A-SDK 开发手册

V4.0.0.1

易通星云（北京）科技发展有限公司

官网地址：www.kaifakuai.com

官网 QQ 群：445880047

微信公众号：开发快

文档修改记录

版本号	发布日期	描述	作者
4.0.0.0	2017.02.04	新建文档	Hehui
4.0.0.1	2017.02.23	修改文档，对应 sdk4.0.0.1	Hehui

1 .功能简介

本 SDK 为 Java 应用、Android 应用提供相关 API 调用，与 iLink 其它平台的 SDK 实现通信。通过 SDK 中的相关功能接口实现消息发送、文件传送、好友管理等功能，并支持 SSL 安全访问模式。

1.1缩略词

名词	描述
iLink	易通的物联网/智能硬件云服务平台(www.kaifakuai.com)
AppKey	应用标识码，当开发者需要为一款智能产品开发应用（包括终端与设备）时，申请生成，应用开发时需要填入
SecretKey	应用安全识别码，在调用一些管理接口时要填入
UID	平台标识码（系统唯一，相当于账号），应用开发时需要填入

表 1-1 缩略词解释表

2 .运行环境

JVM：最低需 JDK1.6，推荐 JDK1.7 的编译环境。

3 .SDK 使用流程

SDK 的所有功能都通过 ISDKContext 实现，使用流程图如下：

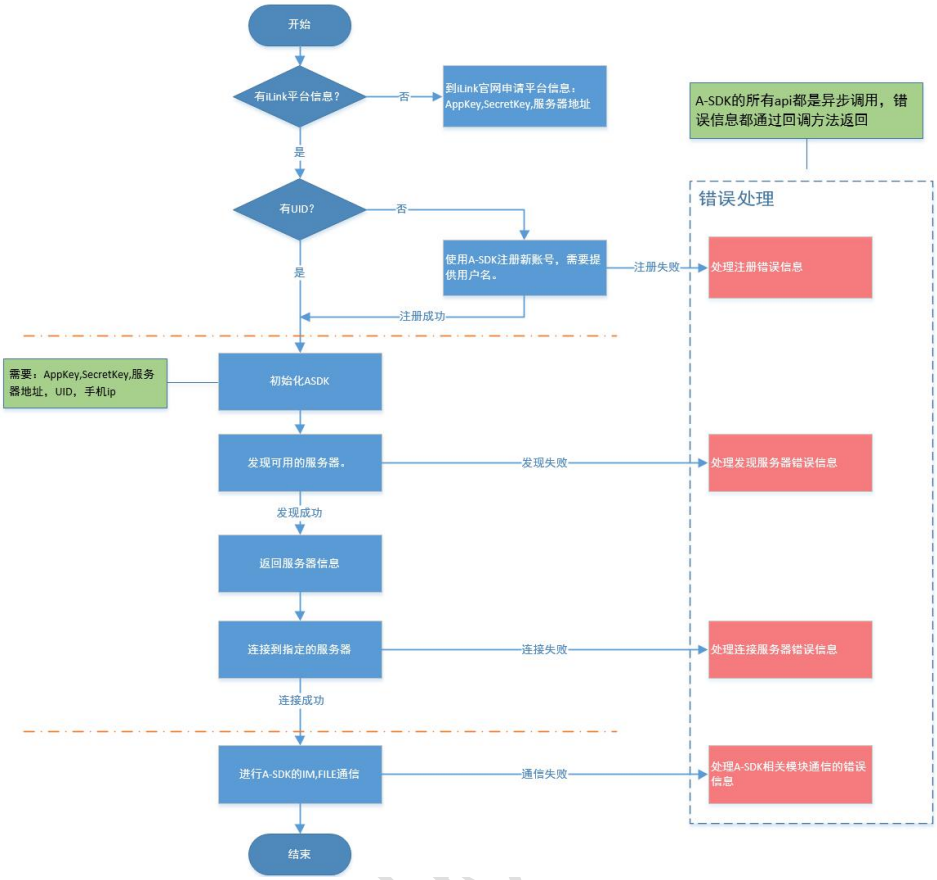

图 3-1 ASDK 使用流程图

4 .SDK 接口说明

4.1 注册新用户

在使用 ASDK 的其它功能之前，首先需要注册一个用户，使用 SDKContextManager 注册一个新用户，注册之前需要向平台申请 AppKey, SecretKey，注册新用户示例代码：

```
String username = "用户名";
String nickname="昵称"
String serverHost = "平台主机地址";
String appKey = "申请的 appkey";
String secretKey = "申请的 secretKey";
int serverPort=服务器端口号
SDKContextManager.addUser(username, nickname,serverHost,serverPort
      appKey, secretKey,
      new IFriendsActionListener() {

            @Override
            public void onSuccess() {
                  //该方法不会调用
            }

            @Override
            public void onFailure(final ErrorInfo arg0) {
                  //注册用户失败

            }

            @Override
            public void onResultData(final Object arg0) {
                  UserInfo userInfo = (UserInfo) arg0;
                  //注册用户成功
            }
```

示例代码 4-1 注册新用户

4.2 ISDKContext 上下文

ISDKContext 上下文定义了 SDK 的所有操作，详细接口说明参见 附录一。

4.3 ISDKContextCallback 全局回调

在执行 SDK 的操作之前，必须首先设置全局回调，ISDKContextCallback 定义了 ISDKContext 的全局回调函数，包括发现服务器结果，消息接收，文件接收，账户状态变化，服务器连接状态等。包含的接口如下：

```
void onServer(Server svr);
void onMessage(MessageType type, String topic, Message msg);
void onFileReceived(String senderId, DocumentInfo documentInfo);
void onPeerState(String uid, String statusCode);
void onBroken(Server svr, int errorCode, String reason);
```

