**九五云客服**

**电话条与监控SDK集成开发手册**

北京鸿联九五信息产业有限公司

**版本信息**

|  |  |
| --- | --- |
| 文档编号： | HL-CC-026 |
| 版本号： | V1.9 |
| 撰写人： | 张少旭 |
| 撰写日期： | 2018-11-28 |

**版本修改历史**

| 版本编号. | 修订人 | 修订日期 | 修订描述 |
| --- | --- | --- | --- |
| V1.0 | 张少旭 | 2018-11-28 | 建立初始版本 |
| V1.1 | 张少旭 | 2019-01-28 | 初始化参数增加技能组和签入技能组参数和功能 |
| V1.2 | 张少旭 | 2019-05-29 | 文档格式调整 |
| V1.3 | 张少旭 | 2019-11-22 | 细化“术语”章节中部分枚举值的说明 |
| V1.4 | 张少旭 | 2020-06-08 | 拨打电话指令增加透传参数、任务id等业务参数 |
| V1.5 | 张少旭 | 2020-08-12 | 振铃、挂机、弹屏事件中增加外显号码和透明传递参数 |
| V1.6 | 林娟 | 2021-04-08 | 增加监控SDK集成 |
| V1.7 | 张少旭 | 2021-09-09 | 电话指令增加二次拨号功能  增加密码加密传输说明 |
| V1.8 | 张少旭 | 2021-12-09 | 增加话后自动就绪参数 |
| V1.9 | 张少旭 | 2022-05-06 | 增加按钮控制的说明 |

目录

[1 前言 4](#_Toc4584)

[1.1 预期读者 4](#_Toc27237)

[1.2 使用指南 4](#_Toc31138)

[1.3 开发模式 4](#_Toc894)

[1.4 术语 4](#_Toc31266)

[2 PhoneBar 5](#_Toc26809)

[2.1 引用文件 5](#_Toc14302)

[2.2 创建组件 6](#_Toc607)

[2.2.1 属性 6](#_Toc8443)

[2.3 组件方法 8](#_Toc6397)

[2.3.1 Destroy(销毁组件) 8](#_Toc2789)

[2.3.2 按钮组件 8](#_Toc26392)

[2.4 事件 9](#_Toc15788)

[2.4.1 onAgentStatusChange (坐席状态变更) 9](#_Toc13718)

[2.4.2 onScreenPopup (弹屏事件) 9](#_Toc25428)

[2.4.3 onRinging (振铃事件) 10](#_Toc16476)

[2.4.4 onTalking (接通事件) 10](#_Toc17003)

[2.4.5 onHangup(挂机事件) 11](#_Toc19699)

[2.4.6 onLinkDisconnected (与服务器断开连接) 11](#_Toc9905)

[2.5 坐席api 12](#_Toc27434)

[2.5.1 坐席状态 12](#_Toc12176)

[2.5.2 电话指令 13](#_Toc22557)

[2.6 可定制方法 18](#_Toc15278)

[2.6.1 工具库 18](#_Toc32253)

[2.6.2 日志 19](#_Toc8693)

[2.7 密码加密 19](#_Toc8079)

[2.7.1 普通加密 23](#_Toc11180)

[2.7.2 随机码混淆加密 24](#_Toc29764)

[3 九五云客服监控SDK 25](#_Toc8354)

[3.1 引用文件 25](#_Toc25161)

[3.2 集成 25](#_Toc32127)

[3.2.1 使用九五云客服话条SDK PhoneBar集成 25](#_Toc11793)

[3.2.2 使用自定义话条集成 25](#_Toc28074)

# 前言

## 预期读者

本文档适用于呼叫中心电话功能开发的相关开发人员。

## 使用指南

通过调用下文所述的JavaSciprt SDK可以定制化开发话务功能，达到客户特定的应用需求.

## 开发模式

在工程中引入PhoneBar.js和PhoneBar.css即可生成一个电话条，如果UI样式与当前项目风格不一致，可以通过覆盖PhoneBar.css定制UI显示效果。

