置sip via header中rport是否使用 6](#_Toc405134026)

[9.设置DNS查找 6](#_Toc405134027)

[10.设置语音传输协议 6](#_Toc405134028)

[11.设置在呼叫的Session是否超时 6](#_Toc405134029)

[12.设置是否使能symmetric RTP streams 7](#_Toc405134030)

[13.选择输入音频设备 7](#_Toc405134031)

[14.选择输出音频设备 7](#_Toc405134032)

[15.WebRtc设置 7](#_Toc405134033)

[16.设置回波消除器 8](#_Toc405134034)

[17.防抖动设置 8](#_Toc405134035)

[18.设置登录用户信息 9](#_Toc405134036)

[19.加密设置 9](#_Toc405134037)

[20.设置SIP消息中的Supported字段 9](#_Toc405134038)

[21.设置SIP消息中的Allowed字段 9](#_Toc405134039)

[22.设置音频的采样频率 10](#_Toc405134040)

[23. 设置音频增益 10](#_Toc405134041)

[24. 设置回音限制 10](#_Toc405134042)

[25. 设置事件等待时间 11](#_Toc405134043)

[26. 释放事件 11](#_Toc405134044)

[27. 获得事件类型 11](#_Toc405134045)

[28. 获得transactions的id 11](#_Toc405134046)

[29. 获得SIP calls的id 12](#_Toc405134047)

[30. 获得SIP dialogs的id 12](#_Toc405134048)

[31. 获得registration的id 12](#_Toc405134049)

[32. 获得incoming subscriptions的id 12](#_Toc405134050)

[33. 获得outgoing subscriptions的id 13](#_Toc405134051)

[34. 获得事件的Method 13](#_Toc405134052)

[35. 获得事件的Reason 13](#_Toc405134053)

[36. 获得事件的状态码 13](#_Toc405134054)

[37. 获得请求事件的字段内容 14](#_Toc405134055)

[38. 获得响应事件的字段内容 14](#_Toc405134056)

[39. 从SIP URI中获得用户名 14](#_Toc405134057)

[40. 从SIP URI中获得服务器地址 14](#_Toc405134058)

[41. 从SIP URI中获得用户的Displayname 15](#_Toc405134059)

[42.获得SIP请求消息体 15](#_Toc405134060)

[43.设置网络类型和端口 15](#_Toc405134061)

[44.注册 15](#_Toc405134062)

[45.注册更新 16](#_Toc405134063)

[46.停止注册 16](#_Toc405134064)

[47.发起呼叫 16](#_Toc405134065)

[48.停止呼叫 16](#_Toc405134066)

[49.呼叫应答 17](#_Toc405134067)

[50.呼叫保持 17](#_Toc405134068)

[51.静音 17](#_Toc405134069)

[52.取消静音 17](#_Toc405134070)

[53.通话中创建SIP的session 18](#_Toc405134071)

[54.发送DTMF by SIP INFO 18](#_Toc405134072)

[55. 发送DTMF by RTP 18](#_Toc405134073)

[56. 发送DTMF by Inband 18](#_Toc405134074)

[57. 注册音频设备 19](#_Toc405134075)

[事件类型表 19](#_Toc405134076)

## 1.初始化接口

【函数原型】public native int aminit(int debug\_level);

【功能】初始化

【参数说明】debug\_level : log显示

值：0<debug\_level<8 :显示log；debug\_level <=0 || debug\_level>=8：不显示

【返回值说明】0:成功，其他：失败

【使用方法】在软件启动时初始化

## 2.重置

【函数原型】public native int amreset(int debug\_level);

【功能】初始化

【参数说明】debug\_level : log显示

值：0<debug\_level<8 :显示log；debug\_level <=0 || debug\_level>=8：不显示

【返回值说明】0:成功，其他：失败

【使用方法】重置时使用

## 3.退出注销

【函数原型】public native int amquit();

【功能】注销

【参数说明】无

【返回值说明】0:成功，其他：失败

【使用方法】退出时使用

## 4.设置语音编码

【函数原型】public native int amcodecinfomodify(int pos, int enable, String codec\_name, int mode, int cng, int vbr);

