Web 开发文档

ChangeLog

Version 1.0

1、用户下线

2、C2C 消息收发（文本、表情）

3、用户关系链管理（好友，黑名单管理）

4、用户资料管理（昵称、性别，加好友设置）

1. 概述

1.1. ImSDK 集成

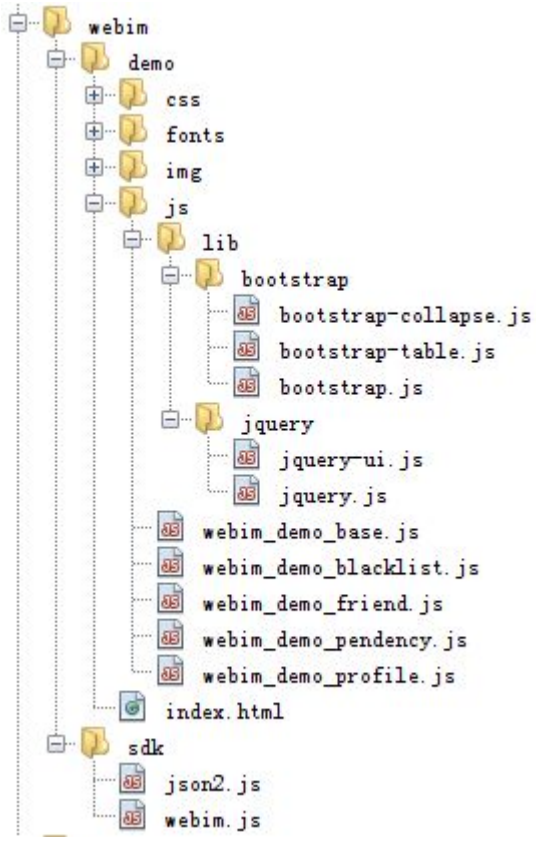
本节主要介绍如何集成 ImSDK。由于 webim sdk 1.0 还没有打通帐号系统（注册，登录），所有目前只能提供几个体验帐号，来模拟用户登录操作。目前可以测试的帐号有 peakerdong、qiyueliuhuo2016、qiyueliuhuo2017、qiyueliuhuo2018、qiyueliuhuo2019。

1.1.1. 下载 ImSDK

从官网下载 ImSDK：包含以下库文件
sdk/webim.js
sdk/json2.js
其中 json2.js 提供了 json 的序列化和反序列化方法，可以将一个 json 对象转换成 json 字符串，也可以将一个 json 字符串转换成一个 json 对象。webim.js 就是 webim sdk 库，提供了私聊，资料管理，关系链（好友，黑名单）管理功能。

1.1.2. 集成 ImSDK

将步骤 1.1.1 下载得到的库文件引入到前台页面中，这里拿 demo 举例，webim sdk 和 demo 目录结构如下，



demo 中的 index.html 加入以下两行代码：

```
<script type="text/javascript" src="../sdk/webim.js"></script>
<script type="text/javascript" src="../sdk/json2.js"></script>
```