Agentbar 有两种开发模式，有界面模式和无界面模式。

1、 有界面模式

有界面模式只需要把控件的高位设为 90，宽设为>450，将会显示如下界面（图1-1）。



图1-1

2、 无界面模式

无界面模式只需要把控件的高位设为 0，宽 0 即可。

两种模式下共用一套接口，实现客户端与服务端的交互。

## 术语

|  |  |
| --- | --- |
| 名称 | 描述 |
| Agent | 一个坐席 |
| AgentStatus | 坐席状态,分为两种表示形式：  服务端通信协议定义的状态值：  0：坐席登出，电话功能暂不可用  1：坐席就绪，可以拨动电话，接听电话  2：坐席未就绪，可以呼出电话，不能接听电话(除了内部呼叫和转接)，未就绪状态的定义可参考文档中改名词的描述  本地易于理解的状态枚举：  'offline'：离线，对应服务器状态0 'ready'：就绪，对应服务器状态1 'busy'：示忙，对应服务器状态2，未就绪状态值3 'resting'：休息，对应服务器状态2，未就绪状态值5 'neatening'：整理中，对应服务器状态2，未就绪状态值0 'talking'：通话中，对应服务器状态2，未就绪状态值1 'ringing'：振铃，对应服务器状态2，未就绪状态值6 |
| thisDN | 坐席的分机号码.thisDN表示当前分机号码 |
| agentID | 坐席的工号，与分机号一致 |
| Line | 线路，SIP支持同时多路电话，所以模拟了线路的概念，即一路电话为一个线路，默认只能有一个线路为通话中，其他必须不在通话中或者保持状态。 |
| lineState | 线路的状态：  'idle'：空闲,  'dialing'：外呼振铃  'ringing'：呼入响铃  'talking'：通话  'held'：保持 |
| CallID | 一通电话的唯一ID，不管当前有多少参与人，参与这种电话 |
| CallType | 呼叫类型, 表示电话的方向：  0: 未知类型 1: 内部通话 2: 客户呼入 3: 手动呼出 4: 电话转接 5: 多方通话 6: 预约回呼 7: 电话回访 8: 预测外呼 9: 精确预览 10: 网页电话 11: 电话监听 |

# PhoneBar

## 引用文件

*<!--引入css-->*

<link rel="stylesheet" type="text/css" href="./css/PhoneBar.css">

*<!--引入js-->*

<script src='./PhoneBar.js'></script>

## 创建组件

创建组件时可以配置服务器信息、坐席信息、坐席配置信息和自定义事件回调等，该组件提供了默认配置，可以只写入服务器连接和坐席账号即可生成电话条。如下示例：

***phoneBar*** = **new** PhoneBar({  
 **renderTo**: **'test1'**,  
 **proxyUrl**: **'ws://192.168.1.111:8787/websocket'**,  
 **sipServerUrl**: **'192.168.1.111:5188'**,  
 **startupSoftPhone**: **true**,  
  
 **tid**: **'xxxxx'**,  
 **thisDN**: **'xxxxx'**,  
 **agentID**: **'xxxxx'**,  
 **password**: **'xxxxx'**,  
 **thisQueues**: [**'xxxxx', 'xxxxx'**],  
 **defaultQueue**: **'xxxxx'**,

**autoIdleWhenLogin**: **true**,  
 **isPhoneTakeAlong**: **false**,   
  
 onLinkDisconnected: **function**() {***location***.href = **'login.html'**}  
})

### 属性

PhoneBar 的属性较多，一般情况下，在开发中需要设置基本属性，其他属性使用默认值就可以生产标准的电话条。 设置渲染位置renderTo， 设置 CTI 服务器地址proxyUrl，软电话网管地址softPhoneServerUrl，设置坐席基本信息tid，thisDN， agentID，password ，thisQueues，defaultQueue。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| renderTo | String | 是 | body | 页面元素id，渲染到置顶元素内，默认追加到body内。创建对象前需保证改dom对象已存在。 |
| proxyUrl | String | 是 | ws:// 127.0.0.1:8787/websocket | CTI 服务器地址 |
| startupSoftPhone | Boolean | 是 |  | 是否自动启动软电话，如果自动启动请配置软电话服务地址 |
| sipServerUrl | String | 否 |  | SIP服务地址 |
| tid | String | 是 |  | 租户ID |
| thisDN | String | 是 |  | 分机号 |
| pstnDN | String | 是 |  | PSTN号码，可以为null |
| agentID | String | 是 |  | 坐席的工号，与分机号一致 |
| password | String | 否 |  | 密码 |
| loginType | Number | 否 | 0 | 密码类型，默认不加密，详见密码加密说明：  0 不加密;  1 普通加密;  2 随机码混淆加密; |
| thisQueues | Array | 否 |  | 所在技能组  格式如：[100018000,100018001] 或  [100019000]  当defaultQueue=1时可以为空 |
| defaultQueue | String | 是 |  | 默认技能组，默认值”1”  不等于1时值必须是thisQueues数组中的其中一个 |
| autoIdleWhenAfterWork | boolean | 否 | false | 通话后自动置闲  null 使用服务端配置  true 开启  false 关闭 |
| autoIdleWhenLogin | boolean | 否 | true | 登录后自动置闲 |
| isPhoneTakeAlong | boolean | 否 | false | 是否手机随行，即手机在线 |
| workPhone | String | 否 |  | 随行手机号 |
| autoAnswer | Boolean | 否 | false | 自动应答，软电话协议预留 |
| onAgentStatusChange | Function | 否 |  | [坐席状态变更事件](#_onAgentStatusChange_(座席状态变更)) |
| onScreenPopup | Function | 否 |  | [弹屏事件](#_onScreenPopup_(弹屏事件)) |
| onRinging | Function | 否 |  | [振铃事件](#_onRinging_(振铃事件)) |
| onTalking | Function | 否 |  | [接通事件](#_onTalking_(接通事件)) |
| onHangup | Function | 否 |  | [挂机事件](#_onHangup(挂机事件)) |
| onLinkDisconnected | Function | 否 |  | [连接被服务器断开事件](#_onLinkDisconnected_(与服务器断开连接)) |