每个接口的说明，详见附录一 API Reference。

4.4 初始化 ISDKContext

使用 SDKContextManager 创建 ISDKContext 实例，代码示例：

```
SDKContextParameters sdkContextParameters = new SDKContextParameters();
sdkContextParameters.setUid(uid);
sdkContextParameters.setAppKey("your Appkey");
sdkContextParameters.setSecretKey("your SecretKey");
sdkContextParameters.setBlanceServerAddress("server address");
sdkContextParameters.setBlanceServerPort(server port);
sdkContextParameters.setLocalIp(ip);//如果只使用外网模式，可以不填该字段,使用内网必须填入此字段
ISDKContext sdkContext = SDKContextManager.createContext(sdkContextParameters);
```

示例代码 4-2 实例化 SDKContext

SDKContextParameters 指定了 ISDKContext 需要的参数，说明如下：

参数	描述
Blance Server Address	负载服务器的地址。
Local Ip	手机的 ip 地址，只用于内网发现服务器。如果只使用外网可以忽略此字段。

表 4-1 SDKContext 参数说明表

使用 ISDKContext 的 setCallback 方法设置 ISDKContextCallback 全局回调函数，代码示例：

```
mSdkContext.setCallback(new ISDKContextCallback() {

        @Override
        public void onServer(Server svr) {
            // 发现可用的服务器
        }

        @Override
        public void onMessage(MessageType type, String topic, Message message) {
            //接收到新消息
        }

        @Override
        public void onPeerState(String uid, String stateCode) {
            //用户状态改变
        }

        @Override
        public void onFileReceived(String senderUid, DocumentInfo documentInfo) {
            // 有新的文件需要下载
        }

        @Override
        public void onBroken(Server svr, int errorCode, String reason) {
            //已经和服务器断开连接
        }
    });
}
```

示例代码 4-3 设置 SDK 全局回调

4.5 发现服务器

iLink 支持两种服务器：内网服务器和外网服务器。

使用 ISDKContext 发现可用的服务器，支持默认发现模式和自定义发现模式。发现的服务器通过 ISDKContextCallback 的 onServer()方法返回。

4.5.1 默认发现模式

使用 SDK 默认规则发现可用的服务器，通常使用该模式。示例代码：

```
mSdkContext.discoverSvrs(10, new IActionListener() {

        @Override
        public void onSuccess() {
            //正在扫描服务器，
            //扫描到的服务器通过 sdk 全局回调返回。
        }

        @Override
        public void onFailure(ErrorInfo errorInfo)
            //扫描服务器失败
        }
});
```