【功能】设置语音编码

【参数说明】pos：编码在sdp中的位置；enable：是否可用，0：不可用，1：可用；codec\_name：编码名称，iLBC/8000，PCMA/8000，PCMU/8000，SILK/8000；mode：模式，默认0;cng:是否使能CNG，默认0；vbr:默认0

【返回值说明】0:成功，其他：失败

【使用方法】不同编码不同调用方式如下：

amcodecinfomodify(0, 1, "iLBC/8000", 30, 0,0);

amcodecinfomodify(1, 1, "PCMU/8000", 0, 0, 0);

amcodecinfomodify(2, 1, "PCMA/8000", 0, 0, 0);

amcodecinfomodify(3, 1, "SILK/8000", 0, 0, 0);

## 5.修改语言编码的比特率

【函数原型】

**public** **native** **int** amcodecattrmodify(**int** ptime, **int** audio\_bitrate);【功能】设置语音编码

【参数说明】ptime:默认0；audio\_bitrate：在3 g模式下,如果音频比特率< = 30,SDP将只包含SDP的低比特率编解码器

【返回值说明】0:成功，其他：失败

【使用方法】

if (wifi\_mode) {

amcodecattrmodify(0, 0);

} else {

amcodecattrmodify(0, 30);

}

## 6.设置语言传输的初始端口

【函数原型】

public native int amoptionsetinitialaudioport(int initialport);

【功能】设置语言传输的初始端口

【参数说明】initialport:端口号，>1024;

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetinitialaudioport(6789);

## 7.设置服务器地址

【函数原型】

public native int amoptionsetipv4forgateway(String ipv4forgateway);

【功能】设置服务器地址

【参数说明】ipv4forgateway：domain地址

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetipv4forgateway(server.domain);

## 8.设置sip via header中rport是否使用

【函数原型】

public native int amoptionenablerport(int enable);

【功能】设置sip中rport是否使用

【参数说明】enable：1：使能，0：禁用

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionenablerport(1);

## 9.设置DNS查找

【函数原型】

public native int amoptionsetdnscapabilities(int dnscapabilities);

【功能】使能NAPTR和SRV来DNS查找

【参数说明】dnscapabilities：>=1：使能，<=0：禁用

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetdnscapabilities (1);

## 10.设置语音传输协议

【函数原型】

public native int amoptionsetaudioprofile(String profile);

【功能】设置语音传输协议

【参数说明】profile：RTP/SAVP

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetaudioprofile("RTP/SAVP");

## 11.设置在呼叫的Session是否超时

【函数原型】

public native int amoptionenablesessiontimers(int sessionexpires);

【功能】设置在呼叫的Session是否超时

【参数说明】enable：1：使能，0：禁用

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionenablesessiontimers(0);

## 12.设置是否使能symmetric RTP streams

【函数原型】

public native int amoptionenablesymmetricrtp(int enable);

【功能】设置是否使能symmetric RTP streams

【参数说明】enable：1：使能，0：禁用

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionselectinsoundcard(0);

## 13.选择输入音频设备

【函数原型】

public native int amoptionselectinsoundcard(int card);

【功能】选择输入音频设备

【参数说明】enable：1：使能，0：禁用

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionselectinsoundcard(0);

## 14.选择输出音频设备

【函数原型】

public native int amoptionselectoutsoundcard(int card);

【功能】选择输入音频设备

【参数说明】enable：1：使能，0：禁用

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionselectoutsoundcard (0);

## 15.WebRtc设置

【函数原型】

**public** **native** **int** amoptionenablewebrtcapm(**int** enable, **int** aecm, **int** aecm\_comfort\_noise, **int** aecm\_routing\_mode,