## 组件方法

Phonebar 的方法较多，但一般我们采用有界面的开发模式，可以大大简化开发难度，您只需要使用到构造方法、destroy。

### Destroy(销毁组件)

【功能】

该方法会删除页面上显示的电话条元素、绑定的事件和断开与CTI服务器的连接

【函数说明】

void destroy()

【参数说明】

无

【返回值】

无

### 按钮组件

话条有多个按钮排列组成，每个按钮绑定唯一key，通过key可获得对应的按钮，按钮组件中可通过保留公共函数控制按钮组件的显示效果。

【功能】

该方法使用getComponent获取对应按钮。

【函数说明】

Component getComponent(key)

【参数说明】

|  |  |
| --- | --- |
| 按钮key | 按钮名称 |
| openDialPad | 呼出 |
| answer | 接听 |
| hangup | 挂断 |
| hold | 保持 |
| retrieve | 接回 |
| transfer | 转接 |
| rollout | 转出 |
| conference | 会议 |

【返回值】

根据以上key 返回对应的组件

#### 按钮显示

【功能】

该方法使用getComponent获取对应按钮组件，通过show方法显示改按钮。

【函数说明】

void show()

【参数说明】

无

【返回值】

无

#### 按钮隐藏

【功能】

该方法使用getComponent获取对应按钮组件，通过hide方法隐藏改按钮。

【函数说明】

void hide()

【参数说明】

无

【返回值】

无

## 事件

电话条的所有功能与业务时完全分离的，只能满主普通的话务需求，想在嵌入系统后做定制化的业务，可以通过监听响应的事件进行扩展。

### onAgentStatusChange (坐席状态变更)

【事件说明】

当坐席的状态发生改变时触发此事件。

【事件定义】

onAgentStatusChange (status, beforeStatus)

【参数说明】

|  |  |  |
| --- | --- | --- |
| 属性名称 | 类型 | 描述 |
| status | String | 坐席当前状态值，参考[1.4术语](#_术语)描述 |
| beforeStatus | String | 变更前的状态值，参考[1.4术语](#_术语)描述 |

### onScreenPopup (弹屏事件)

【事件说明】

弹屏事件

【事件定义】

onScreenPopup (lineState, callInfo)

【参数说明】

|  |  |  |
| --- | --- | --- |
| 属性名称 | 类型 | 描述 |
| line.lineState | String | 当前线路状态值，参考[1.4术语](#_术语)描述 |
| callInfo | Object | 通话信息 |
| callInfo.callType | String | 呼叫类型值，参考[1.4术语](#_术语)描述 |
| callInfo.callId | String | 一通电话的唯一ID |
| callInfo.phoneNumber | String | 对方电话号码 |
| callInfo.dnis | String | 外显号码 |
| callInfo.talkSec | Number | 通话时长，挂机消息中可获取到此参数 |
| callInfo.attachDatas | Object | 附加数据 |
| callInfo.attachDatas.trans\_para | String | 透名传递参数 |

### onRinging (振铃事件)

【事件说明】

电话振铃时触发此事件，此处振铃分为两种，可以通过lineState或者callType区分是呼入振铃还是呼出振铃

【事件定义】

onRinging(callInfo)

【参数说明】

|  |  |  |
| --- | --- | --- |
| 属性名称 | 类型 | 描述 |
| callInfo | Object | 通话信息 |
| callInfo.callType | String | 呼叫类型值，参考[1.4术语](#_术语)描述 |
| callInfo.callId | String | 一通电话的唯一ID |
| callInfo.phoneNumber | String | 对方电话号码 |
| callInfo.dnis | String | 外显号码 |
| callInfo.attachDatas | Object | 附加数据 |
| callInfo.attachDatas.trans\_para | String | 透名传递参数 |

### onTalking (接通事件)

【事件说明】

电话接通后触发此事件

【事件定义】

onTalking(callInfo)

【参数说明】

|  |  |  |
| --- | --- | --- |
| 属性名称 | 类型 | 描述 |
| callInfo | Object | 通话信息 |
| callInfo.callType | String | 呼叫类型值，参考[1.4术语](#_术语)描述 |
| callInfo.callId | String | 一通电话的唯一ID |
| callInfo.phoneNumber | String | 对方电话号码 |
| callInfo.dnis | String | 外显号码 |
| callInfo.attachDatas | Object | 附加数据 |
| callInfo.attachDatas.trans\_para | String | 透名传递参数 |

