目录

[ 概述： 4](#_Toc466367035)

[ JS SDK使用步骤： 4](#_Toc466367036)

[步骤一：绑定域名 4](#_Toc466367037)

[步骤二：引入JS文件 5](#_Toc466367038)

[步骤三：通过deviceReady接口监听客户端插件加载完毕 5](#_Toc466367039)

[步骤四：通过config接口注入权限验证配置 6](#_Toc466367040)

[步骤五：通过ready和error接口处理成功和失败验证 6](#_Toc466367041)

[ 接口调用说明 7](#_Toc466367042)

[1 回调函数说明 7](#_Toc466367043)

[2 params参数传递说明 9](#_Toc466367044)

[3 接口调用注意事项 10](#_Toc466367045)

[ Debug模式说明 10](#_Toc466367046)

[1 Debug模式介绍 10](#_Toc466367047)

[2 debugCallbackType参数说明 12](#_Toc466367048)

[3 Zip包形式的Web app使用Debug模式 13](#_Toc466367049)

[ 监听客户端接口 14](#_Toc466367050)

[1 监听客户端已经进入后台接口 14](#_Toc466367051)

[2 监听客户端即将从后台恢复接口 14](#_Toc466367052)

[3 监听安卓用户点击返回物理键事件接口 15](#_Toc466367053)

[4 监听安卓用户点击菜单物理键事件接口 15](#_Toc466367054)

[ 插件接口 15](#_Toc466367055)

[1 基础接口 15](#_Toc466367056)

[1.1 判断当前客户端是否支持指定JS接口 15](#_Toc466367057)

[1.2 开启调试模式接口 16](#_Toc466367058)

[1.3 关闭调试模式接口 17](#_Toc466367059)

[1.4 获取插件版本号接口 17](#_Toc466367060)

[2 网络请求接口 17](#_Toc466367061)

[2.1 配置网络请求接口 17](#_Toc466367062)

[2.2 通过司信客户端请求网络接口 19](#_Toc466367063)

[2.3 取消网络请求接口 20](#_Toc466367064)

[3 分享接口 20](#_Toc466367065)

[3.1 默认分享接口 20](#_Toc466367066)

[3.2 自定义内容分享接口 21](#_Toc466367067)

[3.3 开启自定义内容分享功能接口 22](#_Toc466367068)

[3.4 关闭自定义内容分享功能接口 23](#_Toc466367069)

[4 图像接口 24](#_Toc466367070)

[4.1 拍照或从手机相册中选图接口 24](#_Toc466367071)

[4.2 头像图片选择接口 25](#_Toc466367072)

[4.3 图片预览接口 26](#_Toc466367073)

[4.4 上传图片接口 27](#_Toc466367074)

[4.5 批量上传图片接口 28](#_Toc466367075)

[4.6 下载图片接口 28](#_Toc466367076)

[4.7 批量下载图片接口 29](#_Toc466367077)

[5 音频接口 30](#_Toc466367078)

[5.1 开始录音和监听录音自动停止接口 30](#_Toc466367079)

[5.2 停止录音接口 31](#_Toc466367080)

[5.3 播放语音和监听语音自动播放完毕接口 32](#_Toc466367081)

[5.4 暂停播放接口 32](#_Toc466367082)

[5.5 停止播放接口 33](#_Toc466367083)

[5.6 上传语音接口 33](#_Toc466367084)

[5.7 下载语音接口 34](#_Toc466367085)

[6 地理位置 35](#_Toc466367086)

[6.1 获取地理位置接口 35](#_Toc466367087)

[6.2 使用司信内置地图查看位置接口 36](#_Toc466367088)

[7 摇一摇 36](#_Toc466367089)

[7.1 开启和监听摇一摇事件接口 36](#_Toc466367090)

[7.2 关闭摇一摇功能接口 37](#_Toc466367091)

[8 二维码接口 37](#_Toc466367092)

[8.1 调起司信扫一扫接口 37](#_Toc466367093)

[8.2 生成二维码接口 38](#_Toc466367094)

[9 界面操作 39](#_Toc466367095)

[9.1 改变导航条标题接口 39](#_Toc466367096)

[9.2 隐藏右上角菜单按钮接口 40](#_Toc466367097)

[9.3 显示右上角菜单按钮接口 40](#_Toc466367098)

[9.4 批量隐藏功能按钮接口 40](#_Toc466367099)

[9.5 批量显示功能按钮接口 41](#_Toc466367100)

[9.6 显示所有功能按钮接口 41](#_Toc466367101)

[9.7 关闭当前网页窗口接口 42](#_Toc466367102)

[9.8 通过司信客户端alert信息接口 42](#_Toc466367103)

[9.9 通过司信客户端显示提示信息接口 43](#_Toc466367104)

[9.10 通过司信客户端显示loading接口 43](#_Toc466367105)

[9.11 通过司信客户端隐藏loading接口 44](#_Toc466367106)

[10 进入机构号 44](#_Toc466367107)

[10.1 判断用户是否关注机构号接口 44](#_Toc466367108)

[10.2 查看机构号接口 45](#_Toc466367109)

[10.3 给机构号发信息接口 45](#_Toc466367110)

[11 获取用户及公司信息 46](#_Toc466367111)

[11.1 获取司信当前用户接口 46](#_Toc466367112)

[11.2 获取司信当前用户常用联系人接口 47](#_Toc466367113)

[11.3 获取员工信息接口 48](#_Toc466367114)

[11.4 获取司信通讯录接口 49](#_Toc466367115)

[11.5 选取公司组织接口 51](#_Toc466367116)

[11.6 获取组织信息接口 51](#_Toc466367117)

[11.7 查看员工信息接口 52](#_Toc466367118)

[11.8 给员工发送消息接口 52](#_Toc466367119)

[12 设备信息 53](#_Toc466367120)

[12.1 获取网络状态接口 53](#_Toc466367121)

[12.2 获取设备系统类型接口 53](#_Toc466367122)

[12.3 获取设备系统版本号接口 54](#_Toc466367123)

[12.4 获取设备机型接口 55](#_Toc466367124)

[12.5 获取设备唯一标识接口 55](#_Toc466367125)

[12.6 获取设备信息接口 56](#_Toc466367126)

[13 数据缓存接口 56](#_Toc466367127)

[13.1 使用司信客户端缓存数据接口 56](#_Toc466367128)

[13.2 使用司信客户端读取缓存接口 57](#_Toc466367129)

[13.3 移除司信客户端缓存数据接口 58](#_Toc466367130)

[13.4 移除当前应用在司信客户端所有缓存数据接口 59](#_Toc466367131)

[14 支付接口 59](#_Toc466367132)

[14.1 使用司信客户端调用微信支付宝支付接口 59](#_Toc466367133)

[15 其他附加接口 61](#_Toc466367134)

[15.1 获取司信客户端版本号接口 61](#_Toc466367135)

[ 附录 61](#_Toc466367136)

[**附录1：JS-SDK使用权限签名算法** 61](#_Toc466367137)

[**签名算法** 62](#_Toc466367138)

[**注意事项** 62](#_Toc466367139)

[**附录2：网络请求签名算法** 63](#_Toc466367140)

[**签名算法** 63](#_Toc466367141)

[**签名示例** 64](#_Toc466367142)

[**附录3：结果代码说明** 64](#_Toc466367143)

[**附录4：接口列表** 68](#_Toc466367144)

[**附录5：菜单项ID列表** 71](#_Toc466367145)

[**附录6：Demo页面和示例代码** 72](#_Toc466367146)

[**Demo页面：** 72](#_Toc466367147)

[**示例代码：** 72](#_Toc466367148)

* **概述：**

司信JS-SDK是司信公众机构号平台面向网页开发者提供的基于司信内的网页开发工具包。SDK通过Plugin插件的方式实现了客户端Native与Web网页之间的通信和交互。

通过使用司信JS-SDK，网页开发者可借助司信插件高效地使用拍照、选图、语音、位置等手机系统的功能，同时可以直接使用司信自定义分享、摇一摇、扫一扫、获取用户信息等特有的插件，为司信用户提供更优质的网页体验。

此文档面向网页开发者介绍司信JS-SDK接口如何使用及相关注意事项。

* **JS SDK使用步骤：**

## 步骤一：绑定域名

登录司信机构号运营平台进入“应用管理”填写JS 接口安全域名。绑定了父级域名，其合法的子域名也可以使用。

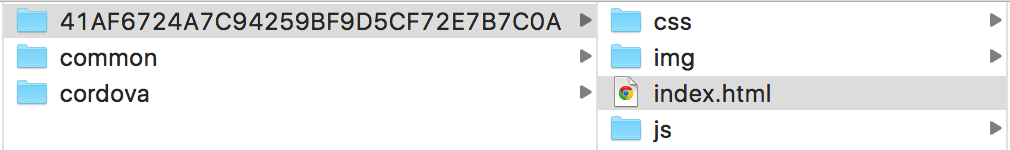
域名填写支持一级或一级以上域名，如isoftstone.com，或者www.isoftstone.com，当填写为一级域名时其子域名的url都能访问，当填写为子域名时则只有当前子域名的url可以访问；支持IP地址，如192.168.0.1，不支持端口号和短链域名。