1.1.3. 功能开发

在工程中引入上述 1.1.2 提及的库文件，根据后续章节的开发指引进行功能的开发。其中函数调用顺序可参见（1.2.2 调用顺序介绍）。

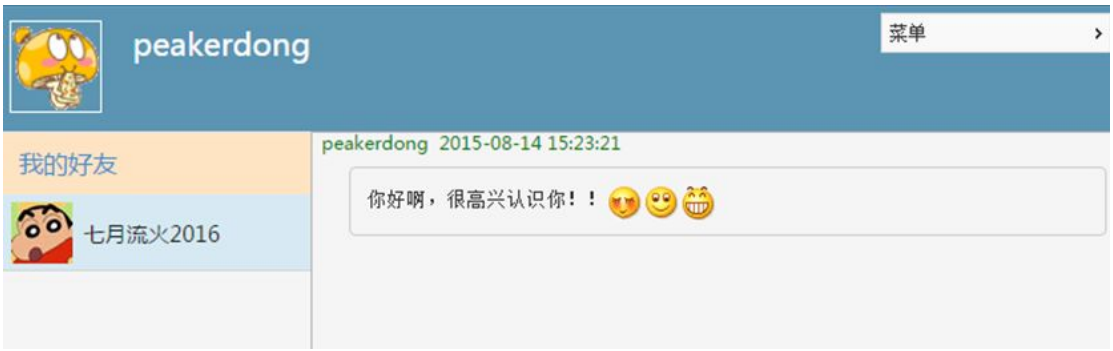
1.1.4. 支持版本

ImSDK 支持 IE 10+，Chrome 4+，FireFox 3.5+，Opera 12+和 Safari 4+。

1.2. ImSDK 基本概念

会话：ImSDK 中会话(Session)分为两种，一种是 C2C 会话，表示单聊情况自己与对方建立的对话；另一种是群会话，表示群聊情况下，群内成员组成的会话，后续版本会支持。

如下图所示，一个会话表示与一个好友的对话：



消息：ImSDK 中消息(webim.Msg)表示要发送给对方的内容，消息包括若干属性，如自己是否为发送者，发送人帐号，消息产生时间等；一条消息由若干 Elem 组合而成，每种 Elem 可以是文本、表情等等，消息支持多种 Elem 组合发送。

1.2.1. ImSDK 对象简介

ImSDK 对象主要分为消息，会话，消息存储对象，自带表情对象，工具对象，具体的含义参见下表：

对象	介绍	功能
webim.Msg	一条消息的描述类,	消息发送、接收的 API 中都会涉及此类型的对象
webim.Session	一个会话的描述类	包括获取会话类型，会话 ID，

		会话中的未读消息数，会话中的总消息数等功能
webim.MsgStore	消息数据的 Model 对象(参考 MVC 概念)	提供接口访问当前存储的会话和消息数据，包括获取当前会话的个数，根据会话类型和会话 ID 取得相应会话等功能
webim.EmotionPicData	表情数据	表情数据，格式是 base64 编码的
webim.EmotionPicData Index	表情数据索引	表情数据对应的索引
webim.Tool	工具对象	提供了一些公用的函数。比如格式化时间戳函数 formatTimeStamp(),获取字符串（utf-8 编码）所占字节数 getStrBytes()等等。

1.2.2. 调用顺序介绍

ImSDK 调用 API 需要遵循以下顺序。

步骤	对应函数	说明
初始化	webim.init	初始化 SDK，需要传入当前用户信息,新消息通知回调函数
消息收发	webim.syncMsgs	获取未读 c2c 消息
	webim.setAutoRead	设置会话自动已读标记
	webim.sendMsg	发送消息
资料管理	webim.getProfilePortrait webim.setProfilePortrait	获取/设置个人资料
关系链管理	webim.applyAddFriend webim.getAllFriend 等	申请添加好友，获取我的好友等
下线	webim.offline	用户下线

2. 初始化

2.1. 初始化 init

webim.init 表示初始化 web sdk,需要传入当前用户信息，新消息通知回调函数。

```
/* function init
```

```
        *    初始化 SDK
        * params:
        *    loginInfo          - Object, 登录身份相关参数集合，详见下面
        *    {
        *      sdkAppID       - String, 用户标识接入 SDK 的应用 ID
        *      appIDAt3rd     - String, App 用户使用 OAuth 授权体系分配的 Appid，
和 sdkAppID 一样
        *        accountType    - int, 账号类型
        *      identifier   - String, 用户帐号
        *      userSig          - String, 鉴权 Token
        *    }
        *    listeners       - Object, 事件回调函数集合, 详见下面
        *    {
        *      onConnNotify - function(connInfo), 用于收到连接状态相关通知的回调
函数,目前未使用
        *      onMsgNotify    - function(notifyInfo), 用于收到消息通知的回调函数,
notifyInfo 为[{msg: Msg 对象}]
        *                              使用方有两种处理回调: 1)直接访问 webim.MsgStore
获取最新的消息  2)处理 notifyInfo 中的增量消息
        *    }
        *    options        - Object, 其它选项, 目前未使用
        * return:
        *    (无)
        */
       init: function(loginInfo, listeners, options) {},
```