### onHangup(挂机事件)

【事件说明】

挂机事件

【事件定义】

onHangup(callInfo)

【参数说明】

|  |  |  |
| --- | --- | --- |
| 属性名称 | 类型 | 描述 |
| callInfo | Object | 通话信息 |
| callInfo.callType | String | 呼叫类型值，参考[1.4术语](#_术语)描述 |
| callInfo.callId | String | 一通电话的唯一ID |
| callInfo.phoneNumber | String | 对方电话号码 |
| callInfo.dnis | String | 外显号码 |
| callInfo.talkSec | Number | 通话时长 |
| callInfo.attachDatas | Object | 附加数据 |
| callInfo.attachDatas.trans\_para | String | 透名传递参数 |

### onLinkDisconnected (与服务器断开连接)

【事件说明】

CTI服务器的连接断开后触发此事件

【事件定义】

onLinkDisconnected ()

【参数说明】

无

## 坐席api

一般集成开发中使用组件功能无法满足需要时，可以使用坐席与CTI的操作API自定义业务实现。创建PhoneBar实例后默认会创建AgentApi对象，通过AgentApi对象实现与CTI服务器的交互。比如PhoneBar的示例为phoneBar想要拨打电话，可以使用phoneBar.agentApi.makeCall(dest, id, type)即可呼出一通电话。

### 坐席状态

坐席的状态是一个典型的有限状态机。共有三个状态Logout,Ready,NotReady。如下图所示：图2-4-1 (黑色字体为状态，蓝色字体为动作)



图2-1

#### agentLogin（坐席登录）

【功能】

坐席登录

【函数说明】

void agentLogin ()

【参数说明】

无

【返回值】

无

#### agentLogout（坐席退出）

【功能】

坐席退出

【函数说明】

void agentLogout()

【参数说明】

无

【返回值】

无

#### agentNotReady（坐席非就绪）

【功能】

坐席设置为非就绪状态，原因码设置具体非就绪原因

【函数说明】

void agentNotReady(reasonCode)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| reasonCode | String | 是 |  | 修改为非就绪的原因，原因码参考[1.4术语AgentStatus](#_术语)描述  0 整理，1 通话，2 话机不可用，3 示忙5 休息 |

【返回值】

无

#### agentReady（坐席就绪）

【功能】

坐席设置为就绪状态

【函数说明】

void agentReady ()

【参数说明】

无

【返回值】

无

### 电话指令

在CTI中，电话状态主要是参与人Party的有限状态机。Party有这个5个IDLE、Dialing、Ringing、Talk、Held状态，分布如图所示（图2-2）:

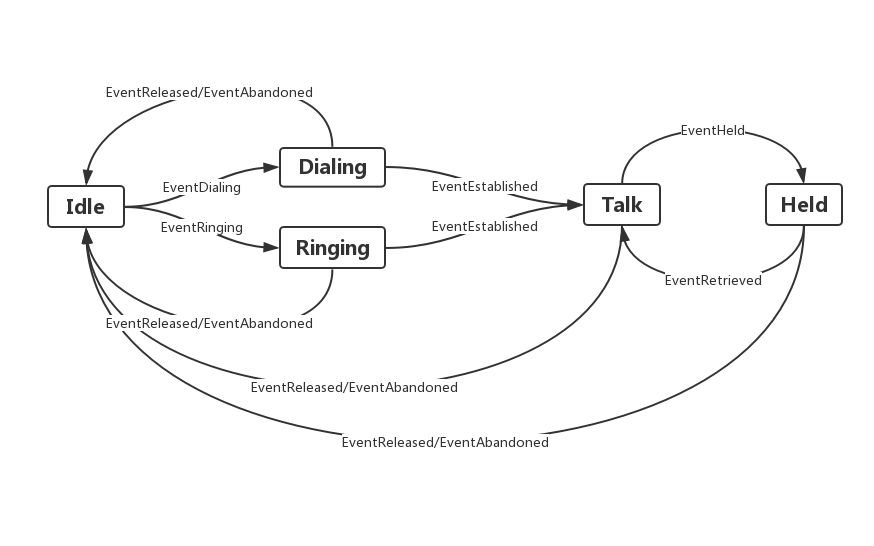


图2-2

#### makeCall（拨打电话）

【功能】

拨打电话

【函数说明】

void makeCall(dest, id, type, module, callId, queue, newTransPara, taskId)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| dest | String | 是 |  | 目标坐席工号或者外线号码 |
| id | String | 是 |  | 不确定ID的情况写-1 |
| type | String | 是 |  | 呼叫类型，参考CALLTYPE常量 |
| module | String | 否 |  | 预约回呼标识，不为空代表预约回呼 |
| callId | String | 否 |  | 原电话callId |
| queue | String | 否 |  | 任务技能组 |
| newTransPara | String | 否 |  | 透明传递参数，拨打电话时传递该参数，可在其他时间和话单中获得该数据 |
| taskId | String | 否 |  | 外呼任务ID |

