**电话条开发手册**

文档版本1.2

文件日期 2018/11/26

目录

[1 前言 3](#_Toc536638607)

[1.1 预期读者 3](#_Toc536638608)

[1.2 使用指南 3](#_Toc536638609)

[1.3 开发模式 3](#_Toc536638610)

[1.4 术语 3](#_Toc536638611)

[2 PhoneBar 4](#_Toc536638612)

[2.1 引用文件 4](#_Toc536638613)

[2.2 创建组件 5](#_Toc536638614)

[2.2.1 属性 5](#_Toc536638615)

[2.3 组件方法 6](#_Toc536638616)

[2.3.1 Destroy(销毁组件) 7](#_Toc536638617)

[2.4 事件 7](#_Toc536638618)

[2.4.1 onAgentStatusChange (坐席状态变更) 7](#_Toc536638619)

[2.4.2 onScreenPopup (弹屏事件) 7](#_Toc536638620)

[2.4.3 onRinging (振铃事件) 8](#_Toc536638621)

[2.4.4 onTalking (接通事件) 8](#_Toc536638622)

[2.4.5 onHangup(挂机事件) 8](#_Toc536638623)

[2.4.6 onLinkDisconnected (与服务器断开连接) 9](#_Toc536638624)

[2.5 坐席api 9](#_Toc536638625)

[2.5.1 坐席状态 9](#_Toc536638626)

[2.5.2 电话指令 11](#_Toc536638627)

[2.6 可定制方法 15](#_Toc536638628)

[2.6.1 工具库 15](#_Toc536638629)

[2.6.2 日志 16](#_Toc536638630)

# 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：   1. 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   1. 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

创建组件时需要配置服务器信息、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'**],  
  
 **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。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | 坐席代理服务地址 |
| 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 |  |  |  | 密码 |
| thisQueues | Array | 是 |  | 所在技能组,呼入任务可以同时多个技能组，呼出任务同时只能在一个技能组中，即数组长度必须为一。  格式如：[100018000,100018001] 或  [100019000] |
| 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

## 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.attachDatas | Object | 附加数据 |

### 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.attachDatas | Object | 附加数据 |
| callInfo | Object | 通话信息 |

### 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.attachDatas | Object | 附加数据 |
| callInfo | Object | 通话信息 |

### 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.attachDatas | Object | 附加数据 |
| callInfo | Object | 通话信息 |

### 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，那么想要拨打电话使用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）:

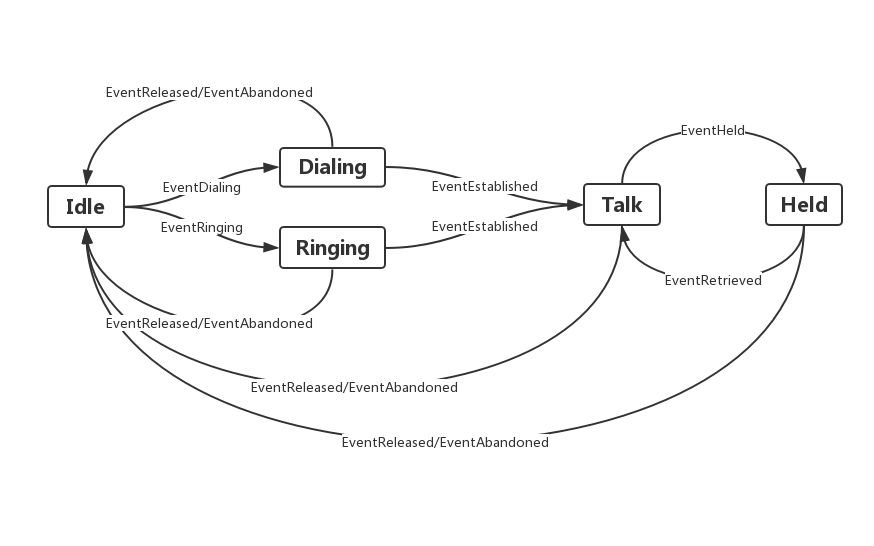


图2-2

#### makeCall（拨打电话）

【功能】

拨打电话

【函数说明】

Boolean makeCall(dest, id, type, module, call\_id, queue, newTransPara, taskId, numberId)

【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 |  |  |  | 手动回拨 |
| call\_id |  |  |  | 原电话 callId |
| queue |  |  |  |  |
| newTransPara |  |  |  |  |
| taskId |  |  |  |  |
| numberId |  |  |  |  |

【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

## 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;  
}