Demo 中使用例子：

```
//当前用户身份
        var loginInfo = {
            sdkAppID: 1104620500,
            appIDAt3rd: 1104620500,
            identifier: 'qiyueliuhuo2016',
            accountType: 107,
            userSig: 'fjsjfaf84rojfa9ufefdffe33',
            headurl: 'img/2016.gif'
        };

        //监听事件
        var listeners = {
            onConnNotify: null,
            onMsgNotify: onMsgNotify
        };
//初始化 demo
        function initDemoApp() {
```

```
                document.getElementById("webim_demo").style.display = "block";//展开聊天界面
                document.getElementById("p_my_face").src = loginInfo.headurl;
                document.getElementById("t_my_name").innerHTML = loginInfo.identifier;
                //菜单
                $("#t_my_menu").menu();

                //web sdk  初始化
                webim.init(loginInfo, listeners, null);
                //读取我的好友列表
                getAllFriend(getAllFriendsCallbackOK);

                //默认选中用于发送消息的文本框
                $("#send_msg_text").focus();

                //默认展开我的好友列表
                $('#collapseFriend').collapse('show');

            }


    //监听新消息事件
            function onMsgNotify(newMsg) {
                //获取所有聊天会话
                var sessMap = webim.MsgStore.sessMap();
                for (var i in sessMap) {
                    var sess = sessMap[i];
                    if (selToID == sess.id()) {//处于当前聊天界面
                        selSess = sess;
                        //获取当前会话消息数
                        var msgCount = sess.msgCount();
                        // add new msgs
                        if (msgCount > curMsgCount) {

                            for (var j = curMsgCount; j < msgCount; j++) {
                                var msg = sess.msg(j);
                                //在聊天窗体中新增一条消息
                                addMsg(msg);
                                curMsgCount++;
                            }
                            //消息已读上报，以及设置会话自动已读标记
                            webim.setAutoRead(selSess, true, true);
                        }
```

```
                } else {
                    //更新其他聊天对象的未读消息数
                    updateSessDiv(sess.id(), sess.unread());
                }

            }
        }
```

3. 登录

3.1. 登录

Im sdk 1.0 还不支持登录功能

3.2. 登出

如用户主动注销或需要进行用户的切换，则需要调用注销操作：

```
/* function offline
    *    用户下线
    * params:
    *    cbOk    - function()类型, 成功时回调函数
    *    cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
    *    (无)
    */
   offline: function(cbOk, cbErr) {},
```

示例：

```
//登出
webim.offline();
```

4. 消息收发

4.1. 获取未读 c2c 消息

```
/* function syncMsgs
    *    拉取最新 C2C 消息
```

```
            *     一般不需要使用方直接调用, SDK 底层会自动同步最新消息并通知使用方, 一种有用的调用场景是用户手动触发刷新消息
            * params:
            *     cbOk    - function(notifyInfo)类型, 当同步消息成功时的回调函数, notifyInfo 同上面 cbNotify 中的说明,
            *                   如果此参数为 null 或 undefined 则同步消息成功后会像自动同步那样回调 cbNotify
            *     cbErr    - function(err)类型, 当同步消息失败时的回调函数, err 为错误对象
            * return:
            *     (无)
            */
           syncMsgs: function(cbOk, cbErr) {},
```

示例:

```
webim.syncMsgs(syncMsgsCallbackOK);
//获取 C2C 最新消息成功回调函数
            function syncMsgsCallbackOK() {

                if (webim.MsgStore.sessCount() > 0) {

                    var sessMap = webim.MsgStore.sessMap();
                    for (var i in sessMap) {
                        //console.info("sessMap[i]=%O",sessMap[i]);
                        var sess = sessMap[i];
                        if (selToID == sess.id()) {//处于当前聊天界面
                            selSess = sess;

                            var msgCount = sess.msgCount();
                            // add new msgs
                            if (msgCount > curMsgCount) {

                                for (var mj = curMsgCount; mj < msgCount; mj++) {
                                    var msg = sess.msg(mj);
                                    addMsg(msg);
                                    curMsgCount++;
                                }
                                //消息已读上报，以及设置会话自动已读标记
                                webim.setAutoRead(selSess, true, true);
                            }
                        } else {
```

```
                    //更新其他聊天对象的未读消息数
                    updateSessDiv(sess.id(), sess.unread());
                }

            }
        }
    }
```