【返回值】

无

#### answerCall（接听电话）

【功能】

接听电话

【函数说明】

void answerCall ()

【参数说明】

无

【返回值】

无

#### holdCall（呼叫保持）

【功能】

该方法把坐席正在通话的呼叫保持，客户听到等待音，坐席状态不变仍为通话状态，系统不会分配新的电话，坐席可以主动呼出。

【函数说明】

void holdCall ()

【参数说明】

无

【返回值】

无

#### retrieveCall（取回保持）

【功能】

坐席取回被保持的通话。

【函数说明】

void retrieveCall ()

【参数说明】

无

【返回值】

无

#### releaseCall（挂断电话）

【功能】

挂断电话

【函数说明】

void releaseCall ()

【参数说明】

无

【返回值】

无

#### singleStepTransfer（呼叫转移）

【功能】

将当前通话转移给同事或外线号码

【函数说明】

void singleStepTransfer (targetDN)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| targetDN | String | 是 |  | 目标坐席工号或者外线号码 |

【返回值】

无

#### consult（两步呼叫转移-咨询）

【功能】

保持当前通话，呼叫目标坐席工号或者外线号码

【函数说明】

void consult (targetDN)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| targetDN | String | 是 |  | 目标坐席工号或者外线号码 |

【返回值】

无

#### completeTransfer（两步呼叫转移-转出）

【功能】

两步呼叫接通后，转出当前被保持的通话给咨询对象

【函数说明】

void completeTransfer ()

【参数说明】

无

【返回值】

无

#### threeWayCall（三方通话）

【功能】

通话过程中，呼叫另外一个工号或者外线号码，使之形成三方会议形式的通话

【函数说明】

void threeWayCall (targetDN)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| targetDN | String | 是 |  | 目标坐席工号或者外线号码 |

【返回值】

无

#### releaseThreeWayCall（结束三方通话）

【功能】

通话过程中，呼叫另外一个工号或者外线号码，使之形成三方会议形式的通话

【函数说明】

void releaseThreeWayCall (callId)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| callId | String | 是 |  | 其中一个通话的呼叫id |

【返回值】

无

#### sendDTMF（二次拨号）

【功能】

通话过程中再次拨号发送DTMF指令

【函数说明】

void sendDTMF(lineId, digit)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| lineId | String | 否 |  | lineId 要挂断的线路ID，当为空时取当前默认线路 |
| digit | String | 是 |  | 发送的按键值，仅允许包含0-1和\* |

【返回值】

无

## 可定制方法

在开发过程中部分方法默认情况无法满足需求， sdk中部分方法可以覆写一些函数来达到定制效果。

### 工具库

PhoneBar中内置了一个工具库来供内部程序的运行，工具库可以通过PhoneBar.utils获得，其中部分方法实现了某种动作的简单的默认操作，开发者可以覆盖这些方法实现丰富的功能。

#### showMessage（提示消息）

【功能】

缺省插件以alert方式弹出，用户可以覆盖的弹窗方法。提示消息默认使用alert。

【函数说明】

void showMessage (msg)

【参数说明】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 必须 | 默认值 | 描述 |
| msg | String | 是 |  | 消息内容 |

【返回值】

无

【示例】

PhoneBar.**utils**.showMessage = **function** (msg) {  
 swal({  
 **title**: **'系统消息'**,  
 **text**: msg,  
 **type**: **'info'** })  
}

### 日志

PhoneBar中使用日志对象Log输出日志，可以通过PhoneBar.Log获得，默认使用浏览器的console对象打印，您可以通过重新定义Log对象或者覆写Log对象的log、info、error方法，可以重新改变日志输出方式。

【示例】

PhoneBar.**Log**.*log* = **function** (content, msgType) {  
 **var** oTextareaInfo = ***document***.getElementById(**"log"**);  
 oTextareaInfo.innerHTML += content;  
}

## 密码加密

在集成过程中座席密码需要在前端传输，为了防止在前端传输过程密码泄露，需要对密码加密后在JavaScript代码中使用。话条给开发者提供了普通加密和随机码混淆加密两种加密方式，分别应对客户使用环境可信和不可信的两种场景使用。**加密模式需要配合九五云客服软电话使用。**