**int** ns, **int** ns\_level,

**int** agc, **int** agc\_mode, **int** agc\_target\_level\_dbfs, **int** agc\_compression\_gain\_db,

**int** high\_pass\_filter);

【功能】设置webrtc相关参数

【参数说明】enable：1：使能，0：禁用；aecm：[回声抵消](http://www.baidu.com/link?url=ukprnT7ZZ0JGd8lrKZbO_GiQfX97eDpzXNZa6Wgy7Lc0CwyfrpwOvOd4J-S7UtSwK-bJ-XGMDbgCOKT20rjqTmxQGdWKXdfRu-5OMDro1iu" \t "_blank)；ns: 降噪；agc：自动增益；high\_pass\_filter:高通过滤器

【返回值说明】0:成功，其他：失败

【使用方法】

**if** (耳机模式) {

amoptionenablewebrtcapm(0,

0,1, 3, //confort noise , kSpeakerphone

0, 3, //kVeryHigh

0, 1, 15, 9, //kAdaptiveDigital, 15dbfs, 9 compression

0);

}

else{

//根据配置传入参数调用接口，建议全部使能即都为1

amoptionenablewebrtcapm(1,

1,1, 3, //confort noise , kSpeakerphone

1, 3, //kVeryHigh

1, 1, 15, 9, //kAdaptiveDigital, 15dbfs, 9 compression

1);

}

## 16.设置回波消除器

【函数原型】

public native int amoptionenableechocanceller(int enable, int framesize, int taillength);

【功能】设置回波消除器

【参数说明】enable：1：使能，0：禁用；framesize：大小；taillength：长度

【返回值说明】0:成功，其他：失败

【使用方法】

禁用：

amoptionenableechocanceller(0, 0, 0)

使能：

amoptionenableechocanceller(1, 160, 1600 \* 2)

## 17.防抖动设置

【函数原型】

public native int amoptionsetjittermode(int mode);

【功能】防抖动设置

【参数说明】mode: <0：RTP抖动缓冲, 0 :默认抖动缓冲 >=1 : 静态抖动缓冲

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetjittermode(-40);

## 18.设置登录用户信息

【函数原型】

public native int amoptionsetpassword(String realm, String login, String passwd);

【功能】设置登录用户信息

【参数说明】realm：服务器地址，login：登陆名；passwd：密码

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetpassword("","user"," passwd");

## 19.加密设置

【函数原型】

public native int amoptionenableoptionnalencryption(int optionnalencryption);

【功能】设置是否加密

【参数说明】optionnalencryption： <=0：禁用， >=1 ： 使能SRTP加密传输.

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionenableoptionnalencryption(0);

## 20.设置SIP消息中的Supported字段

【函数原型】

public native int amoptionsetsupportedextensions(String supportedextensions);

【功能】设置SIP消息中的Supported字段

【参数说明】supportedextensions：NULL：禁用 Supported字段.!=NULL:使能Supported字段

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetsupportedextensions("100rel, replaces");

## 21.设置SIP消息中的Allowed字段

【函数原型】

public native int amoptionsetallowedmethods(String allowedmethods);

【功能】设置SIP消息中的Allowed字段

【参数说明】allowedmethods：NULL：禁用 Allowed字段.!=NULL:使能Allowed字段

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetallowedmethods("INVITE, ACK, BYE, OPTIONS, CANCEL, INFO, UPDATE, REFER, NOTIFY, MESSAGE");

## 22.设置音频的采样频率

【函数原型】

public native int amoptionsetrate(int rate);

【功能】设置音频的采样频率

【参数说明】rate：频率值,系统支持8000, 16000, 32000 , 48000

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetrate(16000);

## 23. 设置音频增益

【函数原型】

public native int amoptionsetvolumegain(float capturegain, float playbackgain);

【功能】设置音频的增益

【参数说明】capturegain:录音音量；playbackgain：播放音量

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetvolumegain(inputgain, outputgain);

## 24. 设置回音限制

【函数原型】

**public** **native** **int** amoptionsetecholimitation(**int** enabled,

**float** threshold,

**float** speed,

**float** force,

**int** sustain);

【功能】设置回音限制

【参数说明】enabled：0：禁用，1：使能；threshold：0.0-1.0，当扬声器的信号能量大于threshold值时，回音限制就会生效，自动降低Mic音量；speed：0.0-1.0，speed系数表示新增量对当前增量影响的速度快慢；force：force值越高，削减的增益越深；sustain：