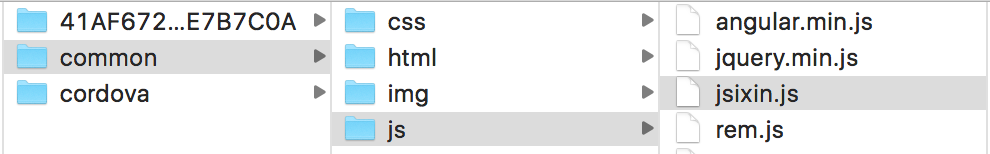
（zip 包形式的Web app，无需该步骤）。

## 步骤二：引入JS文件

在需要调用JS接口的页面引入[jsixin.js](http://jssdk.issmobile.com/JS-SDK/jsixin/jsixin.js)文件。开发者可以直接引用链接地址，也可下载文件加入到自己的工程。

zip包形式的Web app，为了提高运行速度和用户体验，我们要求开发者使用客户端内置JS SDK，开发者引用客户端沙盒路径下的jsixin.js。Web app同客户端内置JS SDK依赖库的目录结构如下：





其中以41A开头命名的目录为Web app目录，如index.html需要引用jsixin.js，则引用相对路径为“../common/js/jsixin.js”，开发者必须按以上目录结构引用jsixin.js，否则将导致JS SDK无法使用。需要注意的是，Web app的首页html页面需要命名为index.htm，当找不到该html页面时Web应用将无法正常加载。

## 步骤三：通过deviceReady接口监听客户端插件加载完毕

JS SDK所有接口都必须等待客户端插件加载完毕后才能调用，对于需要在页面加载时就调用的相关接口，则须把该接口放在deviceReady函数中调用来确保正确执行。对于用户触发时才调用的接口，则可以直接调用，不需要放在deviceReady函数中，因为此时插件已经加载完毕。（注意：为了安全起见，deviceReady最好在JS加载时调用，不要尝试在函数中调用，deviceReady方法类似window.onload等方法，只在客户端插件加载完后调用一次，在window.onload后执行。）

sx.deviceReady(**function** () {  
});

## 步骤四：通过config接口注入权限验证配置

所有需要使用JS-SDK的页面都必须先获取插件接口访问权限，否则将无法调用，相同域名仅需调用一次，当域名发生变化时需要重新调用获取权限。

config接口一般放在deviceReady函数中调用。

***sx***.config({  
 **debug**: **false**, *// 是否打开调试模式，默认false不打开，true开启调试模式调用所有api的返回值会在客户端alert出来* **appId**: **''**, *// 应用唯一标识（必传）* **timestamp**: 1414587457, *// 用于生成签名的时间戳（必传）* **nonceStr**: **'Wm3WZYTPz0wzccnW'**, *// 用于生成签名的随机字符串（必传）* **signature**: **''** *// 签名（必传）具体签名算法见****附录1***});

（zip 包形式的Web app，无需该步骤）。

## 步骤五：通过ready和error接口处理成功和失败验证

当调用config接口注入权限验证成功时会回调ready接口函数：

***sx***.ready(**function** (res) {  
 *// config信息验证后会执行ready方法，所有接口调用都必须在config接口获得结果之后，如果需要在页面加载时就调用相关接口，则须把相关接口放在ready函数中调用来确保正确执行。对于用户触发时才调用的接口，则可以直接调用，不需要放在ready函数中。*});

（zip 包形式的Web app，无需该步骤）。

失败时会回调error接口函数：

***sx***.error(**function** (err) {  
 *// config信息验证失败会执行error函数，如签名过期导致验证失败，具体错误信息可以打开config的debug模式查看，也可以在返回的err参数中查看，同时参数传递错误也会导致验证不通过。*});

（zip 包形式的Web app，无需该步骤）。

* **接口调用说明**

1. 回调函数说明

所有接口通过sx对象来调用，参数是一个对象，除了每个接口本身需要传的参数params外，还有以下通用参数：

* + - success：接口调用成功时执行的回调函数。
    - fail：接口调用失败时执行的回调函数。
    - enable：开启某个功能或监听某事件成功执行的回调函数。（目前SDK1.0.0版本接口3.3，5.1，5.3，7.1带有该参数）
    - trigger：开启某个功能或监听某事件后触发了该事件执行的回调函数。（目前SDK1.0.0版本接口3.1，3.2，3.3，7.1带有该参数）
    - cancel：用户取消操作后的回调函数。（目前SDK1.0.0版本接口2.2，3.1，3.2，3.3，4.1，4.2，4.4，4.5，4.6，4.7，5.6，5.7，8.1，9.8，9.10，11.4，11.5带有该参数）
    - refuse：访问地理位置，麦克风等用户隐私服务时被用户拒绝访问后的回调函数。（目前SDK1.0.0版本接口5.1，6.1带有该参数）
    - complete：接口调用完成时执行的回调函数。带有enable和trigger参数的接口不包含complete参数，其他接口回调success，fail，cancel，refuse都会执行complete回调。
    - debugCallbackType：该参数适用于开发者在调试模式下使用，传递相关参数值后，JS SDK会回调对应的函数，方便开发者调试。具体说明见“Debug模式说明”。

开发者可视自己的开发需要传递对应的回调函数，以上前7个函数都带有一个参数，类型为一个对象，开发者同样也可以视实际需要决定是否接收该参数。

该对象参数值和格式的说明如下：

调用接口成功时，如有数据返回则返回数据，否则返回通用属性resCode和resMsg，格式如下：

{**resCode**: **'0'**, **resMsg**: **'OK'**}

fail时，会返回具体错误代码和具体错误信息：

{**resCode**: **'xx'**, **resMsg**: **'xxx'**}

enable时，会返回01代码和具体enable信息：

{**resCode**: **'01'**, **resMsg**: **'xxx'**}

trigger时，会返回02代码和具体trigger信息：

{**resCode**: **'02'**, **resMsg**: **'xxx'**}

cancel时，会返回03代码和具体cancel信息：

{**resCode**: **'03'**, **resMsg**: **'xxx'**}

refuse时，会返回04代码和具体refuse信息：

{**resCode**: **'04'**, **resMsg**: **'xxx'**}

关于resCode值的具体含义请见附录3。

1. params参数传递说明
2. 参数包含必须和非必须，凡是标注为“必传”的参数必须传递，否则将调用接口失败。非必须参数可以不传，当不传时有默认值则自动采用默认值。
3. 所有参数都必须按规定格式传递，否则将影响到接口的使用。当参数为必须参数时如格式有误则会调用接口失败，非必须参数格式有误时，如有默认值则自动转为使用默认值，无默认值则全部置空处理。客户端的兼容处理能保障该接口的正常调用，但是任何一种参数错误类型都无法让客户端执行开发者想要真正执行的操作，所以建议开发者务必按照文档要求来传递参数。
   * + 对于bool类型的参数：比如是否获取地址isGetAddress，是否显示下载进度条isShowProgressTips等，传递“ture”或者“false”,也接受“0”和“1”number类型参数，但是不建议使用字符串“0”和“1”；其他任何类型的参数都将不接受而自动使用默认值。
     + 对于type类型的参数：比如图片的质量类型sizeType，坐标的类型coordinateType，为规定范围内的从0开始的连续自然数number类型（不建议使用字符串数字类型），任何其他类型的参数都将不接受而自动使用默认值。
     + 对于数值类型的参数：比如图片的数量count，坐标经纬度longitude和latitude，为规定范围内的连续自然数，浮点数等number类型（不建议使用字符串数字类型），任何其他类型的参数都将不接受而自动采用默认值或者返回错误信息。
     + 对于ID类型参数：如按钮ID menuId，用户ID userId等，必须为字符串格式，否则将调用接口失败。
     + 对于必须传递的字符串参数：必须为字符串格式，且不能传空或者传递空字符串，否则将调用接口失败。
     + 对于必须传递的数组参数：必须为数组格式，且不能传空，数组元素至少包含一个，否则将调用接口失败。
4. 接口调用注意事项

JS SDK各接口调用都是异步操作，当开发者需要在同一个方法中调用多个接口，比如先后调用接口A和B，如果B接口需要用到A接口的数据或者必须在A接口操作生效时才能调用，那么为了保证B接口的正常调用，需要把B接口放在A接口的success函数中执行。

* **Debug模式说明**

1. Debug模式介绍

司信JS-SDK所有接口的调用都对应依赖于客户端的插件去实现，在脱离客户端的情况下，调用SDK的方法，是无法去执行相关操作和获取客户端数据的。为了提高开发者的开发效率，尽量减少开发者在开发初期就花大量时间在客户端上来调试自己的Web app，司信JS-SDK为开发者提供了debug开发模式。

（注意：此处debug模式与config接口中的debug模式以及后面onDebug和closeDebug中的不同，此处为JS-SDK的开发者调试模式，而调用config等接口开启的Debug模式作用于客户端，开启后客户端在给Web返回数据时会把返回的信息alert出来，方便开发者查看返回数据）。

司信JS-SDK可自动识别Web app的运行平台，当Web app在司信客户端上运行时，正常通过客户端插件去处理相应事情和获取数据；当Web app运行在PC浏览器上时，Web app已脱离了司信客户端，JS-SDK自动开启debug模式，开发者调用SDK接口时，SDK自动返回该接口的模拟数据。

在debug模式下，SDK同样采用了同客户端一样的参数检验体系，开发者务必按文档要求来传递参数，否则将返回对应错误信息；debug模式返回的模拟数据，尽大可能的做到接近真实数据的使用场景，根据传过来的不同参数返回对应需要的数据，数据格式也严格遵守文档要求。

司信JS-SDK debug模式通过jsixin-debug-x.x.x.js实现，该js存储在网络服务器上，jsixin.js通过判断平台去决定是否引用该js，所以开发者不需要担心自己的app上线后会额外引用jsixin-debug-x.x.x.js。