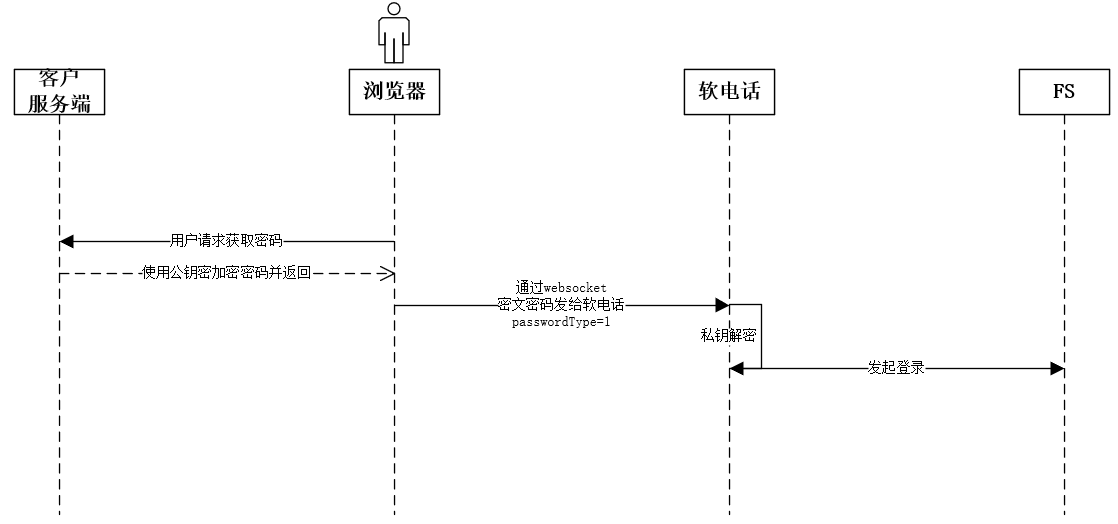
密码加密两种方式都使用非对称RAS加密算法加密，在对接阶段向开发者提供公钥和坐席账号密码，服务端根据对密码加密后返回给前端使用。

JAVA加密示例代码：

|  |
| --- |
| import org.apache.commons.codec.binary.Base64;  import javax.crypto.BadPaddingException;  import javax.crypto.Cipher;  import javax.crypto.IllegalBlockSizeException;  import javax.crypto.NoSuchPaddingException;  import java.io.ByteArrayOutputStream;  import java.io.IOException;  import java.security.InvalidKeyException;  import java.security.Key;  import java.security.KeyFactory;  import java.security.NoSuchAlgorithmException;  import java.security.interfaces.RSAPrivateKey;  import java.security.interfaces.RSAPublicKey;  import java.security.spec.PKCS8EncodedKeySpec;  import java.security.spec.X509EncodedKeySpec;  public class RSAEncrypt {  /\*\*  \* 加密算法RSA  \*/  public static final String KEY\_ALGORITHM = "RSA";  /\*\*  \* RSA最大加密明文大小  \*/  private static final int MAX\_ENCRYPT\_BLOCK = 117;  /\*\*  \* RSA最大解密密文大小  \*/  private static final int MAX\_DECRYPT\_BLOCK = 128;  /\*\*  \* 从字符串中加载公钥  \*  \* @param publicKey 公钥数据字符串  \* @throws Exception 加载公钥时产生的异常  \*/  public static RSAPublicKey loadPublicKeyByStr(String publicKey) throws Exception {  byte[] keyBytes = Base64.decodeBase64(publicKey);  KeyFactory keyFactory = KeyFactory.getInstance(KEY\_ALGORITHM);  X509EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);  return (RSAPublicKey) keyFactory.generatePublic(keySpec);  }  /\*\*  \* 从字符串中加载私钥  \*  \* @param privateKeyStr 私钥数据字符串  \* @throws Exception 加载公钥时产生的异常  \*/  public static RSAPrivateKey loadPrivateKeyByStr(String privateKeyStr) throws Exception {  byte[] buffer = Base64.decodeBase64(privateKeyStr);  PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(buffer);  KeyFactory keyFactory = KeyFactory.getInstance(KEY\_ALGORITHM);  return (RSAPrivateKey) keyFactory.generatePrivate(keySpec);  }  /\*\*  \* <p>  \* 已加密数据解密  \* </p>  \*  \* @param key 私钥 RSAPrivateKey  \* @param cipherData 已加密数据  \* @return  \* @throws NoSuchPaddingException  \* @throws NoSuchAlgorithmException  \* @throws InvalidKeyException  \* @throws BadPaddingException  \* @throws IllegalBlockSizeException  \* @throws IOException output输出流关闭失败  \*/  public static byte[] decrypt(Key key, byte[] cipherData) throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException, BadPaddingException, IllegalBlockSizeException, IOException {  Cipher cipher = Cipher.getInstance(key.getAlgorithm());  cipher.init(Cipher.DECRYPT\_MODE, key);  int inputLen = cipherData.length;  ByteArrayOutputStream out = new ByteArrayOutputStream();  int offSet = 0;  