4.2. 消息发送

```
/* function sendMsg
     *    发送一条消息
     * params:
     *    msg      - webim.Msg 类型, 要发送的消息对象
     *    cbOk     - function()类型, 当发送消息成功时的回调函数
     *    cbErr    - function(err)类型, 当发送消息失败时的回调函数, err 为错误对象
     * return:
     *    (无)
     */
    sendMsg: function(msg, cbOk, cbErr) {},
```

示例：

```
//发送消息
    function onSendMsg() {
        if (!selToID) {
            alert("您还没有好友，暂不能聊天");
            $("#send_msg_text").val('');
            return;
        }
        //获取消息内容
        var msgtosend = document.getElementsByClassName("msgedit")[0].value;
        var msgLen = webim.Tool.getStrBytes(msgtosend);
        if (msgtosend.length < 1) {
            alert("发送的消息不能为空!");
            $("#send_msg_text").val('');
            return;
        }
        if (msgLen > 12000) {
            alert("发送的消息过长!");
            $("#send_msg_text").val('');
```

```
                return;
            }
            if (!selSess) {
                selSess = new webim.Session(selType, selToID, selToID,
friendHeadUrl, Math.round(new Date().getTime() / 1000));
            }
            var msg = new webim.Msg(selSess, true);

            //解析文本和表情
            var emessage = [];
            var expr = /\[[^[\]]{1,2}\]/mg;
            var emotions = msgtosend.match(expr);
            if (!emotions || emotions.length < 1) {
                var text_obj = new webim.Msg.Elem.Text(msgtosend);
                msg.addText(text_obj);
            } else {
                var isemotion = false;
                for (var i = 0; i < emotions.length; i++) {
                    var tmsg = msgtosend.substring(0,
msgtosend.indexOf(emotions[i]));
                    if (tmsg) {
                        var text_obj = new webim.Msg.Elem.Text(tmsg);
                        msg.addText(text_obj);
                    }
                    var emotion = webim.EmotionPicData[emotions[i]];
                    if (emotion) {
                        isemotion = true;
                        var face_obj = new
webim.Msg.Elem.Face(webim.EmotionPicDataIndex[emotions[i]], emotions[i]);
                        msg.addFace(face_obj);

                    } else {
                        var text_obj = new
webim.Msg.Elem.Text(emotions[i]);
                        msg.addText(text_obj);
                    }
                    var restMsgIndex = msgtosend.indexOf(emotions[i]) +
emotions[i].length;
                    msgtosend = msgtosend.substring(restMsgIndex);
                }
                if (msgtosend) {
                    var text_obj = new webim.Msg.Elem.Text(msgtosend);
                    msg.addText(text_obj);
                }
```

```
        }

        webim.sendMsg(msg, function (resp) {
            addMsg(msg);
            curMsgCount++;
            $("#send_msg_text").val('');
            turnoffFaces_box();
        }, function (err) {
            alert(err.ErrorInfo);
            $("#send_msg_text").val('');
        });
    }
```

4.3. 设置会话自动已读标记

4.4. 获取所有会话

Session 对象,简单理解为最近会话列表的一个条目

```
    /* class Session
        *       type   - String, 会话类型(如"C2C", "GROUP", ...)
        *       id     - String, 会话 ID(如 C2C 类型中为对方帐号,"C2C"时为好友ID,"GROUP"时为群 ID)
        * properties:
        *    (Session 对象未对外暴露任何属性字段, 所有访问通过下面的 getter 方法进行)
        * methods:
        *    type()         - String, 返回会话类型,"C2C"表示与好友私聊，"GROUP"表示群聊
        *    id()       - String, 返回会话 ID
        *    name()         - String, 返回会话标题(如 C2C 类型中为对方的昵称)
        *    icon()         - String, 返回会话图标(对 C2C 类型中为对方的头像 URL)
        *    unread()               - Integer, 返回会话未读条数
        *    time()         - Integer, 返回会话最后活跃时间, 为 unix timestamp
        *    curMaxMsgSeq()  - Integer, 返回会话最大消息序列号
        *    msgCount()   - Integer, 返回会话中所有消息条数
        *    msg(index)    - webim.Msg, 返回会话中第 index 条消息
        */
```