当SDK 对应接口的debug数据没及时更新时，会在fail回调里返回401错误信息：

{**resCode**: **'401'**, **resMsg**: **'No debug data, please test you web app in sixin'**}

需要注意的是，与实际在客户端调用SDK方法不同，在debug模式下，所有接口调用都变成同步操作；

config接口在dubug模式下不会进行权限验证操作，默认权限直接开启，开发者调用任何接口都可拿到测试数据。

司信JS-SDK debug模式旨在为开发者提供更高效的开发环境，方便开发者在不使用客户端的情况下可以对自己开发的Web app进行调试。但是debug模式毕竟提供的是模拟数据，类似分享、选取图片等需要客户端去做的操作，debug模式是无法真实模拟的，开发者不能过渡依赖，不应该只在debug模式下调试好app后就上架应用，这样无法保证上架的app在司信客户端能够运行良好，所以我们要求开发者在开发初期时可以使用我们的debug模式进行调试，开发完成后应当在司信客户端上进行测试且稳定后再上架app。

1. debugCallbackType参数说明

司信JS-SDK在debug模式下默认回调success函数，或者少数接口回调enable函数，debug模式无法模拟多种情况去回调cancel，fail等其他函数，为了解决这一问题，SDK引用了debugCallbackType参数，如代码示例所示：

**sx**.shareCustomContent({  
 **params**: params,  
 **debugCallbackType**:**'trigger'**, *//需要指定回调的函数，参数值可以传'success','enable','trigger','cancel','refuse','fail'，该参数只在Debug模式下生效* trigger: **function** (res) {  
 *//Debug模式下会直接回调该函数* },  
 success: **function** (res) {  
 },  
 cancel: **function** (err) {  
 },  
 fail: **function** (err) {  
 }  
});

该参数只在Debug模式下生效，SDK会根据传递的不同参数而回调相应的函数，需要注意的是，Debug模式下一旦传递了debugCallbackType，那么回调于哪个函数只取决于debugCallbackType，且返回的结果信息res和err的resMsg值为固定消息，不会显示具体消息，如下所示：

{**resCode**: **'0'**, **resMsg**: **'OK'**}

{**resCode**: **'01'**, **resMsg**: **'enable'**}

{**resCode**: **'02'**, **resMsg**: **'trigger'**}

{**resCode**: **'03'**, **resMsg**: **'cancel'**}

{**resCode**: **'04'**, **resMsg**: **'refuse'**}

{**resCode**: **'400'**, **resMsg**: **'fail'**} *//此处fail的resCode为固定值'400'*

debugCallbackType意在帮助开发者测试失败，取消等场景下的事件处理，提高开发者在Debug模式下可调试的最大化。

1. Zip包形式的Web app使用Debug模式

前文中我们要求zip包形式的Web app引用客户端内置的JS SDK，包含JS SDK在内的公共Web文件我们都打包在common目录存储在客户端本地，下载到客户端沙盒的web app从客户端common目录下引用公共文件。开发者在开发时可以在SVN下载common公共包加入到自己的工程当中，要求common目录同web app的目录同级，这样才能真实模拟web app在客户端本地引用公共文件的运行场景。加入common目录，引入jsixin.js后，开发者就可以正常使用Debug模式了。开发者开发完毕准备上架应用时只需要把web app的目录打包上架就行，common目录不需要打包进去。

* **监听客户端接口**

1. 监听客户端已经进入后台接口

客户端进入后台后会回调该接口函数，开发者可以在客户端进入后台后进行数据保存等相关操作。

sx.appDidEnterBackground(**function** () {});

1. 监听客户端即将从后台恢复接口

客户端将要从后台恢复时会回调该接口函数，开发者可以在客户端即将从后台恢复时做相关操作。

sx.appWillEnterForeground(**function** () {  
});

1. 监听安卓用户点击返回物理键事件接口

当安卓用户点击返回物理键时回调该接口函数。

sx.clickOnBackButton(**function** () {});

1. 监听安卓用户点击菜单物理键事件接口

当安卓用户点击返回物理键时回调该接口函数。

sx.clickOnMenuButton(**function** () {});

* **插件接口**

1. 基础接口
   1. 判断当前客户端是否支持指定JS接口

描叙：开发者可以通过该接口判断客户端是否可以调用指定的JS接口。

代码示例：

**var *apiList*** = [  
 **'Share|onMenuShareCustomContent'**,  
 **'Audio|playVoice'**,  
 **'Audio|uploadVoice'**,  
 **'UserInfo|getUserInfo'**,  
 **'Device|getDeviceNetworkType'**];  
**var *params*** = {  
 **apiList**: ***apiList*** *// 需要验证的aip数组（必传）具体接口详细列表见附录4*};  
  
***sx***.checkPluginApi({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// 通过接口名去取* **var** checkResult = res.playVoice;  
 },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **onMenuShareCustomContent**: **true**,  
 **playVoice**: **true**,  
 **uploadVoice**: **true**,  
 **getUserInfo**: **true**,  
 **getDeviceNetworkType**: **true**}

Debug模式说明：由于无法检测客户端插件接口是否存在，Debug模式下的返回数据，换成为检测当前SDK对应接口是否存在，所以开发者并不能把当前结果作为判断客户端接口存在的依据。

* 1. 开启调试模式接口

描叙：开启后调用所有api的返回值会在客户端alert出来。（开发者除通过config接口开启调试模式外，也可以使用我们提供的单独接口来开启）

代码示例：

***sx***.onDebug({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 关闭调试模式接口

描叙：关闭调试模式。

代码示例：

***sx***.closeDebug({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 获取插件版本号接口

描叙：获取当前客户端插件SDK的版本号。

代码示例：

sx.getPluginSDKVersion({  
 success: **function** (res) {  
 **var** version = res.**version**; *// 插件版本号* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **version**: **'1.0.0'**}

1. 网络请求接口
   1. 配置网络请求接口

描叙：配置网络请求的相关参数。（注意：所有接口调用都为异步，如需网络请求配置在请求时生效，需保证在configRequest接口回调成功后再去调用startResquest接口；configRequest接口同一应用调用一次，只要页面不关闭，配置始终生效）

代码示例：

**var** staticParams = {  
 **sysType**: **'iOS'**,  
 **sysVer**: **'9.3.1'**,  
 **devicemModel**: **'iPhone8,2'**};  
  
*//调用该接口时必须有一个参数有值,否则调用接口失败***var** params = {  
 **staticParams**: staticParams, *//静态请求参数，适用于网络请求接口每次需要传递固定的一些参数* **timeoutInterval**: 60, *//超时时间，单位秒，整形值，范围5~120，默认20s* **isValidatesSecureCertificate**: **false**, *//是否验证安全证书，默认true* **isValidatesSign**: **true,** *//是否进行签名验证，默认false，true客户端网络请求时对请求参数进行加密生成签名和服务器进行验证，具体签名算法见****附录2****，开发者需要按该算法在自己服务器上对签名进行校验。*

**appId**: **''**, *//应用唯一标识，该参数只在debug调试模式下应用于网络请求签名*

**appSecret**: **''** *//应用安全秘钥，该参数只在debug调试模式下应用于网络请求签名*};  
  
**sx**.configRequest({  
 **params**: params,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

Debug模式说明：注意，appId和appSecret参数只在debug模式下用于网络请求签名，开发者在通过debug模式调试时可传递该参数帮助开发者进行网络请求签名，开发者在上线APP应用时请务必移除这两个参数，尤其appSecret，切勿遗留该参数导致安全秘钥泄露。

* 1. 通过司信客户端请求网络接口

描叙：为了进一步保障Web网络请求的安全和高效性，我们对开发者提供网络请求接口，开发者可以选择掉用该接口通过司信客户端请求网络数据。

代码示例：

**var *requestParams*** = {  
 **type**: 2,  
 **version**: **'1.0.0'**,  
 **device**: 0  
};  
  
**var *params*** = {  
 **url**: **''**, *// 网络请求地址（必传）* **method**: 0, *// 请求方法，0 Get，1 Post，2 MultipartPost（content type = @"multipart/form-data" 类型）默认0* **requestParams**: ***requestParams***, *// 网络请求参数* **cancelTag**: **''**, *// 用于取消网络请求的标记* **isShowLoading**: **true**, *// 是否显示加载loading 默认true* **isUseStaticParams**: **false**, *// 是否使用接口configRequest传递的静态参数，默认true，staticParams为全局变量，如重新设置则覆盖之前的数值，不设置值时为空* **isCloseSignVerification**: **false** *//是否关闭签名验证，默认false，如果接口configRequest设置了开启验证签名,可通过该参数关闭当前网络请求的签名验证*};  
  
**sx**.startRequest({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// 返回请求服务器的结果* },  
 cancel: **function** (err) {  
 *// 网络请求被取消时回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

视各接口具体返回数据格式。最终客户端将数据转换为对象返回。

Debug模式说明：通过ajax进行网络请求，返回数据即为接口返回的真实数据。

* 1. 取消网络请求接口

描叙：取消指定的网络请求。

代码示例：

**var** params = {  
 **cancelTag**: cancelTag *// 用于取消网络请求的标记（必传）*};  
  
***sx***.cancelRequest({  
 **params**: params,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 分享接口
   1. 默认分享接口

描叙：从网页中获取分享内容进行分享。分享入口除了点击右上角选项按钮菜单列表中的 “分享”外，开发者还可以在Web页面添加触发事件，调起分享操作。当开发者在网页中添加分享触发事件后，则客户端会自动关联“分享”按钮点击事件，点击“分享”按钮会回调该接口的trigger函数，当取消分享时会回调cancel函数。（注意：该接口不包含complete回调函数）