byte[] cache;  int i = 0;  // 对数据分段解密  while (inputLen - offSet > 0) {  if (inputLen - offSet > MAX\_DECRYPT\_BLOCK) {  cache = cipher.doFinal(cipherData, offSet, MAX\_DECRYPT\_BLOCK);  } else {  cache = cipher.doFinal(cipherData, offSet, inputLen - offSet);  }  out.write(cache, 0, cache.length);  i++;  offSet = i \* MAX\_DECRYPT\_BLOCK;  }  byte[] decryptedData = out.toByteArray();  out.close();  return decryptedData;  }  /\*\*  \* <p>  \* 未加密源数据加密  \* </p>  \*  \* @param key 公钥 RSAPublicKey  \* @param plainTextData 未加密源数据  \* @return  \* @throws InvalidKeyException  \* @throws NoSuchPaddingException  \* @throws NoSuchAlgorithmException  \* @throws BadPaddingException  \* @throws IllegalBlockSizeException  \* @throws IOException output输出流关闭失败  \*/  public static byte[] encrypt(Key key, byte[] plainTextData) throws InvalidKeyException, NoSuchPaddingException, NoSuchAlgorithmException, BadPaddingException, IllegalBlockSizeException, IOException {  Cipher cipher = Cipher.getInstance(key.getAlgorithm());  cipher.init(Cipher.ENCRYPT\_MODE, key);  int inputLen = plainTextData.length;  ByteArrayOutputStream out = new ByteArrayOutputStream();  int offSet = 0;  byte[] cache;  int i = 0;  // 对数据分段加密  while (inputLen - offSet > 0) {  if (inputLen - offSet > MAX\_ENCRYPT\_BLOCK) {  cache = cipher.doFinal(plainTextData, offSet, MAX\_ENCRYPT\_BLOCK);  } else {  cache = cipher.doFinal(plainTextData, offSet, inputLen - offSet);  }  out.write(cache, 0, cache.length);  i++;  offSet = i \* MAX\_ENCRYPT\_BLOCK;  }  byte[] encryptedData = out.toByteArray();  out.close();  return encryptedData;  }  public static void main(String[] args) throws Exception {  String random = "fd4d3b013aa043c0a8b26b8360111e0a";  String password = "666666";  String plainText = random+password;  String privateKey = "";  System.out.println("私钥："+ privateKey);  String publicKey = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcVffytOZfACrG238ln00sNsk4YynXE4IQ+1OmfTaM2wHMiCdmBclQzHCGqnz4cFDVBuOlHc/fCGdEmIBqY4NVjp9RM4VxEoMA66T+5Lh/hfa7iI6BzOB1NX/FApYxnS35Th/I2OKxQwowIJ0c4i4oAplkTuqt1GCuu3VMtZAVHwIDAQAB";  System.out.println("公钥："+ publicKey);  RSAPrivateKey rsaPrivateKey = RSAEncrypt.loadPrivateKeyByStr(privateKey);  RSAPublicKey rsaPublicKey = RSAEncrypt.loadPublicKeyByStr(publicKey);    // 使用公钥来加密字符串  byte[] encodedData = RSAEncrypt.encrypt(rsaPublicKey, plainText.getBytes());  System.out.println("密文："+ Base64.encodeBase64String(encodedData));  // 使用私钥解密已加密的字符串  byte[] decodedData = RSAEncrypt.decrypt(rsaPrivateKey, encodedData);  System.out.println("解密后文字: "+ new String(decodedData));  }  } |