单位毫秒，当扬声器增量低于threshold值超过sustain时间时，将恢复默认增量

【返回值说明】0:成功，其他：失败

【使用方法】在系统初始化时可以设置amoptionsetecholimitation(1, 0.02, 0.02, 50.0, 300);

## 25. 设置事件等待时间

【函数原型】

public native long ameventwait(int tvs, int tvms);

【功能】设置事件等待时间

【参数说明】tvs:秒数；tvms：毫秒数

【返回值说明】0:超时，>0：事件地址

【使用方法】在获得消息的循环中使用ameventwait(0, 200);

## 26. 释放事件

【函数原型】

public native void ameventrelease(long evt);

【功能】释放事件

【参数说明】evt：事件地址

【返回值说明】无

【使用方法】释放事件时调用

## 27. 获得事件类型

【函数原型】

public native int ameventgettype(long evt);

【功能】获得事件类型

【参数说明】evt：事件地址

【返回值说明】<0:错误；>0: 事件类型，数值见[事件类型表](#_SIP事件类型)

【使用方法】在事件处理中使用ameventgettype(evt);

## 28. 获得transactions的id

【函数原型】

public native int ameventgettid(long evt);

【功能】获得transactions的id

【参数说明】evt：事件地址

【返回值说明】<0:失败，>=0：transactions的id

【使用方法】在事件处理中使用ameventgettid (evt);

## 29. 获得SIP calls的id

【函数原型】

public native int ameventgetcid(long evt);

【功能】获得SIP calls的id

【参数说明】evt：事件地址

【返回值说明】<0:失败，>=0：SIP calls的id

【使用方法】在事件处理中使用ameventgetcid (evt);

## 30. 获得SIP dialogs的id

【函数原型】

public native int ameventgetdid (long evt);

【功能】获得SIP dialogs的id

【参数说明】evt：事件地址

【返回值说明】<0:失败，>=0：SIP dialogs的id

【使用方法】在事件处理中使用ameventgetdid (evt);

## 31. 获得registration的id

【函数原型】

public native int ameventgetrid (long evt);

【功能】获得registration的id

【参数说明】evt：事件地址

【返回值说明】<0:失败，>=0：registration的id

【使用方法】在事件处理中使用ameventgetrid (evt);

## 32. 获得incoming subscriptions的id

【函数原型】

public native int ameventgetnid (long evt);

【功能】获得incoming subscriptions的id

【参数说明】evt：事件地址

【返回值说明】<0:失败，>=0：incoming subscriptions的id

【使用方法】在事件处理中使用ameventgetnid (evt);

## 33. 获得outgoing subscriptions的id

【函数原型】

public native int ameventgetsid (long evt);

【功能】获得outgoing subscriptions的id

【参数说明】evt：事件地址

【返回值说明】<0:失败，>=0：outgoing subscriptions的id

【使用方法】在事件处理中使用ameventgetsid (evt);

## 34. 获得事件的Method

【函数原型】

public native String ameventgetmethod(long evt);

【功能】获得事件的Method

【参数说明】evt：事件地址

【返回值说明】NULL:失败，其他：SIP消息中的Method

【使用方法】在事件处理中使用ameventgetmethod (evt);

## 35. 获得事件的Reason

【函数原型】

public native String ameventgetreason(long evt);

【功能】获得事件的Reason

【参数说明】evt：事件地址

【返回值说明】NULL:失败，其他：SIP消息中的Reason

【使用方法】在事件处理中使用ameventgetreason (evt);

## 36. 获得事件的状态码

【函数原型】

public native int ameventgetstatuscode(long evt);

【功能】获得事件的状态码

【参数说明】evt：事件地址

【返回值说明】0:失败，其他：SIP消息中的状态码

【使用方法】在事件处理中使用ameventgetstatuscode (evt);

## 37. 获得请求事件的字段内容

【函数原型】

public native String ameventgetrequestheader(long evt, String hname, int n);