代码示例：

sx.share({  
 trigger: **function** (res) {  
 *//点击右上角选项中“分享”按钮回调该函数"* },  
 success: **function** (res) {  
 },  
 cancel: **function** (err) {  
 *//用户取消分享后回调该函数* },  
 fail: **function** (err) {  
 }  
});

* 1. 自定义内容分享接口

描叙：由开发者自定义分享内容。当开发者在网页中添加自定义分享触发事件后，则客户端会自动关联右上角选项中“分享”按钮的点击事件，并开启自定义分享，点击“分享”按钮会回调该接口的trigger函数，当取消分享时会回调cancel函数。（注意：该接口不包含complete回调函数）

代码示例：

**var *params*** = {  
 **title**: **''**, *// 分享标题（不传默认从网页获取）* **desc**: **''**, *// 分享描叙* **link**: **''**, *// 分享链接，当分享的页面为本地页面时，需要传递相对路径，同时开发者还可以在url后面传参数，如/index.html?id=456&title=sixin（不传默认从网页获取）* **imgUrl**: **''**, *// 分享的头图链接* **type**: 2, *// 分享类型，默认0 链接；1 音乐；2 视频* **dataUrl**: **''** *// 如果type是音乐或视频，则需提供数据链接（type为1或2时必传）*};  
  
***sx***.shareCustomContent({  
 **params**: ***params***,  
 trigger: **function** (res) {  
 *//点击右上角选项中“分享”按钮回调该函数""* },  
 success: **function** (res) {  
 },  
 cancel: **function** (err) {  
 *//用户取消分享后回调该函数* },  
 fail: **function** (err) {  
 }  
});

* 1. 开启自定义内容分享功能接口

描叙：调用该接口后，点击右上角选项中“分享”按钮后则分享的是开发者自定义的内容。开发者在不使用接口share，shareCustomContent的前提下可以使用该接口开启自定义分享功能。onMenuShareCustomContent开启自定义分享后，页面切换时再次后退回来会自动恢复默认分享，一般使用场景下建议onMenuShareCustomContent放在deviceReady中调用，保证页面在首次打开或者后退至该页面时都能达到开启自定义分享的作用。（注意：该接口不包含complete回调函数；）

代码示例：

**var *params*** = {  
 **title**: **''**, *// 分享标题（不传默认从网页获取）* **desc**: **''**, *// 分享描叙（不传如是新闻则获取新闻描叙）* **link**: **''**, *// 分享链接，当分享的页面为本地页面时，需要传递相对路径，同时开发者还可以在url后面传参数，如/index.html?id=456&title=sixin（不传默认从网页获取）* **imgUrl**: **''**, *// 分享的头图链接* **type**: 2, *// 分享类型，0 链接；1 音乐；2 视频* **dataUrl**: **''** *// 如果type是音乐或视频，则需提供数据链接（type为1或2时必传）*};  
  
***sx***.onMenuShareCustomContent({  
 **params**: ***params***,  
 enable: **function** (res) {  
 *//开启自定义分享成功回调该函数* },  
 trigger: **function** (res) {  
 *//点击右上角选项中“分享”按钮回调该函数"* },  
 success: **function** (res) {  
 *//分享成功回调该函数* },  
 cancel: **function** (err) {  
 *//用户取消分享后回调该函数* },  
 fail: **function** (err) {  
 }  
});

Debug模式说明：默认回调enable函数。

* 1. 关闭自定义内容分享功能接口

描叙：关闭自定义内容分享功能后，点击右上角选项中“分享”按钮后，分享的为默认从网页获取的内容，不再使用自定义的内容。

代码示例：

***sx***.closeMenuShareCustomContent({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 图像接口
   1. 拍照或从手机相册中选图接口

描叙：调用客户端拍照或从相册选图功能。

代码示例：

**var *params*** = {  
 **count**: 9, *// 图片最多选择数量，默认9张，最大传值9* **sizeType**: 0, *// 返回的图片质量，默认0 原图和压缩图都返回，1 原图，2 压缩图* **sourceType**: 0 *// 图片来源，默认0 相机和相册，1 相机，2 相册*};  
  
***sx***.chooseImage({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// 返回选定照片的本地原图ID列表，localId可以作为img标签的src属性显示图片* **var** originalLocalIds = res.originalLocalIds;  
 *// 返回选定照片的本地压缩图ID列表* **var** compressedLocalIds = res.compressedLocalIds;  
  
 *// 当sourceType为0或者2时，sizeType为0时返回数组originalLocalIds和compressedLocalIds；sizeType为1或者2时返回数组localIds  
 // 当sourceType为1时，sizeType为0时返回originalLocalId和compressedLocalId；sizeType为1或者2时返回localId* },  
 cancel: **function** (err){  
 *//用户取消选图后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

*//当sourceType为0或者2时，sizeType为0时：*

{  
 **originalLocalIds**: [**''**, **''**, **''**],  
 **compressedLocalIds**: [**''**, **''**, **''**]  
}

*//当sourceType为0或者2时，sizeType为1或者2时：*

{  
 **localIds**: [**''**, **''**, **''**]  
}

*// 当sourceType为1时，sizeType为0时：*

{  
 **originalLocalId**: **''**,  
 **compressedLocalId**: **''**  
}

*// 当sourceType为1时，sizeType为1或者2时：*

{  
 **localId**: **''**  
}

Debug模式说明：返回的localId实为图片网络链接，方便开发者在页面上显示图片。

* 1. 头像图片选择接口

描叙：调用客户端拍照或从相册选图功能，选取图片后可修剪用于头像上传。

代码示例：

**var** params = {  
 **sizeType**: 0, *// 返回的图片质量，默认0 原图和压缩图都返回，1 原图，2 压缩图* **sourceType**: 0 *// 图片来源，默认0 相机和相册；1 相机；2 相册*};  
  
**sx**.chooseAvatarImage({  
 **params**: params,  
 success: **function** (res) {  
 *// 返回选定照片的本地原图ID，localId可以作为img标签的src属性显示图片* **var** originalLocalId = res.**originalLocalId**;  
 *// 返回选定照片的本地压缩图ID* **var** compressedLocalId = res.**compressedLocalId**;  
  
 *// 当sizeType为0时返回数组originalLocalId和compressedLocalId，sizeType为1或者2时返回数组localId* },  
 cancel: **function** (err) {  
 *//用户取消选图后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

*//当sizeType为0时：*

{  
 **originalLocalId**: **''**,  
 **compressedLocalId**: **''**}

*//当sizeType为1或者2时：*

{  
 **localId**: **''**}

Debug模式说明：返回的localId实为图片网络链接，方便开发者在页面上显示图片。

* 1. 图片预览接口

描叙：网页中点击图片跳转到客户端图片预览页面。

代码示例：

**var *params*** = {  
 **current**: **''**, *// 当前显示图片的链接,如果不传则默认使用数组第一张图片* **urls**: [**''**, **''**, **''**, **''**] *// 需要浏览的图片url数组（必传）*};  
  
***sx***.previewImage({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 上传图片接口

描叙：客户端执行上传图片操作。

代码示例：

**var *params*** = {  
 **localId**: **''**, *// 需要上传的图片的本地ID，由chooseImage或chooseAvatarImage接口获得（必传）* **isShowProgressTips**: **true** *// 是否显示进度提示，默认true显示，false不显示*};  
  
**sx**.uploadImage({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** serverId = res.**serverId**; *// 返回图片上传至服务器端的ID* },  
 cancel: **function** (err) {  
 *//用户取消上传图片后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **serverId**: **''**}

* 1. 批量上传图片接口

描叙：客户端执行批量上传图片操作。

代码示例：

**var *params*** = {  
 **localIds**: [**''**,**''**,**''**,**''**], *// 需要上传的图片的本地ID数组，由chooseImage接口获得，最大上传数量不超过9张（必传）* **isShowProgressTips**: **true**};  
  
sx.uploadImages({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** serverIds = res.serverIds; *// 返回图片上传至服务器端的ID列表* **var** failedLocalIds = res.failedLocalIds; *// 返回上传失败的图片本地ID列表* },

cancel: **function** (err) {  
 *//用户取消上传图片后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **serverIds**: [**''**, **''**, **''**, **''**, **''**],

**failedLocalIds**: [**''**, **''**, **''**]  
}

* 1. 下载图片接口

描叙：客户端执行下载图片操作。

代码示例：

**var** params = {  
 **serverId**: **''**, *// 需要下载的图片的服务器端ID，由uploadImage或uploadImages接口获得（必传）* **isShowProgressTips**: **true**};  
  
**sx**.downloadImage({  
 **params**: params,  
 success: **function** (res) {  
 **var** localId = res.**localId**; *// 返回图片下载后的本地ID* },  
 cancel: **function** (err) {  
 *//用户取消下载图片后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **localId**: **''**}

Debug模式说明：返回的localId实为图片网络链接，方便开发者在页面上显示图片。

* 1. 批量下载图片接口

描叙：客户端执行批量下载图片操作。

代码示例：

**var *params*** = {  
 **serverIds**: [**''**,**''**,**''**,**''**], *// 需要下载的图片的服务器端ID数组，由uploadImages接口获得，最大下载数量不超过9张（必传）* **isShowProgressTips**: **true**};  
  
sx.downloadImages({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** localIds = res.**localIds**; *// 返回图片下载后的本地ID列表* **var** failedServerIds = res.failedServerIds; *// 返回下载失败的图片服务器ID列表* },

cancel: **function** (err) {  
 *//用户取消下载图片后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **localIds**: [**''**, **''**, **''**, **''**],