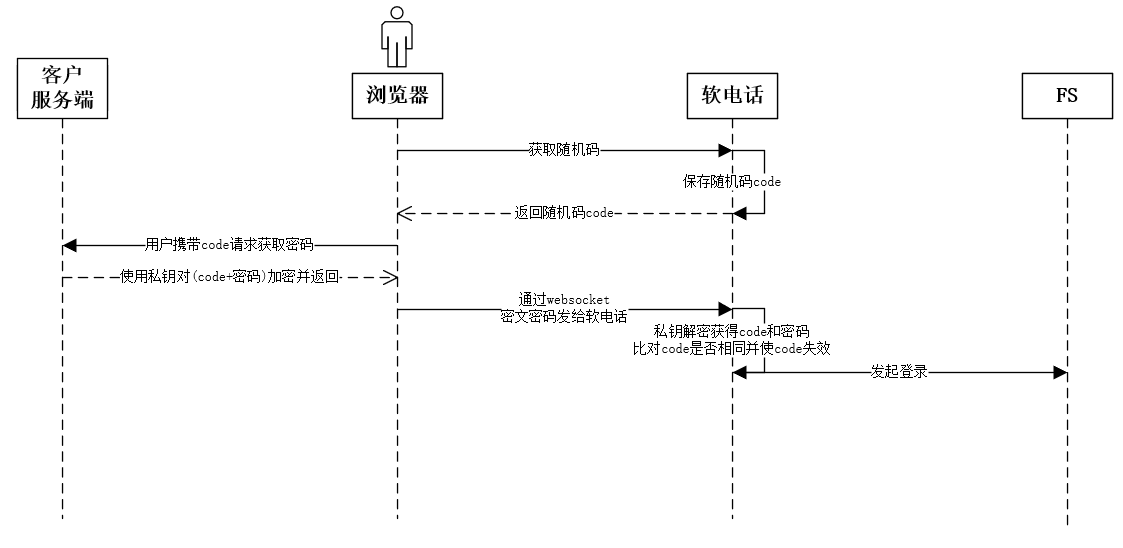
### 普通加密

简单模式适用于相对安全的可信办公环境，比如公司内部内网的情况使用。该模式使用公钥对密码加密后返回前端，话条loginType参数设置1和密文密码传输给软电话登录，优点流程简单对接开发速度快，缺点用户在浏览器中拿到加密串后，可等同于密码直接使用话条登录。

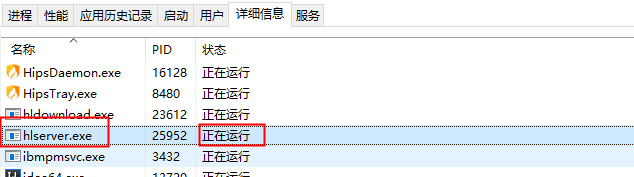


### 随机码混淆加密

随机码混淆模式适用于不可信的办公环境，比如纯外网人员较复杂的的情况使用。该模式首先需要像软电话客户端服务获取随机码code，code只可使用一次，登录成功与否都失效该code，重新登录需重新获取。获得code后后端使用公钥对cod+密码加密后返回前端，话条loginType参数设置2和密文密码传输给软电话登录，软电话客户端解析后比对随机码正确后才发起SIP登录流程。优点即使获取到了加密密码也无法登录，code在本地客户端中保存很难获取，安全级别较高，缺点相对简单模式多一步对接流程，对接开发流程相对复杂。



前端开发在JS使用GET请求http://127.0.0.1:57712/code获取随机码，如果获取失败需检查九五云客服软电话服务hlserver.exe是否启动状态。



# 九五云客服监控SDK

## 引用文件

需要引用的js和css可以通过releases下载，也可以通过npm run build:prod命令重新打包获得。

<!--引入css-->

<link rel="stylesheet" type="text/css" href="./css/AgentMonitor.css">

<!--引入js-->

<script src='./AgentMonitor.js'></script>

## 集成

### 使用九五云客服话条SDK PhoneBar集成

已经使用话条SDK集成了话条的，集成监控也非常的简单。

new AgentMonitor({

renderTo: 'agentMonitor',

phoneBar: phoneBar

});

PS: 初始化监控前必须保证话条已经初始化成功

### 使用自定义话条集成

使用自定义话条集成需要额外覆写多个方法，以获取自定义的数据

初始化坐席监控

agentMonitor = new AgentMonitor({

renderTo: 'agentMonitor',

thisDN: 'xxxxxx',

agentID: 'xxxxxx'

});

* thisDN坐席分机号，当不使用话条组件时此属性必填
* agentID坐席工号，当不使用话条组件时此属性必填

初始化以后需要设置的其他数据

* agentMonitor.setTimeOffset(offset)设置服务器与本地时间偏差，单位毫秒
* setMonitorMembers(monitorGroupList)设置监控成员，入参需要监控的监控组数组如

[{

"monitorId": "102960002",

"monitorName": "XA监控2组",

"groupMemberList": [{

"nameCn": "XA1007",

"agentId": "102961007"

}]

}, {

"monitorId": "102960003",

"monitorName": "XA监控3组",

"groupMemberList": [{

"nameCn": "XA1004",

"agentId": "102961018"

}, {

"nameCn": "XA1005",

"agentId": "102961019"

}]

}]

* updateAgentInfo(agentInfo)变更监控的坐席信息，入参类型AgentInfo

覆写以下方法

//---------------对第三方开放接口------------------------

/\*\*

\* 开始监控坐席

\* @param agentDNs

\*/

startAgentsMonitoring(agentDNs) {

}

/\*\*

\* 设置坐席状态为就绪状态

\* @param agentDN

\*/

agentReadyM(agentDN) {

}

/\*\*

\* 设置坐席状态为非就绪状态

\* @param agentDN

\* @param reason 非就绪状态原因

\*/

agentNotReadyM(agentDN, reason) {

}

/\*\*

\* 监听

\* @param callID

\* @param agentDN

\*/

monitorCall(callID, agentDN) {

}

/\*\*

\* 强插

\* @param callID

\* @param agentDN

\* @param otherDN

\*/

interruptCall(callID, agentDN, otherDN) {

}

/\*\*

\* 拦截

\* @param callID

\* @param agentDN

\* @param otherDN

\*/

substitute(callID, agentDN, otherDN) {

}

/\*\*

\* 挂断电话

\* @param callID

\* @param agentDN

\*/

releaseAgentCall(callID, agentDN) {

}