【功能】获得请求事件的字段内容

【参数说明】evt：事件地址；hname：SIP字段关键字；n：位置

【返回值说明】NULL:失败，其他：请求事件的字段内容

【使用方法】在事件处理中使用ameventgetrequestheader(evt, "to", 0);

## 38. 获得响应事件的字段内容

【函数原型】

public native String ameventgetresponseheader (long evt, String hname, int n);

【功能】获得响应事件的字段内容

【参数说明】evt：事件地址；hname：SIP字段关键字；n：位置

【返回值说明】NULL:失败，其他：响应事件的字段内容

【使用方法】在事件处理中使用ameventgetresponseheader(evt,"content-type", 0);

## 39. 从SIP URI中获得用户名

【函数原型】

public native String amurigetusername(String mSipUri);

【功能】从SIP URI中获得用户名

【参数说明】mSipUri：SIP的Uri地址

【返回值说明】NULL:失败，其他：用户名

【使用方法】一般从to字段中获得uri然后调用获得username，amurigetusername(sip\_uri);

## 40. 从SIP URI中获得服务器地址

【函数原型】

public native String amurigetdomain (String mSipUri);

【功能】从SIP URI中获得服务器地址

【参数说明】mSipUri：SIP的Uri地址

【返回值说明】NULL:失败，其他：服务器地址

【使用方法】先获得uri然后调用获得username，amurigetdomain (sip\_uri);

## 41. 从SIP URI中获得用户的Displayname

【函数原型】

public native String amurigetdisplayname (String mSipUri);

【功能】从SIP URI中获得用户的Displayname

【参数说明】mSipUri：SIP的Uri地址

【返回值说明】NULL:失败，其他：用户的Displayname

【使用方法】先获得uri然后调用获得username，amurigetdisplayname (sip\_uri);

## 42.获得SIP请求消息体

【函数原型】

public native byte[] ameventgetrequestbody(long evt, int n);

【功能】获得SIP请求消息体

【参数说明】evt：事件地址；n：位置

【返回值说明】NULL:失败，其他：消息体数组

【使用方法】ameventgetrequestbody(evt,0);

## 43.设置网络类型和端口

【函数原型】

public native int amnetworkstart(String transport, int port);

【功能】设置网络类型和端口

【参数说明】transport："UDP", "TCP", "TLS"； port：端口号，0：随机

【返回值说明】0:成功，其他：失败

【使用方法】amnetworkstart (“UDP”,0);

## 44.注册

【函数原型】

public native int amregisterstart(String identity, String proxy, String outboundproxy, int expires);

【功能】注册

【参数说明】identity：注册标识；proxy：注册服务器，（sip:服务器地址）；outboundproxy：outbound地址；expires：注册间隔时间

【返回值说明】<0：失败,>=0:成功，注册id

【使用方法】amregisterstart("","sip:test.com", "outbound.com", 3600);

## 45.注册更新

【函数原型】

public native int amregisterrefresh(int rid, int expires);

【功能】注册更新

【参数说明】rid：注册Id；expires：注册间隔时间

【返回值说明】<0：失败,>=0:成功

【使用方法】amregisterrefresh(registration\_id,3600);

## 46.停止注册

【函数原型】

public native int amregisterstop(int rid);

【功能】停止注册

【参数说明】rid：注册Id；

【返回值说明】<0：失败,>=0:成功

【使用方法】退出系统时调用amregisterstop(registration\_id);

## 47.发起呼叫

【函数原型】

public native int amsessionstart(String identity, String target, String proxy, String outboundproxy);

【功能】发起呼叫

【参数说明】identity：呼叫标识，""；target:被叫, "sip:callee@domain"；proxy：代理服务器，"sip: proxy "；

outboundproxy：outbound代理服务器，"<sip:outboundproxy;lr>"

【返回值说明】<0：失败,>=0:成功,call id;

【使用方法】发起呼叫时调用amsessionstart("","sip:callee@domain",

"sip: proxy ","<sip:outboundproxy;lr>");