**failedServerIds**: [**''**, **''**, **''**]  
}

Debug模式说明：返回的localId实为图片网络链接，方便开发者在页面上显示图片。

1. 音频接口
   1. 开始录音和监听录音自动停止接口

描叙：客户端开始录音，并监听录音自动停止，当录制时间到达最大时限时回调该接口成功函数。（注意：该接口不包含complete回调函数）

代码示例：

**var *params*** = {  
 **maxRecordingTime**: 10 *// 录音最大时长，默认60s,范围1~60整数*};  
  
**sx**.onRecord({  
 **params**: ***params***,  
 enable: **function** (res) {  
 *//录音和监听成功后回调该函数* },  
 success: **function** (res) {  
 *//录音自动停止后回调该函数* **var** localId = res.**localId**; *// 返回音频存储在本地的ID* },  
 refuse: **function** (err) {  
 *//用户拒绝允许访问麦克风回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **localId**: **''**}

Debug模式说明：默认回调enable函数，模拟录音过程，到达最大时限后回调success函数返回localId。

* 1. 停止录音接口

描叙：客户端停止录音。

代码示例：

sx.stopRecord({  
 success: **function** (res) {  
 **var** localId = res.**localId**; *// 返回音频存储在本地的ID* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **localId**: **''**}

* 1. 播放语音和监听语音自动播放完毕接口

描叙：客户端播放语音，并监听语音自动播放完毕。（注意：该接口不包含complete回调函数）

代码示例：

**var *params*** = {  
 **localId**: voice.**localId** *// 需要播放的音频的本地ID，由stopRecord或onVoiceRecordEnd接口获得（必传）*};  
  
**sx**.onPlayVoice({  
 **params**: ***params***,  
 enable: **function** (res) {  
 *//播放和监听成功后回调该函数* },  
 success: **function** (res) {  
 *//语音自动播放完毕后回调该函数* },  
 fail: **function** (err) {  
 }  
});

Debug模式说明：默认回调enable函数。 模拟播放过程，10S后回调success函数。

* 1. 暂停播放接口

描叙：客户端暂停播放录音。

代码示例：

**var *params*** = {  
 **localId**: **''** *// 需要暂停的音频的本地ID，由stopRecord或onVoiceRecordEnd接口获得（必传）*};  
  
***sx***.pauseVoice({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 停止播放接口

描叙：客户端停止播放录音。

代码示例：

**var *params*** = {  
 **localId**: **''** *// 需要停止的音频的本地ID，由stopRecord或onVoiceRecordEnd接口获得（必传）*};  
  
***sx***.stopVoice({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 上传语音接口

描叙：客户端上传语音文件。

代码示例：

**var *params*** = {  
 **localId**: **''**, *// 需要上传的音频的本地ID，由stopRecord或onVoiceRecordEnd接口获得（必传）* **isShowProgressTips**: **true**};  
  
***sx***.uploadVoice({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** serverId = res.**serverId**; *// 返回上传后音频的服务器端ID* },

cancel: **function** (err) {  
 *//用户取消上传语音后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **serverId**: **''**}

* 1. 下载语音接口

描叙：客户端下载语音文件。

代码示例：

**var *params*** = {  
 **serverId**: **''**, *// 需要下载的音频的服务器端ID，由uploadVoice接口获得（必传）* **isShowProgressTips**: **true**};  
  
***sx***.downloadVoice({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** localId = res.**localId**; *// 返回下载后音频的本地ID* },

cancel: **function** (err) {  
 *//用户取消下载语音后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **localId**: **''**}

1. 地理位置
   1. 获取地理位置接口

描叙：获取当前设备的地理位置。

代码示例：

**var *params*** = {  
 **coordinateType**: 0, *// 坐标类型，默认GPS坐标“wgs84”，如果想使用国测局gcj02坐标，可以传1* **isGetAddress**: **true** *// 是否获取地址，默认false不获取*};  
  
***sx***.getLocation({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** latitude = res.**latitude**; *// 纬度，浮点数，范围为90 ~ -90* **var** longitude = res.**longitude**; *// 经度，浮点数，范围为180 ~ -180* **var** altitude = res.**altitude**; *// 海拔，浮点数* **var** speed = res.**speed**; *// 速度，以米/每秒计* **var** altitudeAccuracy = res.**altitudeAccuracy**; *// 海拔精度，当为负值的时候说明altitude无效* **var** accuracy = res.**accuracy**; *// 位置精度，正数* **var** address = res.**address**; *// 详细地址，当isGetAddress为true时返回* },  
 refuse: **function** (err){  
 *//当用户拒绝位置访问或已拒绝状态时回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **latitude**: 40.0431653706774,  
 **longitude**: 116.278956936499,  
 **altitude**: 55.5869140625,  
 **speed**: -1,  
 **altitudeAccuracy**: 42.25038137771658,  
 **accuracy**: 65,  
 **address**: **'北京市海淀区西北旺东路10号院东区16号楼'**}

* 1. 使用司信内置地图查看位置接口

描叙：打开客户端地图查看功能

代码示例：

*// 坐标类型gcj02***var *params*** = {  
 **latitude**: 40.044400036241754, *// 纬度，浮点数，范围为90 ~ -90（必传）* **longitude**: 116.28491337681341, *// 经度，浮点数，范围为180 ~ -180（必传）* **name**: **'软通动力'**, *// 位置名 （必传）* **address**: **'北京市海淀区西北旺东路10号院东区16号楼'**, *// 地址详情说明* **scale**: 0 *// 地图缩放级别,整形值,范围从0~30。默认为0不缩放*};  
  
***sx***.openLocation({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 摇一摇
   1. 开启和监听摇一摇事件接口

描叙：打开客户端摇一摇功能，并监听摇一摇事件。onShake开启摇一摇监听事件后，页面切换时再次后退回来会自动关闭监听，一般使用场景下建议onShake放在deviceReady中调用，保证页面在首次打开或者后退至该页面时都能达到监听摇一摇事件的作用。（注意：该接口不包含complete回调函数）

代码示例：

sx.onShake({  
 enable: **function** (res) {  
 *//开启成功后回调该函数* },  
 trigger: **function** (res) {  
 *//摇一摇事件触发后回调该函数* },  
 fail: **function** (err) {  
 }  
});

Debug模式说明：默认回调enable函数。

* 1. 关闭摇一摇功能接口

描叙：关闭客户端摇一摇功能。

代码示例：

sx.closeShake({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 二维码接口
   1. 调起司信扫一扫接口

描叙：调起客户端扫一扫功能

代码示例：

**var *params*** = {  
 **needHandleResult**: **true**, *// 扫描结果是否需要由司信处理，默认true返回结果的同时并由司信处理，false只返回结果* **scanType**: 0 *// 扫码类型，可以指定扫二维码还是一维码，默认0二者都有，1 二维码，2 一维码*};  
  
***sx***.scanQRCode({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** scanResult = res.scanResult; *// 扫描结果* },  
 cancel: **function** (err) {  
 *//用户取消扫描操作后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **scanResult**: **''**}

* 1. 生成二维码接口

描叙：生成二维码图片。

代码示例：

**var *params*** = {  
 **string**: **''**, *// 需要生成二维码的字符串（必传）* **resolution**: 320, *// 分辨率，整型值，值范围100~1280，默认值400* **iconUrl**:**''** *//需要添加图标的icon链接，传值后自动在二维码上添加icon（icon客户端自动圆角）*};  
  
**sx**.generateQRCode({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// 返回二维码图片存储在本地的ID，localId可以作为img标签的src属性显示图片* **var** localId = res.**localId**;  
 },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **localId**: **''**  
}

Debug模式说明：返回一个网络二维码图片链接地址。

1. 界面操作
   1. 改变导航条标题接口

描叙：改变上导航的标题（注意：司信客户端会自动从Web网页获取标题，如果你已经添加了title标签，请不必在页面加载完毕后调用changeNavTitle设置标题；请不要尝试在window.onload或者sx.deviceReady方法中动态修改title标签值，客户端在这之前已经自动从网页获取完标题，此时修改title值对客户端不起作用；changeNavTitle改变标题后即时生效，页面切换时标题会自动再次更改为从网页获取。）

代码示例：

**var *params*** = {  
 **title**: **'iPSA'** *// 需要显示的导航条标题（必传）*};  
  
***sx***.changeNavTitle({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 隐藏右上角菜单按钮接口

描叙：隐藏右上角菜单按钮

代码示例：

sx.hideOptionMenu({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 显示右上角菜单按钮接口

描叙：显示右上角菜单按钮

代码示例：

sx.showOptionMenu({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 批量隐藏功能按钮接口

描叙：批量隐藏右上角选项菜单里面的功能按钮

代码示例：

**var *menuList*** = [**'2'**, **'3'**];  
**var *params*** = {  
 **menuList**: ***menuList*** *// 需要隐藏的菜单ID 详细菜单列表见附录5（必传）*};  
  
***sx***.hideMenuItems({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 批量显示功能按钮接口

描叙：批量显示右上角选项菜单里面的功能按钮

代码示例：

**var *menuList*** = [**'2'**, **'3'**];  
**var *params*** = {  
 **menuList**: ***menuList*** *// 需要显示的菜单ID 详细菜单列表见附录5（必传）*};  
  
***sx***.showMenuItems({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 显示所有功能按钮接口

描叙：显示右上角选项菜单里面的所有功能按钮

代码示例：

sx.showAllMenuItems({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 关闭当前网页窗口接口