webim.MsgStore 是消息数据的 Model 对象,它提供接口访问当前存储的会话和消息数据。

```
MsgStore: {
        /* function sessMap
         *     获取所有会话
         * return:
         *     所有会话对象
         */
        sessMap: function() {return {/*Object*/};},
        /* function sessCount
         *     获取当前会话的个数
         * return:
         *     Integer, 会话个数
         */
        sessCount: function() {return 0;},
        /* function sessByTypeId
         *     根据会话类型和会话 ID 取得相应会话
         * params:
         *     type      - String, 会话类型(如"C2C", "GROUP", ...)
         *     id        - String, 会话 ID(如对方 ID)
         * return:
         *     Session, 会话对象(说明见上面)
         */
        sessByTypeId: function(type, id) {return {/*Session Object*/};},
        /* function delSessByTypeId
         *     根据会话类型和会话 ID 删除相应会话
         * params:
         *     type      - String, 会话类型(如"C2C", "GROUP", ...)
         *     id        - String, 会话 ID(如对方 ID)
         * return:
         *     Boolean, 布尔类型
         */
        delSessByTypeId: function(type, id) {return true;},

        /* function resetCookieAndSyncFlag
         *     重置上一次读取新 c2c 消息 Cookie 和是否继续拉取标记
         * return:
         *
         */
        resetCookieAndSyncFlag: function() {}
    }
```

示例：

```
//获取所有聊天会话
var sessMap = webim.MsgStore.sessMap();

//遍历会话列表
for (var i in sessMap) {
    var sess = sessMap[i];
    if (selToID == sess.id()) {//处于当前聊天界面
        selSess = sess;
        //获取当前会话消息数
        var msgCount = sess.msgCount();
        // add new msgs
        if (msgCount > curMsgCount) {

        for (var j = curMsgCount; j < msgCount; j++) {
            var msg = sess.msg(j);
            //在聊天窗体中新增一条消息
            addMsg(msg);
            curMsgCount++;
        }
        //消息已读上报，以及设置会话自动已读标记
        webim.setAutoRead(selSess, true, true);
        }
    } else {
        //更新其他聊天对象的未读消息数
        updateSessDiv(sess.id(), sess.unread());
    }
}
```

4.5. 获取会话

可以根据会话类型和会话 ID 取得相应会话。

```
/* function sessByTypeId
        *    根据会话类型和会话 ID 取得相应会话
        * params:
        *    type      - String, 会话类型(如"C2C", "GROUP", ...)
        *    id        - String, 会话 ID(如对方 ID)
        * return:
        *    Session, 会话对象(说明见上面)
        */
        sessByTypeId: function(type, id) {return {/*Session Object*/};},
```

示例：

```
selSess = webim.MsgStore.sessByTypeId(selType, selToID);
```

5. 未读计数

5.1. 获取当前会话未读消息数

可以根据 Session 对象定义的 unread()方法获取未读消息数

示例：

```
//更新其他聊天对象的未读消息数
updateSessDiv(sess.id(), sess.unread());
```

5.2. 设置会话自动已读标记

当用户阅读某个会话的数据后，需要进行会话消息的已读上报，SDK 根据会话中最后一条阅读的消息，设置会话中之前所有消息为已读。

```
/* function setAutoRead
                *     //设置会设置会话自动已读标记话自动已读上报标志
                * params:
                *     selSess   - webim.Session 类型, 当前会话
                *     isOn      - boolean, 将 selSess 的自动已读消息标志改为 isOn，同时是否上
    报当前会话已读消息
                *     isResetAll     - boolean，是否重置所有会话的自动已读标志
                * return:
                *     (无)
                */
               setAutoRead: function(selSess, isOn, isResetAll) {},
```

示例：

```
//消息已读上报，以及设置会话自动已读标记
webim.setAutoRead(selSess, true, true);
```

5.3. 重置上一次读取新 c2c 消息 Cookie 和是否继续拉取标记

当切换聊天对象时，需要调用 resetCookieAndSyncFlag 方法。

```
/* function resetCookieAndSyncFlag
                *     重置上一次读取新 c2c 消息 Cookie 和是否继续拉取标记
                * return:
                *
                */
```

```
        resetCookieAndSyncFlag: function() {}
```

示例：

```
webim.MsgStore.resetCookieAndSyncFlag();
```

6. 用户资料

IM 云提供的帐号分为两种，一种是独立帐号体系，由用户自己保存并更新用户资料和关系链信息，另一种是托管模式，此模式下用户可完全不用搭建自己的后台服务，把用户的资料和关系链托管在 IM 云服务。

