**H5Client开发使用说明书**

**版本：2.2.6**

**2016/05/24**

**目录**

[1.概述 3](#_Toc451515110)

[1.1. 目标 3](#_Toc451515111)

[1.2. 说明 3](#_Toc451515112)

[2. 开发流程 3](#_Toc451515113)

[2.1. 简介 3](#_Toc451515114)

[2.2. 使用 3](#_Toc451515115)

[2.3. 开发流程 3](#_Toc451515116)

[3. 开发接口 4](#_Toc451515117)

[H5Client 4](#_Toc451515118)

[Properties 4](#_Toc451515119)

[Methods 5](#_Toc451515120)

[Sessions 5](#_Toc451515121)

[Properties 6](#_Toc451515122)

[Query 7](#_Toc451515123)

[Methods 7](#_Toc451515124)

[Agent 8](#_Toc451515125)

[Properties 8](#_Toc451515126)

[Station 9](#_Toc451515127)

[Properties 9](#_Toc451515128)

[Methods 9](#_Toc451515129)

[Group 13](#_Toc451515130)

[Properties 13](#_Toc451515131)

[Methods 14](#_Toc451515132)

[Calls 15](#_Toc451515133)

[Properties 15](#_Toc451515134)

[Methods 15](#_Toc451515135)

[Call 16](#_Toc451515136)

[Properties 16](#_Toc451515137)

[Event 17](#_Toc451515138)

[Methods 18](#_Toc451515139)

[4. 事件 19](#_Toc451515140)

[5. 错误码 23](#_Toc451515141)

[6. 配置 26](#_Toc451515142)

[7. 报工号示例 27](#_Toc451515143)

[8. 满意度示例 28](#_Toc451515144)

[9. 查询排队数示例 29](#_Toc451515145)

[10. 使用示例 30](#_Toc451515146)

# 1.概述

**H5Client** SDK是在Wilcom CTI系统中连接服务端和客户端的工具，利用该控件可以实现软电话等客户端程序的基本功能。

## 目标

本文的目的是使读者可以了解SDK的接口和事件，阅读本文后能够编写基于CTI的软电话功能软件。

## 说明

H5Client基于Promise/A+规范（已加入ECMAScript 6），需要大致熟悉Promise/A+规范。

# 开发流程

## 简介

H5Client SDK基于HTML5 WebSocket API， 在CTI系统之上的一个通信工具，用户可以调用SDK接口，发送请求给CTI服务器，并且通过SDK的事件接口获取到服务器的返回和事件通知。

SDK需要支持HTML5 WebSocket的现代浏览器运行。

兼容情况

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chrome | Firefox | IE | Safari | Opera |
| 4+ | 4+ | 10+ | 5.0.2+ | 11+ |

## 使用

在页面script标签导入promise.js和H5Client.js文件。

## 开发流程

用户进行一般性软电话开发时，首先需要创建**Session**对象（用于连接、关闭、管理Station对象等），从Session对象中获取**Event**对象（用于监听、触发事件），然后创建**Station**对象（需要传入session对象，用于呼叫控制、登录、就绪等），从Station对象中获取**Calls**对象（用于获取呼叫数据）。

调用Session的start接口建立和CTI服务器之间的连接，然后调用其他接口与CTI服务器进行通信。电话功能需要首先调用Station的signIn接口。产生的数据可以直接访问相应对象的属性获得。

# 开发接口

## H5Client

SDK全局对象，所有对象的入口

### Properties

#### settings : Object

全局设置

#### Error : Object

错误对象

#### onerror : Function

所有错误的回调函数

#### Session : Object

会话对象

#### Event : Object

事件对象（由会话对象产生）

#### Query : Object

查询器对象（由会话对象产生）

#### Station : Object

分机对象（依赖于会话对象）

#### Agent : Object

坐席对象（由分机对象产生）

#### Call : Object

呼叫对象

#### Calls : Object

呼叫集合（由分机对象产生）

### Methods

#### **openLog**(addr)

打开SDK日志

**Parameters**

addr : String

日志服务器地址

## Sessions

会话对象

### Properties

**静态属性**

**DEAD** 0

**ALIVE** 1

**CONNECTING** 2

**RECOVERING** 3

#### **State** : Number

会话状态，静态属性之一

#### [event](#_Event) : Object

事件对象

#### **heartbeatInterval** : Number

心跳间隔时间，单位毫秒，设置值将会覆盖全局设置

#### **stations** : Array

station对象列表

#### **start**(dcmp,[connector])

打连接并初始化Session

**Parameters**

dcmp : String

dcmp服务器根目录地址

connector : String

可选，connector服务名，默认为’DefaultConnector’