描叙：关闭当前网页窗口

代码示例：

**var** HandlerResult = {  
 **voteTitle**: **''**,  
 **voteType**:1,  
 **voteStatus**:**'已结束'**};  
  
*//传递参数时，说明关闭窗口后开发者需要客户端继续处理其他事情，适用于内部开发者调用***var** params = {  
 **needCompletionHandler**: **true**, *//窗口关闭后是否需要执行其他事情，默认false* **handlerResult**:HandlerResult *//窗口关闭后执行其他事情传给客户端的结果数据，该参数必须为对象*};  
  
**sx**.closeWindow({  
 **params**: params,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 通过司信客户端alert信息接口

描叙：通过司信客户端弹出alert信息

代码示例：

**var *params*** = {  
 **title**: **'关闭网页提示'**, *//alert标题* **message**: **'是否关闭当前网页预览?'**, *//alert详细提示信息* **confirmTitle**: **'关闭'**, *//确认按钮标题* **cancelTitle**: **'取消'** *//取消按钮标题（confirmTitle和cancelTitle两都不传默认显示“确认”和“取消”，二传一只显示一个按钮）*};  
  
**sx**.alertMessage({  
 **params**: ***params***,  
 success: **function** (res) {  
 *//点击确认按钮后回调该函数*  
 },  
 cancel: **function** (err) {  
 *//点击取消按钮后回调该函数*  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 通过司信客户端显示提示信息接口

描叙：通过司信客户端显示提示信息

代码示例：

**var *params*** = {  
 **message**: **'司信JS SDK'**,*//提示信息（必传）*

**type**: 0 *//提示框显示类型，默认0显示带有警示感叹号图标的提示框，1显示带有√ 图标的提示框*

};  
  
**sx**.showMessage({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 通过司信客户端显示loading接口

描叙：通过司信客户端显示loading加载提示框

代码示例：

**var *params*** = {  
 **message**: **'努力加载中...'** *//loading提示信息（不传默认显示“加载中...”）*};  
  
**sx**.showLoading({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 cancel: **function** (err) {  
 *//用户关闭loading提示时回调该函数* },  
 fail: **function** (err) {  
 }  
});

* 1. 通过司信客户端隐藏loading接口

描叙：隐藏客户端显示的loading加载提示框

代码示例：

sx.hideLoading({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 进入机构号
   1. 判断用户是否关注机构号接口

描叙：判断当前用户是否关注该机构号

代码示例：

**var *params*** = {  
 **appId**: **''** *// 应用ID（必传）*};  
  
***sx***.checkIfUserFollowInstitution({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** isUserFollow = res.isUserFollow; *// 检查结果true或者false* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **isUserFollow**: **true**}

* 1. 查看机构号接口

描叙：调用该接口可跳转至查看机构号详情页面

代码示例：

**var *params*** = {  
 **appId**: **''** *// 应用ID（必传）*};  
  
***sx***.viewInstitutionInfo({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 给机构号发信息接口

描叙：调用该接口可直接跳转至于机构号信息发送页面，如用户还未关注该机构号则跳入机构号详情页面。

代码示例：

**var *params*** = {  
 **appId**: **''** *// 应用ID（必传））*};  
  
***sx***.sendMsgToInstitution({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 获取用户及公司信息
   1. 获取司信当前用户接口

描叙：获取司信当前登录用户的信息

代码示例：

**var *params*** = {  
 **isGetDetailInfo**: **false** *// 是否获取详细用户信息，默认false只获取基本信息*};  
**sx**.getUserInfo({

**params**: ***params***,  
 success: **function** (res) {  
 *// res为User对象，isGetDetailInfo为false时只返回基本信息userId，headImgUrl，name字段;为true时返回详细用户信息* **var** userId = res.**userId**; *// 用户ID* **var** headImgUrl = res.headImgUrl; *// 用户头像小图地址* **var** name = res.**name**; *// 姓名  
  
 // isGetDetailInfo为true时才返回以下字段* **var** largeHeadImgUrl = res.largeHeadImgUrl; *// 用户头像大图地址* **var** sex = res.**sex**; *// 性别* **var** mobilePhone = res.mobilePhone; *// 手机号码* **var** telphone = res.telphone; *// 座机号码* **var** email = res.email; *// 电子邮件* **var** sign = res.sign; *// 个性签名* **var** employeeNum = res.employeeNum; *// 员工号* **var** company = res.company; *// 员工所在公司名* **var** department = res.department; *// 员工所在部门* **var** post = res.post; *// 员工职务* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

*// isGetDetailInfo为false时返回*

{  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **name**: **''**  
 }

*// isGetDetailInfo为true时返回*

{  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **largeHeadImgUrl**: **''**,  
 **name**: **''**,  
 **sex**: **''**,  
 **mobilePhone**: **''**,  
 **telphone**: **''**,  
 **email**: **''**,  
 **sign**: **''**,  
 **employeeNum**: **''**,  
 **company**: **''**,  
 **department**: **''**,  
 **post**: **''**}

Debug模式说明：返回有效的个人用户信息，开发者可通过该userId调用服务器接口。

* 1. 获取司信当前用户常用联系人接口

描叙：获取当前用户的常用联系人

代码示例：

**sx**.getUserRecentContacts({  
 success: **function** (res) {  
 **var** contacts = res.contacts; *// 最近联系人列表，最多返回最近9个联系人的数据，无最近联系人时返回空数组* **if** (contacts.**length** > 0){  
 **var** contact = contacts[0];  
 *// contact为最近联系人对象* **var** userId = contact.**userId**; *// 用户ID* **var** headImgUrl = contact.headImgUrl; *// 用户头像地址* **var** name = contact.**name**; *// 姓名* }  
 },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **contacts**: [{  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **name**: **''** }, {  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **name**: **''** }, {  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **name**: **''** }]  
}

* 1. 获取员工信息接口

描叙：获取员工的详细信息。

代码示例：

**var *params*** = {  
 **userId**: **''** *// 员工用户ID，由接口getUserRecentContacts获取（必传）*};  
  
***sx***.getEmployeeInfo({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// res为User对象* **var** userId = res.**userId**; *// 用户ID* **var** headImgUrl = res.headImgUrl; *// 用户头像小图地址* **var** largeHeadImgUrl = res.largeHeadImgUrl; *// 用户头像大图地址* **var** name = res.name; *// 姓名* **var** sex = res.sex; *// 性别* **var** mobilePhone = res.mobilePhone; *// 手机号码* **var** telphone = res.telphone; *// 座机号码* **var** email = res.email; *// 电子邮件* **var** sign = res.sign; *// 个性签名* **var** employeeNum = res.employeeNum; *// 员工号* **var** company = res.company; *// 员工所在公司名* **var** department = res.department; *// 员工所在部门* **var** post = res.post; *// 员工职务* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：见接口getUserInfo

Debug模式说明：返回固定的员工信息，信息并不与开发者传过来的员工ID匹配。

* 1. 获取司信通讯录接口

描叙：客户端弹出通讯录员工选择窗口，提供搜索，单选和多选功能

代码示例：

**var *params*** = {  
 **selectType**: 0 *// 联系人支持多选和单选，默认0多选，1单选*};  
  
***sx***.chooseContacts({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// 当selectType为0时返回联系人列表，每个元素是一个User对象，同接口getUserInfo返回的User对象一致，当selectType为1时返回一个User对象* **var** contacts = res.contacts;  
 },  
 cancel: **function** (err) {  
 *//用户取消选择联系人操作回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **contacts**: [{  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **largeHeadImgUrl**: **''**,  
 **name**: **''**,  
 **sex**: **''**,  
 **mobilePhone**: **''**,  
 **telphone**: **''**,  
 **email**: **''**,  
 **sign**: **''**,  
 **employeeNum**: **''**,  
 **company**: **''**,  
 **department**: **''**,  
 **post**: **''** }, {  
 **userId**: **''**,  
 **headImgUrl**: **''**,  
 **largeHeadImgUrl**: **''**,  
 **name**: **''**,  
 **sex**: **''**,  
 **mobilePhone**: **''**,  
 **telphone**: **''**,  
 **email**: **''**,  
 **sign**: **''**,  
 **employeeNum**: **''**,  
 **company**: **''**,  
 **department**: **''**,  
 **post**: **''** }]  
}

* 1. 选取公司组织接口

描叙：从客户端弹出的组织架构选择列表中选取需要的组织。

代码示例：

sx.chooseOrganization({  
 success: **function** (res) {  
 *// res为Organization对象* **var** orgId = res.**orgId**; *// 组织ID* **var** name = res.**name**; *// 组织名称* **var** memberNumber = res.memberNumber; *// 组织下所属人数* **var** parentId = res.parentId; *// 该组织的上一级组织ID* },  
 cancel: **function** (err) {  
 *//用户取消选择组织后回调该函数* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **orgId**: **''**,  
 **name**: **''**,  
 **memberNumber**: **''**,  
 **parentId**: **''**}

* 1. 获取组织信息接口

描叙：获取组织的详细信息。

代码示例：

**var *params*** = {  
 **orgId**: **''** *// 组织ID，由接口chooseOrganization parentId属性获取（必传）*};  
  
***sx***.getOrganizationInfo({  
 **params**: ***params***,  
 success: **function** (res) {  
 *// res为Organization对象* **var** orgId = res.**orgId**; *// 组织ID* **var** name = res.**name**; *// 组织名称* **var** memberNumber = res.memberNumber; *// 组织下所属人数* **var** parentId = res.parentId; *// 该组织的上一级组织ID* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：见接口chooseOrganization