如果开发者选择资料和关系链托管，需要调用本章的接口进行资料和关系链的操作：

6.1. 设置个人资料

```
/* function setProfilePortrait
     *    设置个人资料
     * params:
     *    cbOk    - function()类型, 成功时回调函数
     *    cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
     * return:
     *    (无)
     */
    setProfilePortrait: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-资料管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_profile.js 的示例代码

6.2. 获取个人资料

```
/* function getProfilePortrait
     *    拉取资料（搜索用户）
     * params:
     *    cbOk    - function()类型, 成功时回调函数
     *    cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
     * return:
     *    (无)
     */
    getProfilePortrait: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-资料管理-REST 协议文档》

示例：
请参考 demo 中的 demo/js/webim_demo_profile.js 的示例代码。

7. 关系链

7.1. 申请增加好友

```
/* function applyAddFriend
    *     申请添加好友
    * params:
    *     cbOk     - function()类型, 成功时回调函数
    *     cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
    *     (无)
    */
   applyAddFriend: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》
示例：
请参考 demo 中的 demo/js/webim_demo_friend.js 的示例代码。

7.2. 拉取好友申请

```
/* function getPendency
    *     拉取好友申请
    * params:
    *     cbOk     - function()类型, 成功时回调函数
    *     cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
    *     (无)
    */
   getPendency: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》
示例：
请参考 demo 中的 demo/js/webim_demo_pendency.js 的示例代码。

7.3. 响应好友申请

```
/* function responseFriend
    *     响应好友申请
    * params:
    *     cbOk     - function()类型, 成功时回调函数
    *     cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
```

```
*     (无)
*/
responseFriend: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_pendency.js 的示例代码。

7.4. 删除好友申请

```
/* function deletePendency
*     删除好友申请
* params:
*     cbOk     - function()类型, 成功时回调函数
*     cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
* return:
*     (无)
*/
deletePendency: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_pendency.js 的示例代码。

7.5. 我的好友列表

```
/* function getAllFriend
*     拉取我的好友
* params:
*     cbOk     - function()类型, 成功时回调函数
*     cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
* return:
*     (无)
*/
getAllFriend: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_friend.js 的示例代码。

7.6. 删除好友

```
* function deleteFriend
*     删除好友
* params:
*     cbOk     - function()类型, 成功时回调函数
*     cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
* return:
*     (无)
```

```
    */
    deleteFriend: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_friend.js 的示例代码。

7.7. 增加黑名单

```
* function deleteFriend
    *    删除好友
    * params:
    *    cbOk    - function()类型, 成功时回调函数
    *    cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
    *    (无)
    */
    deleteFriend: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_blacklist.js 的示例代码。

7.8. 我的黑名单

```
/* function getBlackList
    *    删除黑名单
    * params:
    *    cbOk    - function()类型, 成功时回调函数
    *    cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
    *    (无)
    */
    getBlackList: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_blacklist.js 的示例代码。

7.9. 删除黑名单

```
/* function deleteBlackList
    *    我的黑名单
    * params:
    *    cbOk    - function()类型, 成功时回调函数
    *    cbErr    - function(err)类型, 失败时回调函数, err 为错误对象
    * return:
    *    (无)
    */
    deleteBlackList: function(options,cbOk, cbErr) {},
```

具体参数请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》

示例：

请参考 demo 中的 demo/js/webim_demo_blacklist.js 的示例代码。

8. 错误码说明

8.1. 收发消息错误码

命令名	错误码	错误描述
syncMsgs	20001	无效包
	20002	签名鉴权失败
	20003	无效的号码
	20004	网络异常
	20005	服务异常
	20006	因第三方要求，拦截发送方请求
	80001	安全打击
sendMsg	20001	无效包
	20002	签名鉴权失败
	20003	无效的号码
	20004	网络异常
	20005	服务异常
	20006	因第三方要求，拦截发送方请求
	80001	安全打击

8.2. 资料相关错误码

与资料相关的 api 错误码请参考《腾讯云 IM 开放-WebSDK-资料管理-REST 协议文档》

8.3. 关系链相关错误码

与关系链相关的 api 错误码请参考《腾讯云 IM 开放-WebSDK-关系链管理-REST 协议文档》