示例代码 4-4 默认模式发现可用服务器

4.5.2 自定义发现模式

使用自定义规则发现可用的服务器，注意**需要服务器支持自定义规则**，此模式用于服务器定义自己的规则，防止非法客户端发现自己。示例代码：

```
DiscoverOptions opt = new DiscoverOptions();
opt.setContent("app custom rule".getBytes()); //服务器定义的规则
mSdkContext.discoverServers(10, opt, new IActionListener() {

        @Override
        public void onSuccess() {
            //正在扫描服务器
            //扫描到的服务器通过 sdk 全局回调返回。
        }

        @Override
        public void onFailure(ErrorInfo errorInfo)
            //扫描服务器失败
        }
});
```

示例代码 4-5 自定义模式发现可用服务器

4.6 连接服务器

连接服务器时，需要通过 ConnectOptions 设置连接参数，然后使用 ISDKContext 的 connect 方法连接到指定的服务器，示例代码：

```
ConnectOptions connectOptions = new ConnectOptions(); //连接参数
connectOptions.setConnectionTimeout(10); //连接超时时间
connectOptions.setKeepAliveInterval((short) 60); //与服务器的保活时间间隔
connectOptions.setCleanSession(true); //不保存离线消息

// svr 为 discover 发现到的可用服务器
mSdkContext.connect(svr, connectOptions, new IActionListener() {

    @Override
    public void onSuccess() {
        //连接服务器成功
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //连接服务器失败
    }
});
```

示例代码 4-6 连接到指定服务器

4.7 发送消息

SDK 支持点对点消息和群发消息两种模式。

4.7.1 点对点消息

使用 ISDKContext 的 chatTo 方法发送点对点消息，示例代码：

```
Message message = new Message();
message.setPayload("Hello iLink".getBytes("UTF-8"));
mSdkContext.chatTo("412", message, new IActionListener() {

    @Override
    public void onSuccess() {
        //代表消息成功发送到服务器，不表示接收方已经收到消息。
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //发送失败
    }
});
```

示例代码 4-7 发送消息给指定用户

4.7.2 发送主题消息

发送主题消息是把一条消息发送到一个主题(topic)中，所有订阅(subscribe)了该主题的用户都会接收到该消息。

使用 ISDKContext 的 publish 方法发送群发消息，示例代码：

```
Message message = new Message();
message.setPayload("Hello everyone!".getBytes("UTF-8"));
mSdkContext.publish("topic", QOS.Qos_1, message, new IActionListener() {

    @Override
    public void onSuccess() {
        //代表消息成功发送到服务器，不表示接收方已经收到消息。
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //发送失败
    }
});
```

示例代码 4-8 发送主题消息

注意：只有订阅了对应主题的用户才能接收到群发消息。

4.7.3 发送群消息

使用 ISDKContext 的 publishToGroup 方法来发送群消息，示例代码：

```
Message message = new Message();
message.setPayload("Hello everyone!".getBytes("UTF-8"));
mSdkContext.publishToGroup("groupId", message, new IActionListener() {

    @Override
    public void onSuccess() {
        //代表消息成功发送到服务器，不表示接收方已经收到消息。
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //发送失败
    }
});
```

示例代码 4-9 发送群消息

4.8 订阅主题

使用 ISDKContext 的 subscribe 方法订阅主题，示例代码：

```
mSdkContext.subscribe("topic", Qos.Qos_1,new IActionListener() {

    @Override
    public void onSuccess() {
        //订阅主题成功
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //订阅主题失败
    }
});
```

示例代码 4-10 订阅主题

4.9 取消订阅主题

使用 ISDKContext 的 unsubscribe 方法取消已经订阅的主题，示例代码：

```
mSdkContext.unsubscribe("topic", new IActionListener() {

    @Override
    public void onSuccess() {
        //订阅主题成功
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //订阅主题失败
    }
});
```