## 48.停止呼叫

【函数原型】

public native int amsessionstop(int cid, int did, int code);

【功能】停止呼叫

【参数说明】cid：call Id；did ：dialog Id；code：状态码

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionstop(cid,did, 486);

## 49.呼叫应答

【函数原型】

public native int amsessionanswer(int tid, int did, int code, int enable\_audio);

【功能】呼叫应答

【参数说明】cid：transaction Id；did ：dialog Id；code：状态码；enable\_audio：是否带有音频信息

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionanswer(tid,did,200,1);//应答invite的200 OK消息

## 50.呼叫保持

【函数原型】

public native int amsessionhold(int did, String holdmusic);

【功能】呼叫保持

【参数说明】did：dialog id；holdmusic：wav文件的绝对路径

【返回值说明】<0：失败,>=0:成功

【使用方法】呼叫保持(did,”f:\wav\test.wav”);

## 51.静音

【函数原型】

public native int amsessionmute(int did);

【功能】通话中静音

【参数说明】did：dialog id

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionmute (did);

## 52.取消静音

【函数原型】

public native int amsessionunmute(int did);

【功能】取消静音

【参数说明】did：dialog id

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionmute (did);

## 53.通话中创建SIP的session

【函数原型】

public native int amsessionanswerrequest(int tid, int did, int code);

【功能】通话中创建SIP的session ，对应EXOSIP\_CALL\_MESSAGE\_NEW处理

【参数说明】tid：transaction Id；did ：dialog Id；code：状态码

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionanswerrequest (tid,did,200);

## 54.发送DTMF by SIP INFO

【函数原型】

public native int amsessionsenddtmfwithduration(int did, String dtmf, int duration);

【功能】注册更新

【参数说明】did：dialog id；dtmf：dtmf码；duration：持续时间（微秒）

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionsenddtmfwithduration(did, dtmf, 480);

## 55. 发送DTMF by RTP

【函数原型】

public native int amsessionsendrtpdtmf(int did, String dtmf);

【功能】注册更新

【参数说明】did：dialog id；dtmf：dtmf码

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionsendrtpdtmf(did, dtmf);

## 56. 发送DTMF by Inband

【函数原型】

public native int amsessionsendinbanddtmf(int did, String dtmf);

【功能】注册更新

【参数说明】did：dialog id；dtmf：dtmf码

【返回值说明】<0：失败,>=0:成功

【使用方法】amsessionsendinbanddtmf (did, dtmf);

public native void registeraudio();

## 57. 注册音频设备

【函数原型】

public native void registeraudio();

【功能】注册音频设备

【参数说明】无

【返回值说明】无

【使用方法】系统启动时调用registeraudio();