Debug模式说明：返回固定的组织信息，信息并不与开发者传过来的组织ID匹配。

* 1. 查看员工信息接口

描叙：跳转至指定员工详情页面

代码示例：

**var *params*** = {  
 **userId**: **''** *// 员工用户ID (必传)*};  
  
***sx***.viewEmployeeInfo({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 给员工发送消息接口

描叙：跳转至指定员工的聊天界面

代码示例：

**var *params*** = {  
 **userId**: **''** *// 员工用户ID (必传)*};  
  
***sx***.sendMsgToEmployee({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 设备信息
   1. 获取网络状态接口

描叙：获取当前客户端接入的网络类型。

代码示例：

sx.getDeviceNetworkType({  
 success: **function** (res) {  
 *// 设备当前使用的网络类型 值分别为“Disconnected”，“Wifi”，“4G”，“3G”，“2G”* **var** networkType = res.networkType   
 },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **networkType**: **'Wifi'**}

* 1. 获取设备系统类型接口

描叙：获取设备系统是安卓还是iOS。

代码示例：

sx.getDeviceSystemType({  
 success: **function** (res) {  
 **var** sysType = res.sysType; *// 系统类型，iOS系统值为“iOS”，安卓为“Android”* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **sysType**: **'iOS'**}

* 1. 获取设备系统版本号接口

描叙：获取设备系统的版本号。

代码示例：

sx.getDeviceSystemVersion({  
 success: **function** (res) {  
 **var** sysVersion = res.sysVersion; *// 系统版本号* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **sysVersion**: **'9.3'**}

* 1. 获取设备机型接口

描叙：获取设备的型号。

代码示例：

sx.getDeviceModel({  
 success: **function** (res) {  
 **var** model = res.model; *// 机型，iOS平台会直接返回类似“iPhone8,2”的标记，需要开发者自行处理* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **model**: **'iPhone8,2'**}

* 1. 获取设备唯一标识接口

描叙：获取设备的唯一标识。

代码示例：

sx.getDeviceUniqueID({  
 success: **function** (res) {  
 **var** uniqueId = res.uniqueId; *// 设备唯一标识* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **uniqueId**: **''**}

* 1. 获取设备信息接口

描叙：获取设备12.1至12.5所有信息。

代码示例：

sx.getDeviceInfo({  
 success: **function** (res) {  
 *// 返回设备信息接口中12.1至12.5所有信息* **var** networkType = res.networkType  
 **var** sysType = res.sysType;  
 **var** sysVersion = res.sysVersion;  
 **var** model = res.model;  
 **var** uniqueId = res.uniqueId;  
 },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **networkType**: **'Wifi'**,  
 **sysType**: **'iOS'**,  
 **sysVersion**: **'9.3'**,  
 **model**: **'iPhone8,2'**,  
 **uniqueId**: **''**}

1. 数据缓存接口
   1. 使用司信客户端缓存数据接口

描叙：将数据缓存到司信客户端本地

代码示例：

**var *backupData*** = {  
 **name**: **'张三'**,  
 **sex**: **'男'**,  
 **age**: **'18'**};  
  
**var *params*** = {  
 **key**: **''**, *// 缓存数据使用的key值，必须为字符串类型 (必传)* **value**: ***backupData***, *// 需要缓存的数据，支持数字、数组、对象以及多媒体文件图片和语言的缓存，当为多媒体文件时传递文件存储的localId，由4或者5中的接口得到 (必传)* **type**: 0 *//缓存的数据类型，默认0 数字、数组或者对象，1 图片，2 语音*};  
  
**sx**.backupData({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 使用司信客户端读取缓存接口

描叙：读取缓存到司信客户端本地的数据。开发者需要注意，多媒体文件最多缓存时间为30天，30天后客户端会自动进行清理。

代码示例：

**var *params*** = {  
 **key**: **''**, *// 读取缓存数据的key值 (必传)* **type**: 0 *//该缓存数据的类型，默认0 数字、数组或者对象，1 图片，2 语音 (注意:多媒体文件最多缓存时间为30天)*};  
  
**sx**.restoreData({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** value = res.**value**; *//当数据类型为图片或者语音时,会返回该多媒体文件缓存在客户端的localId* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

*//当缓存数据为对象时：*

{  
 **value**: {}}

*//当缓存数据为字符串时：*

{  
 **value**: **''**}

*//当缓存数据为多媒体文件时：*

{  
 **value**: **'http://172.16.8.85:8080/Temp/Images/SXIMG\_2016091717503.jpg'**}

* 1. 移除司信客户端缓存数据接口

描叙：通过指定KEY值移除缓存在客户端本地的数据。我们希望开发者在借用司信客户端缓存数据的同时，应当秉着为用户节省存储空间的原则，在数据用完后第一时间调用移除缓存接口移除不使用的缓存数据。如不清除，30天后客户端自动进行清理。

代码示例：

**var *params*** = {  
 **key**: **''**, *// 读取缓存数据的key值 (必传)* **type**: 0 *//该缓存数据的类型，默认0 数字、数组或者对象，1 图片，2 语音*};  
  
**sx**.removeData({  
 **params**: ***params***,  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

* 1. 移除当前应用在司信客户端所有缓存数据接口

描叙：移除当前应用缓存在客户端的所有数据。

代码示例：

sx.removeAllData({  
 success: **function** (res) {  
 },  
 fail: **function** (err) {  
 }  
});

1. 支付接口
   1. 使用司信客户端调用微信支付宝支付接口

描叙：通过司信客户端调起微信或者支付宝进行支付。

支付接口只会返回三种结果，交易成功、操作成功，交易结果待确认、取消支付，前两者回调success，取消支付回调cancel。需要说明的是，跳转至于微信或者支付宝平台进行支付时，如果取消支付，或者支付失败，接口均不会给Web返回结果，客户端会依然停留在支付选择页面，方便用户继续点击支付，除非用户关闭当前支付选择页面，此时接口会回调cancel函数；

isShowResultPage参数意在告诉客户端是否显示交易结果页面，当isShowResultPage为true时，客户端会帮助Web显示支付结果页面，此时建议开发者在success和cancel回调里面直接切换页面到订单详情页面，如开发者想自己自定义结果页面可将isShowResultPage设为false，此时开发者可在success和cancel回调里面做结果显示操作。

代码示例：

**var *params*** = {  
 **outTradeNo**: **'1476791226840816'**, *// 订单号 (必传)* **totalFee**: 7188, *//总价格，大于0的浮点数，小数点后最多两位(必传),* **isShowResultPage**: **true** *//是否显示支付结果页，默认true支付完成后跳转至支付结果页面，false不跳转*};  
  
**sx**.pay({  
 **params**: ***params***,  
 success: **function** (res) {  
 **var** payResult = res.payResult; *//支付结果，OK 当前交易成功，TBC 支付操作成功，交易结果待确认* },  
 cancel: **function** (err) {  
 *//用户关闭支付页面取消当前支付回调该函数* },  
 fail: **function** (err) {  
 *//微信支付宝支付失败后并不回调该函数，如果SDK回调该函数说明调用接口出错* }  
});

返回数据格式示例：

{  
 **payResult**: **'TBC'**}

Debug模式说明：Debug模式下无法真实模仿支付场景，只会返回“TBC”的假数据。

1. 其他附加接口
   1. 获取司信客户端版本号接口

描叙：获取客户端的版本号

代码示例：

sx.getAppVersion({  
 success: **function** (res) {  
 **var** version = res.**version**; *// 司信版本号* },  
 fail: **function** (err) {  
 }  
});

返回数据格式示例：

{  
 **version**: **'1.0'**}

* **附录**

**附录1：JS-SDK使用权限签名算法**

签名需要用到appSecret参数，appSecret是开发者在申请机构号应用时系统生成的安全秘钥，可在机构号运营平台查看，开发者必须妥善保管，切勿泄露。

**签名算法**

签名生成规则如下：参与签名的字段包括有效的appSecret，nonceStr（随机字符串），timestamp（时间戳）， url（当前网页的URL，不包含#及其后面部分），对所有待签名参数按照字段名的ASCII 码从小到大排序（字典序）后，使用URL键值对的格式（即key1=value1&key2=value2…）拼接成字符串string1。对string1作sha1加密，字段名和字段值都采用原始值，不进行URL 转义。

即signature=sha1(string1)。 示例：

* appSecret=VMCPGGVi4C3VM0P37wVUCFvkVAy\_90u5h9nbSlYy3
* nonceStr=yJm3WZXDP76z0ccnW
* timestamp=1414587457
* url= http://www.isoftstone.com/

步骤1. 对所有待签名参数按照字段名的ASCII 码从小到大排序（字典序）后，使用URL键值对的格式（即key1=value1&key2=value2…）拼接成字符串string1：

appSecret=VMCPGGVi4C3VM0P37wVUCFvkVAy\_90u5h9nbSlYy3&nonceStr=yJm3WZXDP76z0ccnW&timestamp=1414587457&url=http://www.isoftstone.com/

步骤2. 对string1进行sha1签名，得到sign：

9d9a466f581b96ead7f2e35e6e82888d97bbeab4

**注意事项**

* 签名用的nonceStr和timestamp必须与config接口中传递的nonceStr和timestamp相同。
* 签名用的url必须是调用插件接口页面的完整URL。
* 出于安全考虑，开发者必须在服务器端实现签名的逻辑。