示例代码 4-11 取消已经订阅的主题

4.10 获取离线消息

如果用户连接服务器时，指定了允许接收离线消息，即 ConnectOptions 设置了 cleanSession 为 false，那么当用户退出登录后，服务器会保存其他用户发送给该用户的消息，即离线消息。

使用 ISDKContext 的 requestOfflineMessage 方法获取服务器上保存的离线消息，示例代码：

```
mSdkContext.requestOfflineMessage();
```

示例代码 4-12 获取所有离线消息

注意：服务器只接收第一次请求，多次调用服务器会忽略。当用户离线后再次登录，需要重新调用该方法获取离线期间服务器保存的该用户的离线消息。

4.11 好友群组管理

好友群组管理功能必须在成功连接服务器后才能使用。

4.11.1 添加好友

使用 ISDKContext 的 addBuddy 方法添加好友，示例代码：

```
mSdkContext.addBuddy("friendId",isnotify,new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //添加好友成功
    }

    @Override
    public void onResultData(Object data) {
        //该方法不会调用
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //添加好友失败
    }
});
```

示例代码 4-13 增加好友

4.11.2 删除好友

使用 ISDKContext 的 removeBuddy 方法删除好友，示例代码：

```
mSdkContext.removeBuddy("friendId", isNotify,new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //删除好友成功
    }

    @Override
    public void onResultData(Object data) {
        //该方法不会调用
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //删除好友失败
    }
});
```

示例代码 4-14 删除好友

4.11.3 查询所有好友

使用 ISDKContext 的 getBuddies 方法查询所有好友，示例代码：

```
mSdkContext.getBuddies(new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //该方法不会调用
    }

    @Override
    public void onResultData(Object data) {
        List<UserInfo> userInfoList = (List<UserInfo>)data;
        // 返回好友列表
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //查询所有好友失败
    }
});
```

示例代码 4-15 查询所有好友

4.11.4 创建群

使用 ISDKContext 的 createGroup 方法创建群，示例代码：

```
List<String> memberUidList = new ArrayList<String>();
memberUidList.add(“friendId1”);
memberUidList.add(“friendId2”);
mSdkContext.createGroup("groupName", memberUidList, new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //该方法不会调用
    }

    @Override
    public void onResultData(Object data) {
        GroupInfo groupInfo = (GroupInfo)data;
        // 返回已经创建成功的群信息
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //创建群失败
    }
});
```

示例代码 4-16 创建群

4.11.5 查询所有群

使用 ISDKContext 的 getGroups 方法查询所有群，示例代码：

```
mSdkContext.getGroups(new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //该方法不会调用
    }

    @Override
    public void onResultData(Object data) {
        List<GroupInfo> groupInfoList = (List<GroupInfo>)data;
        // 返回该用户的所有群信息
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //查询群消息失败
    }
});
```

示例代码 4-17 查询所有群

4.11.6 查询群成员信息

使用 ISDKContext 的 getGroupMembers 方法查询群成员信息，示例代码：

```
mSdkContext.getGroupMembers("1", new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //该方法不会调用
    }

    @Override
    public void onResultData(Object data) {
        List<UserInfo> userInfoList = (List<UserInfo>)data;
        // 返回群成员列表
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //查询群成员失败
    }
});
```

示例代码 4-18 查询群成员列表

4.11.7 添加群成员

使用 ISDKContext 的 addGroupMember 方法为群添加新成员，示例代码：

```
List<String> memberUidList = new ArrayList<String>();
memberUidList.add(“423”);
memberUidList.add(“424”);
mSdkContext.addGroupMembers("1", memberUidList, new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //添加群成员成功
    }

    @Override
    public void onResultData(Object data) {
        //该方法不会调用
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //查询群消息失败
    }
});
```

示例代码 4-19 添加用户到群

4.11.8 删除群成员

使用 ISDKContext 的 removeGroupMember 方法删除群成员，示例代码：

```
List<String> memberUidList = new ArrayList<String>();
memberUidList.add(“423”);
memberUidList.add(“424”);
mSdkContext.removeGroupMembers("1", memberUidList, new IFriendsActionListener() {

    @Override
    public void onSuccess() {
        //删除群成员成功
    }

    @Override
    public void onResultData(Object data) {
        //该方法不会调用
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //删除群成员失败
    }
});
```