## 事件类型表

|  |  |  |
| --- | --- | --- |
| SIP事件类型 | 值 | 注释 |
| *EXOSIP\_REGISTRATION\_SUCCESS* | 0 | 注册成功 |
| *EXOSIP\_REGISTRATION\_FAILURE* | 1 | 注册失败 |
| *EXOSIP\_CALL\_INVITE* | 2 | 发起一个新呼叫 |
| *EXOSIP\_CALL\_REINVITE* | 3 | 在呼叫中发送Invite消息 |
| *EXOSIP\_CALL\_NOANSWER* | 4 | 无人应答 |
| *EXOSIP\_CALL\_PROCEEDING* | 5 | 处理PROCEEDING消息 |
| *EXOSIP\_CALL\_RINGING* | 6 | RINGING消息，对方振铃 |
| *EXOSIP\_CALL\_ANSWERED* | 7 | 对方应答，开始通话 |
| *EXOSIP\_CALL\_REDIRECTED* | 8 | 重定向 |
| *EXOSIP\_CALL\_REQUESTFAILURE* | 9 | 请求失败 |
| *EXOSIP\_CALL\_SERVERFAILURE* | 10 | 服务器错误 |
| *EXOSIP\_CALL\_GLOBALFAILURE* | 11 | 全局错误 |
| *EXOSIP\_CALL\_ACK* | 12 | 接收到回复Invite的200ok消息的ACK消息 |
| *EXOSIP\_CALL\_CANCELLED* | 13 | 取消呼叫 |
| *EXOSIP\_CALL\_MESSAGE\_NEW* | 14 | 通话中新的呼入请求 |
| *EXOSIP\_CALL\_MESSAGE\_PROCEEDING* | 15 | 通话中正在处理消息 |
| *EXOSIP\_CALL\_MESSAGE\_ANSWERED* | 16 | 通话中200ok应答 |
| *EXOSIP\_CALL\_MESSAGE\_REDIRECTED* | 17 | 通话中重定向 |
| *EXOSIP\_CALL\_MESSAGE\_REQUESTFAILURE* | 18 | 通话中请求错误 |
| *EXOSIP\_CALL\_MESSAGE\_SERVERFAILURE* | 19 | 通话中服务器错误 |
| *EXOSIP\_CALL\_MESSAGE\_GLOBALFAILURE* | 20 | 通话中全局错误 |
| *EXOSIP\_CALL\_CLOSED* | 21 | 通话结束 |
| *EXOSIP\_CALL\_RELEASED* | 22 | 释放呼叫资源 |
| *EXOSIP\_MESSAGE\_NEW* | 23 | 接收到新消息应答 |
| *EXOSIP\_MESSAGE\_PROCEEDING* | 24 | 接收到正在处理消息应答 |
| *EXOSIP\_MESSAGE\_ANSWERED* | 25 | 接收到应答 |
| *EXOSIP\_MESSAGE\_REDIRECTED* | 26 | 接收到重定向应答 |
| *EXOSIP\_MESSAGE\_REQUESTFAILURE* | 27 | 接收到请求失败应答 |
| *EXOSIP\_MESSAGE\_SERVERFAILURE* | 28 | 接收到服务器错误应答 |
| *EXOSIP\_MESSAGE\_GLOBALFAILURE* | 29 | 接收到全局错误应答 |
| *EXOSIP\_SUBSCRIPTION\_NOANSWER* | 30 | SUBSCRIBE消息无应答 |
| *EXOSIP\_SUBSCRIPTION\_PROCEEDING* | 31 | 正在处理SUBSCRIBE消息 |
| *EXOSIP\_SUBSCRIPTION\_ANSWERED* | 32 | SUBSCRIBE消息应答 |
| *EXOSIP\_SUBSCRIPTION\_REDIRECTED* | 33 | SUBSCRIBE消息重定向 |
| *EXOSIP\_SUBSCRIPTION\_REQUESTFAILURE* | 34 | SUBSCRIBE消息请求失败 |
| *EXOSIP\_SUBSCRIPTION\_SERVERFAILURE* | 35 | SUBSCRIBE消息服务器错误 |
| *EXOSIP\_SUBSCRIPTION\_GLOBALFAILURE* | 36 | SUBSCRIBE消息全局错误 |
| *EXOSIP\_SUBSCRIPTION\_NOTIFY* | 37 | 发起NOTIFY消息 |
| *EXOSIP\_IN\_SUBSCRIPTION\_NEW* | 38 | 收到新的SUBSCRIBE消息 |
| *EXOSIP\_NOTIFICATION\_NOANSWER* | 39 | NOTIFY消息无应答 |
| *EXOSIP\_NOTIFICATION\_PROCEEDING* | 40 | 正在处理NOTIFY消息 |
| *EXOSIP\_NOTIFICATION\_ANSWERED* | 41 | NOTIFY消息应答 |
| *EXOSIP\_NOTIFICATION\_REDIRECTED* | 42 | NOTIFY消息重定向 |
| *EXOSIP\_NOTIFICATION\_REQUESTFAILURE* | 43 | NOTIFY消息请求失败 |
| *EXOSIP\_NOTIFICATION\_SERVERFAILURE* | 44 | NOTIFY消息服务器错误 |
| *EXOSIP\_NOTIFICATION\_GLOBALFAILURE* | 45 | NOTIFY消息全局错误 |