**附录2：网络请求签名算法**

签名需要用到appSecret参数，和附录1使用的appSecret一致，开发者必须妥善保管，切勿泄露。

**签名算法**

签名生成规则如下：将所有客户端传给服务端的参数（签名sign除外）按照字段名的ASCII 码从小到大排序（字典序）后，使用URL键值对的格式（即key1=value1&key2=value2…）拼接成字符串string1，再将string1和AppSecret拼接成字符串string2，然后对string2进行MD5加密得到sign

即sign=md5(string2)。

如开发者的机构号应用AppSecret为168253D5F0614255907E36FA5D5C14A1，在调用某个接口时需要传递userId参数，则客户端传给服务端的所有参数如下：

* Appkey=EDE64D0C559947809J31C15F2717632F
* timeStamp=1414587457
* uuid=05171F2491594E9685EAA3208A79E54D
* randomStr=12345678
* userId=45
* sign=d04d4c3ab0d952b380f5e2ca73c82604

需要说明的时，除userId以外的其他参数都是客户端用于签名所自动添加的参数，开发者在服务端接收参数时不需要太关心，只需要按签名算法将各个参数拼接使用即可。

* + AppKey：机构号应用的唯一标示，即AppId；
  + timeStamp：时间戳；
  + uuid：无序字符串；
  + randomStr：8位随机字符串；

**签名示例**

步骤1. 对所有待签名参数按照字段名的ASCII 码从小到大排序（字典序）后，使用URL键值对的格式（即key1=value1&key2=value2…）拼接成字符串string1：

Appkey=EDE64D0C559947809J31C15F2717632F&randomStr=12345678&timeStamp=1414587457&userId=45&uuid=05171F2491594E9685EAA3208A79E54D

步骤2. 将string1和AppSecret拼接成字符串string2：

Appkey=EDE64D0C559947809J31C15F2717632F&randomStr=12345678&timeStamp=1414587457&userId=45&uuid=05171F2491594E9685EAA3208A79E54D168253D5F0614255907E36FA5D5C14A1

步骤3. 对string2进行MD5加密，得到sign：

d04d4c3ab0d952b380f5e2ca73c82604

**附录3：结果代码说明**

* + 0：调用接口成功回调success函数代码
  + 01：开启某个功能或监听某事件成功回调enable函数代码。
  + 02：开启某个功能或监听某事件后触发了该事件回调trigger函数代码。
  + 03：用户取消当前接口操作回调cancel函数代码。
  + 04：访问地理位置，麦克风等用户隐私服务时被用户拒绝访问后回调refuse函数代码。
  + 40000：调用JS SDK接口访问权限未授权错误代码。请先通过config接口获取访问权限。如签名算法没有错误，请仔细检查传递参数是否有误。
  + 40001：无法找到客户端插件方法错误代码。当前客户端版本过低，客户端插件我支持该JS SDK接口方法。
  + 40002：configRequest接口还未回调成功函数，网络请求配置未生效，开发者暂时无法调用startRequest接口。JS SDK各接口调用都是异步操作，为了保证configRequest的配置项能够在网络请求中生效，开发者需要在configRequest成功回调函数中调用startRequest接口进行网络请求。
  + 40003：本地网页模式下相关接口无需调用错误代码。当应用为zip包形式的本地应用时，开发者不需要调用config接口获取接口权限。
  + 40100：无参数错误代码。当该接口需要参数时而开发者没有传递任何参数时报该错误，请检查是否忘记传递参数，或者是否传递了null 和undefined。
  + 40101：参数格式无效错误代码。请检查返回的错误参数是否按规定格式进行传递，是否为要求的字符串，数组类型，对象类型。
  + 40102：必要参数缺失错误代码。请检查返回的错误参数是否传递，或者是否传递了null 和undefined，是否空字符串，空数组等。
  + 41000：无网络连接错误代码。请先连接网络。目前SDK1.0.0版本接口2.2，4.4，4.5，4.6，4.7，5.8，5.9，6.1（当isGetAddress为true需要获取地址时），10.1，10.2，10.3（10.1，10.2，10.3当客户端未拿到机构号数据时），11.3（当客户端还未获取到该用户详细信息时），以及config接口需要接入网络。
  + 41001：网络请求失败错误代码。请根据错误信息查找原因，检查网络是否稳定，或者尝试重新请求网络。
  + 41002：无效的appId错误代码。请确认当前使用的appId是否存在。
  + 41003：无效的访问域名错误代码。请确认当期访问的url域名是否已在机构号运营平台进行过绑定授信。
  + 41004：无效的签名错误代码。请确认当期签名是否正确，注意url为当前访问url，不包含#后面的部分，保证签名使用的参数和config中使用的参数一致。
  + 41005：服务器JS SDK权限验证时系统出错错误代码。请尝试重新调用config接口进行权限验证。
  + 41006：保存文件到本地失败错误代码。请尝试重新调用接口。
  + 41007：处理图片失败错误代码。请尝试重新调用接口进行图片相关操作，或者重启客户端再试。
  + 41008：批量上传和下载图片数量超过最大限制错误代码。请保证图片处理数量不要超过最大限制。
  + 41009：上传文件失败错误代码。请尝试重新上传文件。
  + 41010：下载文件失败错误代码。请尝试重新下载文件。
  + 41011：无效的图片或音频localId错误代码。请确认localId是否正确。
  + 41012：没有可以停止的录音错误代码。请先调用startRecord方法开始录音。
  + 41013：初始化播放器失败错误代码。请尝试退出页面重新进入后再播放；尝试关闭其他后台运行的音频播放器类客户端。
  + 41014：当前语音未被初始化播放错误代码。当前语音并未初始化准备播放，无法执行暂停和停止播放操作。请先调用onPlayVoice初始化播放该段语音。
  + 41015：定位服务不可用错误代码。请检查您的设备是否支持定位服务。
  + 41016：无效的位置错误代码。当获取到的位置精度为负值时会报该错误，说明当前设备获取的位置无效，请尝试重新获取。
  + 41017：系统获取位置失败错误代码。当系统获取位置失败时会报该错误，请尝试重新获取。
  + 41018：系统获取地址address失败错误代码。当系统获取地址失败时会报该错误，请尝试重新获取。
  + 41019：当前设备不支持摇一摇错误代码。请停止调用该接口。
  + 41020：菜单列表menuList数组中所有menuId无效错误代码。请确认menuList中的menuId是否正确。
  + 41021：不可用的机构号错误代码。请确认当期机构号应用是否审批通过或者是否被下架。
  + 41022：无效的用户userId错误代码。请确认userId是否正确。
  + 41023：无效的组织orgId错误代码。请确认orgId是否正确。
  + 41024：缓存文件失败错误代码。可能由于系统原因导致图片和语言缓存失败，请尝试重新调用接口缓存。
  + 41025：读取缓存失败错误代码。请检查读取缓存的Key值是否正确，如正确则很有可能客户端缓存已被清理，请不要再尝试读取缓存。

**附录4：接口列表**

* + Basic|checkPluginApi
  + Basic|onDebug
  + Basic| closeDebug
  + Basic|getPluginSDKVersion
  + Network|configRequest
  + Network|startRequest
  + Network|cancelRequest
  + Share|share
  + Share|shareCustomContent
  + Share|onMenuShareCustomContent
  + Share|closeMenuShareCustomContent
  + Image|chooseImage
  + Image|chooseAvatarImage
  + Image|previewImage
  + Image|uploadImage
  + Image|uploadImages
  + Image|downloadImage
  + Image|downloadImages
  + Audio|onRecord
  + Audio|stopRecord
  + Audio|onPlayVoice
  + Audio|pauseVoice
  + Audio|stopVoice
  + Audio|uploadVoice
  + Audio|downloadVoice
  + Location|getLocation
  + Location|openLocation
  + Shake|onShake
  + Shake|closeShake
  + QRCode|scanQRCode
  + QRCode|generateQRCode
  + InterfaceOperation|changeNavTitle
  + InterfaceOperation|hideOptionMenu
  + InterfaceOperation|showOptionMenu
  + InterfaceOperation|hideMenuItems
  + InterfaceOperation|showMenuItems
  + InterfaceOperation|showAllMenuItems
  + InterfaceOperation|setOptionMenuWithShareEvent
  + InterfaceOperation|setOptionMenuWihtDefaultEvent
  + InterfaceOperation|closeWindow
  + InterfaceOperation|alertMessage
  + InterfaceOperation|showMessage
  + InterfaceOperation|showLoading
  + InterfaceOperation|hideLoading
  + Institution|checkIfUserFollowInstitution
  + Institution|viewInstitutionInfo
  + Institution|sendMsgToInstitution
  + UserInfo|getUserInfo
  + UserInfo|getUserRecentContacts
  + UserInfo|getEmployeeInfo
  + UserInfo|chooseContacts
  + UserInfo|chooseOrganization
  + UserInfo|getOrganizationInfo
  + UserInfo| viewEmployeeInfo
  + UserInfo |sendMsgToEmployee
  + Device|getDeviceNetworkType
  + Device|getDeviceSystemType
  + Device|getDeviceSystemVersion
  + Device|getDeviceModel
  + Device|getDeviceUniqueID
  + Device|getDeviceInfo
  + Device|getDeviceInfo
  + LocalStorage|backupData
  + LocalStorage|restoreData
  + Pay|pay
  + Additions|getAppVerInfo

**附录5：菜单项ID列表**

* + 分享：1
  + 收藏：2
  + 邮件：3
  + 在浏览器中打开：4

**附录6：Demo页面和示例代码**

**Demo页面：**

<http://jssdk.issmobile.com/JS-SDK/demo/jssdkdemo.html>



**示例代码：**

<http://jssdk.issmobile.com/JS-SDK/files/demo/demo.zip>