示例代码 4-20 从群中删除某成员

4.11.9 退出群

使用 ISDKContext 的 exitGroup 方法断开已经连接的服务器，示例代码：

```
mSdkContext.exitGroup(groupId, new IActionListener() {

    @Override
    public void onSuccess() {
        //退出群成功
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //退出群成功
    }
});
```

示例代码 4-21 退出群

4.12 断开连接

使用 ISDKContext 的 disconnect 方法断开已经连接的服务器，示例代码：

```
mSdkContext.disconnect(svr, new IActionListener() {

    @Override
    public void onSuccess() {
        //断开连接成功
    }

    @Override
    public void onFailure(ErrorInfo errorInfo) {
        //断开连接失败
    }
});
```

示例代码 4-22 断开连接

4.13 销毁 ISDKContext

当不再使用 SDK 时，使用 SDKContextManager 的 destroyContext 方法销毁 SDK，示例代码：

```
SDKContextManager.destroyContext(mSdkContext);
```

示例代码 4-23 销毁 SDK 实例

注意：调用 disconnect 方法断开连接后，可以调用 connect 方法再次连接服务器，但是一旦调用 destroyContext 方法销毁 SDK 后，SDK 的所有方法都不能再使用，必须重新初始化 SDK。

4.14 错误处理

SDK 的所有操作都是异步方式，操作结果和错误信息都通过 IActionListener 和 IFriendsActionListener 回调，具体错误信息参见 ErrorInfo。

附录一 API Reference

com.beidouapp.et

SDKContextManager

createContext

createContext
public static ISDKContext createContext(SDKContextParameters parameters) 创建 ISDKContext 实例。 **参数:** parameters - SdkContext 配置参数。 **抛出:** IllegalArgumentException - 如果参数为 null。

destroyContext

destroyContext
public static void destroyContext(ISDKContext sdkContext) 销毁 ISDKContext 实例。 **抛出:** IllegalArgumentException - 如果参数为 null。

addUser

addUser
public static void addUser(String username,String nickname, String serverHost,String server

String appKey,
String secretKey,
IFriendsActionListener listener)

注册新用户.

参数:

username - 用户名

nickname - 昵称

serverHost - 注册服务器的主机地址

serverPort - 注册服务器的端口号

appKey - 平台分配的 APPKey

secretKey - 平台分配的 secretKey

listener - 注册用户成功，回调 IFriendsActionListener.onResultData(Object) ，参数类型是 UserInfo，表示注册成功的用户信息； 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

ISDKContext

public interface **ISDKContext**

A-SDK 的所有操作，如发送消息，发送文件等。

方法详细说明：

setContextParameters

setContextParameters

void setContextParameters(SDKContextParameters parameters)

设置 A-SDK 的配置参数。

参数:

parameters - 配置参数。

getContextParameters

getContextParameters

SDKContextParameters getContextParameters()

获取 A-SDK 的配置参数。

返回:

配置参数。

discoverServers

discoverServers

void discoverServers(int timeout,
IActionListener listener)

扫描服务器，扫描结果通过 ISDKContextCallback.onServer(Server)返回。

在调用该方法之前，需要先设置 SDK 回调方法，setCallback(ISDKContextCallback)。

参数:

timeoutSencond - 扫描的最大持续时间，如果超时时间内扫描到服务器，那么通过 ISDKContextCallback.onServer(Server)通知，否则通过 IActionListener#onFailure(int) 通知。超时时间只能是 5 ~ 30 秒之间。

listener - 操作结果回调。 IActionListener.onSuccess() 只表示发现操作成功，不代表一定会有服务器返回。

抛出:

IllegalArgumentException - 如果参数为 null。

discoverServers

discoverServers

void discoverServers(int timeoutSencond,
DiscoverOptions opt,
IActionListener listener)

扫描服务器，扫描结果通过 ISDKContextCallback.onServer(Server)返回。

在调用该方法之前，需要先设置 SDK 回调方法，
setCallback(ISDKContextCallback)。

参数:

timeoutSencond - 扫描的最大持续时间，如果超时时间内扫描到服务器，那么通过 ISDKContextCallback.onServer(Server)通知，否则通过 IActionListener#onFailure(int) 通知。超时时间只能是 5 ~ 30 秒之间。

opt - 自定义扫描参数 ,可以为 null。

listener - 操作结果回调。 IActionListener.onSuccess() 只表示发现操作成功，不代表一定会有服务器返回。

抛出:

IllegalArgumentException - 如果参数为 null。

connect

connect

void connect(Server svr,
ConnectOptions opt,
IActionListener listener)

连接到服务器，异步连接，操作结果通过 listener 返回。

参数:

svr - 服务器。

opt - 连接参数。

listener - 连接结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

disconnect

disconnect

void disconnect(Server svr,IActionListener listener)

断开已经连接的服务器。

参数:

svr - 已经连接的服务器。

listener - 操作结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

getUserState

getUserState

void getUserState(String uid , StatusListener listener)

查询用户状态。

参数:

uid - 用户的 uid。

listener - 用户状态结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

chatTo

chatTo

void chatTo(String receiverUid, Message msg,IActionListener listener)

发送消息到指定用户。

参数:

receiverUid - 消息接收者的 uid。

msg - 要发送的消息。

抛出:

IllegalArgumentException - 如果参数为 null。

publish

publish

void publish(String topic,Qos qos,Message msg, IActionListener listener)

发布消息给订阅了主题的所有用户。

参数:

topic - 主题

qos - 消息质量级别枚举

msg - 消息

listener - 操作结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

publishToGroup

publishToGroup

```
void publish(String groupId,
             Message msg,
             IActionListener listener)
```

发布消息给订阅了主题的所有用户。

参数:

topic - 群 ID

msg - 消息

listener - 操作结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

subscribe

subscribe

```
void subscribe(String topic,Qos qos,
             IActionListener listener)
```

订阅主题。

参数:

topic - 主题

qos - 消息质量级别枚举

listener - 操作结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

unsubscribe

unsubscribe

void unsubscribe(String topic,
IActionListener listener)

取消已经订阅的主题。

参数:

topic - 主题

listener - 操作结果回调。

抛出:

IllegalArgumentException - 如果参数为 null。

requestOfflineMessage

requestOfflineMessage

void requestOfflineMessage()

获取所有离线消息。

通过 ISDKContextCallback.onMessage 方法返回离线消息。

subUserState

subUserState

void subUserState(String uid,
IActionListener listener)

订阅用户在线状态。

用户状态变化通过 ISDKContextCallback.onPeerState(String, String) 返回。

参数:

uid - 关注的用户 uid。

listener - 操作是否成功。

抛出:

IllegalArgumentException - 如果参数为 null。

unSubUserState

unSubUserState

void unSubUserState(String uid,
IActionListener listener)

取消订阅用户在线状态。

用户状态变化通过 ISDKContextCallback.onPeerState(String, String) 返回。

参数:

uid - 关注的用户 uid。

listener - 操作是否成功。

抛出:

IllegalArgumentException - 如果参数为 null。

fileTo

fileTo

void fileTo(String receiverId,
String fileFullName,String desc,
FileCallBack callBack)

主动发送文件。

参数:

receiverId - 接收文件的用户 ID。

fileFullName - 文件全路径名。

desc - 文件描述信息

callBack - 文件操作回调接口。

抛出:

IllegalArgumentException - 如果参数为 null。

uploadFile

uploadFile

void uploadFile(String filePath,FileCallBack callBack)

上传文件到服务器。

参数:

filePath- 本地文件保存路径。

callBack - 文件操作回调接口。

抛出:

IllegalArgumentException - 如果参数为 null。

downloadFile

downloadFile

void downloadFile(DocumentInfo documentInfo,
String saveFilePath,
FileCallBack callBack)

从服务器下载对方发送给自己的文件。

参数:

documentInfo - 文件信息。

saveFilePath - 本地文件保存路径。

callBack - 文件操作回调接口。

抛出:

IllegalArgumentException - 如果参数为 null。

addBuddy

addBuddy

void addBuddy(String buddyUid,String isNotify,
IFriendsActionListener listener)

添加好友。

参数:

buddyUid - 好友的 uid.

isNotify - 是否通知好友

listener - 增加好友成功，回调 IActionListener.onSuccess(); 否则，回调 IFriendsActionListener#onFailure(ErrorInfo))

抛出:

IllegalArgumentException - 如果参数为 null。

removeBuddy

removeBuddy

void removeBuddy(String buddyUid,String isNotify,
IFriendsActionListener listener)

删除好友。

参数:

buddyUid - 好友的 uid.

isNotify - 是否通知好友

listener - 删除好友成功，回调 IActionListener.onSuccess(); 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

getBuddies

getBuddies

void getBuddies(IFriendsActionListener listener)

获取好友列表.

参数:

listener - 获取好友列表成功，回调 IFriendsActionListener.onResultData(Object) ，参数 Object 类型是 List<UserInfo>，表示好友信息列表； 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

createGroup

createGroup

void createGroup(String groupname,
List<String> userIdList,
IFriendsActionListener listener)

创建群.

参数:

groupName - 组群名称.

userIdList - 拉进群的用户 id.

listener - 创建群成功，回调 IFriendsActionListener.onResultData(Object) ，参数 Object 类型是 GroupInfo，表示群信息； 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

getGroups

getGroups

void getGroups(IFriendsActionListener listener)

获取群列表.

参数:

listener - 获取群列表成功,回调 IFriendsActionListener.onResultData(Object) ，参数 Object 类型是 List<GroupInfo> ，表示该用户的所有群信息列表； 否则，回调 IActionListener.onFailure(ErrorInfo)}

抛出:

IllegalArgumentException - 如果参数为 null。

exitGroup

exitGroup

void exitGroup(String groupId,IActionListener listener)

主动退出该群.

参数:

groupId - 群 Id。

listener - 操作是否成功。

抛出:

IllegalArgumentException - 如果参数为 null。

destoryGroup

destoryGroup

```
void destoryGroup(String grougId,
                  IFriendsActionListener listener)
```

注销群.

参数:

grougId - 群 Id.

listener - 删除群成功，回调 IActionListener.onSuccess(); 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

addGroupMembers

addGroupMembers

```
void addGroupMembers(String grougId,
                     List<String> userlists,
                     IFriendsActionListener listener)
```

添加群成员.

参数:

grougId - 群 Id，在系统里群 Id 唯一.

userList - 进群的用户 id 列表.

listener - 添加群成员成功，回调 IActionListener.onSuccess(); 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

removeGroupMembers

removeGroupMembers

```
void removeGroupMember(String grougId,
                       List<String> userlists,
                       IFriendsActionListener listener)
```

删除群成员.

参数:

grougId - 群 Id，在系统里群名称唯一.

userList - 该群中要删除的用户 id 列表.

listener - 删除群成员成功，回调 IActionListener.onSuccess(); 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

getGroupMembers

getGroupMembers

```
void getGroupMembers(String grougId,
                     IFriendsActionListener listener)
```

获取群成员列表.

参数:

grougId - 群 Id.

listener - 获取群成员成功，回调 IFriendsActionListener.onResultData(Object) ，参数类型是 List<UserInfo>，表示所有群成员信息列表； 否则，回调 IActionListener.onFailure(ErrorInfo)

抛出:

IllegalArgumentException - 如果参数为 null。

getSdkVersion

getSdkVersion

String getSdkVersion()

获取 A-SDK 的版本信息。

getIlinkTime

getIlinkTime

void getIlinkTime(TimeListener listener)

获取 Ilink 系统时间。

参数:

listener - 查询时间回调

抛出:

IllegalArgumentException - 如果参数为 null

setCallback

setCallback

void setCallback(ISDKContextCallback callback)

设置 A-SDK 回调函数。

参数:

callback - 回调函数。

抛出:

IllegalArgumentException - 如果参数为 null

ISDKContextCallback

public interface ISDKContextCallback
通知 A-SDK 的状态变化，接收到新消息，新文件，搜索到新服务器，与服务器丢失连接等。

方法详细说明:

onMessage

onMessage

void onMessage(MessageType type, String topic, Message msg)

接收到新消息，根据 type 不同，topic 的内容不同。

当 type 是 MessageType.CHAT_TO, topic 表示发送者的 uid。

当 type 是 MessageType.PUBLISH, topic 表示消息的主题。

参数:

type - 消息类型。

topic - 主题。

msg - 消息。

onFileReceived

onFileReceived

void onFileReceived(String senderId, DocumentInfo documentInfo)

有新的文件需要接收。

使用 ISDKContext.downloadFile(DocumentInfo, String,
com.beidouapp.et.client.callback.FileCallBack) 下载文件。

参数:

senderId - 发送者的 uid。

documentInfo - 文件信息。

onServer

onServer

void onServer(Server svr)

发现到新的服务器。

参数:

svr - 新加入到网络中的服务器。

onPeerState

onPeerState

void onPeerState(String uid, String statusCode)

关注的用户上下线状态变化通知。

参数:

uid - 关注的用户 uid。

statusCode - 值域[0:离线, 1:在线]。

onBroken

onBroken

void onBroken(Server svr, int errorCode, String reason)

与服务器断开连接。

参数:

svr - 已经断开的服务器

errorCode - 断开连接的原因代码

reason - 断开连接的原因解释

IActionListener

public interface IActionListener
Context 的动作回调

方法详细说明:

onSuccess

onSuccess

void onSuccess()

操作成功

onFailure

onFailure

void onFailure(ErrorInfo errorInfo)

操作失败

IFriendsActionListener

public interface IFriendsActionListener
extends IActionListener

好友群组管理回调。

方法详细说明：

onSuccess

onSuccess
void onSuccess() 操作成功

onFailure

onFailure
void onFailure(ErrorInfo errorInfo) 操作失败

onResultData

onResultData
void onResultData(Object data) 操作结果的数据 **参数:** data - 具体的数据类型详见好友操作方法描述。

Server

public class Server extends Object
服务器信息。

Server

public Server(Server.Type type,
String id,
String ip,
int port)

参数:

type - 服务器的类型，Type.lan, Type.wan

id - 服务器 id

ip - 服务器的 ip

port - 服务器的端口

ErrorInfo

public class ErrorInfo extends Object
SDK 方法调用的错误信息。
通常在 IActionListener.onFailure(ErrorInfo)中返回。

getCode

getCode

public int getCode()

获取错误码。

返回:

调用 sdk 的某个方法，发生异常，或者请求失败时的错误码。

另请参阅:

ErrorCode

getReason

getReason

public String getReason()

获取错误原因。

返回:

调用 sdk 的某个方法，发生异常，或者请求失败时的原因；如果原因不详，为 null。

重要申明

注意

本手册描述的产品及其附件的某些特性和功能，取决于当地网络的设计和性能，以及您安装的软件。某些特性和功能可能由于当地网络运营商或网络服务供应商不支持，或者由于当地网络的设置，或者您安装的软件不支持而无法实现。因此，本手册中的描述可能与您购买的产品或其附件并非完全一一对应。
易通星云（北京）科技发展有限公司保留随时修改本手册中任何信息的权利，无需进行任何提前通知且不承担任何责任。

无担保声明

本手册中的内容均“如是”提供，除非适用法要求，易通星云（北京）科技发展有限公司对本手册中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性或者适用于某一特定目的的保证。
在法律允许的范围内，易通星云（北京）科技发展有限公司在任何情况下，都不对因使用本手册相关内容而产生的任何特殊的、附带的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉或预期节约的损失进行赔偿。

联系方式

易通星云（北京）科技发展有限公司
公司地址：四川省成都市天府大道北段 1480 号德商国际 B703
邮编：610000
官网地址：www.kaifakuai.com
支持邮箱：support@beidouapp.com
服务电话：028-87582803
官网 QQ 群：445880047
微信公众号：开发快