#### **end**()

关闭Session

## Query

查询器对象

### Methods

#### **queryAgentByLoginName**(loginName)

根据登录名查询坐席信息

**Parameters**

loginName : String

登录名

#### **queryStationByDeviceId**(deviceId)

根据分机号查询分机信息

**Parameters**

deviceId : String

分机号

#### **queryStationByAgentId**(agentId)

根据工号查询分机信息

**Parameters**

agentId : String

工号

#### **queryReasonCode**()

查询原因码

#### **queryAllQueues**(option)

查询技能组

**Parameters**

options **:** Object

可选附加参数

realtimeStat: String

实时统计服务名称

## Agent

坐席对象

### Properties

**静态属性**

**PENDING ''** 初始，未获取到状态

**LOGIN 'Login'** 登录

**LOGOUT 'Logout'** 退出

**NOT\_READY 'NotReady'**  未就绪

**READY 'Ready'**  就绪

**WORK\_NOT\_READY 'WorkNotReady'** 后处理

**Agent.WORK\_READY 'WorkReady'**

#### **loginName** : String

登录名

#### **agentId** : String

坐席ID

#### **loginMode** : String

默认登录后状态

**Ready** 就绪

**NotReady** 未就绪

#### **mode** : String

坐席状态，为静态属性之一

#### **queues** : Array

坐席所属技能组列表

## Station

分机对象

### Properties

**静态属性**

**PENDING** 0初始，无状态

**MONITORED 1** 监控中

**CANCELLED 2** 监控被取消

#### **state** : String

分机状态

为静态属性之一

#### **deviceId** : String

分机号

#### [calls](#_Calls): String

呼叫对象列表

### Methods

#### **signIn**(deviceId, agentId, password, [options])

签入

**Parameters**

deviceId : String

分机号

agentId : String

工号

password : String

密码

#### **signOut**( [options])

签出

#### **setMode**( mode,[options])

设置登录后的状态

**Parameters**

mode : String

状态

#### **setState** (deviceId, agentId, password, mode, [options])

设置坐席状态

**Parameters**

deviceId : String

分机号

agentId : String

工号

password : String

密码

password : String

mode : String

状态

#### **setAuto** (agentId)

设置自动后处理模式

#### **sync**(deviceId)

更新station数据

#### **monitor**(deviceId)

监视分机

#### **stopMonitor**()

停止监视分机

#### **dial**(dest, [options])

外拨

**Parameters**

dest : String

拨打的号码

#### **answer**([options])

应答

#### **drop**([options])

挂断

#### **hold**([options])

保持

#### **retrieve**([options])

取回

#### **consultation**(dest, [options])

磋商

**Parameters**

dest : String

磋商的号码

options **:** Object

可选附加参数

userData : Object

随路数据

#### **conference**([options])

会议

#### **transfer**([options])

转移

#### **singleStepConference**(dest, [options])

单步会议

**Parameters**

dest : String

单步会议目标号码

options **:** Object

可选附加参数

reverse: Boolean

是否反转，可取值

false 加入第三方，默认模式

true 插入已存在的通话

type: String

会议类型，可取值：

Active 普通会议模式，默认加入类型

Silent 静默加入，监听模式

#### **singleStepTransfer**(dest, [options])

单步转移

**Parameters**

dest : String

单步转移目标号码

options **:** Object

可选附加参数

userData : Object

随路数据

## Group

技能组对象

### Properties

#### **state** : Number

技能组状态

同分机[Station](#_Station)状态

#### **deviceId** : String

技能组号

#### **availableAgents** : Number

可用坐席数

#### **loggedInAgents**: Number

登录坐席数

#### **otherAgents**: Number

其他坐席数

#### **onCallInAgents**: Number

呼入通话坐席数

#### **onCallOutAgents**: Number

呼出通话坐席数

#### **allAgents**: Array

所有坐席

#### **workNotReadyAgents**: Number

话后处理坐席数

#### **notReadyAgents**: Number

离席坐席数

### Methods

#### **monitor**(deviceId, [options])

监控技能组

**Parameters**

deviceId : String

技能组号

#### **stopMonitor**( [options])

取消监控技能组

## Calls

呼叫对象列表，继承自数组，所有数组方法可用

### Properties

#### **activeCall** : String

当前激活的呼叫ID

#### **consultationFrom** : String

磋商中被保持的呼叫ID

#### **consultationTo** : String

磋商的呼叫ID

#### **consultationType** : String

自定义磋商类型

### Methods

#### **get**(callId)

根据呼叫ID获得呼叫对象

**Parameters**

callId : String

呼叫ID

**Returns**

Object

[Call](#_Call)对象

#### **filter**(call)

过滤呼叫

**Parameters**

call : Object

Call对象

**Returns**

Object

[Calls](#_Calls)对象中所有[Call](#_Call)对象中包含指定对象中属性和值的新[Calls](#_Calls)对象

## Call

呼叫对象

### Properties

#### **callId** : String

呼叫ID，短位

#### **contactId** : String

呼叫ID，长位

#### **state** : String

呼叫状态

#### **createTime** : Number

开始时间

#### **answerTime** : Number

应答时间

#### **ani** : String

主叫号码

#### **dnis** : String

被叫号码

#### **direction** : String

呼叫方向

**In** 来电

**Out** 外呼

#### **isHeld** : Boolean

是否处于保持状态

## Event

事件管理

### Methods

#### **on**(strtypes, fn)

监听事件

**Parameters**

strtypes : String

事件类型，支持多事件（”,“隔开），命名空间（”.“隔开，用于移除事件），如：'session.Open, session.Close

fn : Function

监听函数

#### **one**(strtypes, fn)

监听事件，只触发一次

**Parameters**

strtypes : String

事件类型

fn : Function

监听函数

#### **off**(strtypes, [fn])

取消监听

**Parameters**

strtypes : String

事件类型，支持多事件（”,“隔开），命名空间（”.“隔开，用于移除事件），支持通配符“\*”，如：'session.\*'，匹配如'session.Opened'；'user.\*.Opened'，匹配如'user.agent.Opened'；'Opened'

fn : Function

可选监听函数，省略将移除事件下的所有监听函数

#### **trigger**(strtypes, data)

触发事件

**Parameters**

strtypes : String

事件名，触发多个事件用“,”隔开，如: Opened, Closed

data : Object

触发事件时传递的数据对象

# 事件

呼叫相关事件常用属性

srcDeviceId : String

订阅事件的分机

callId : String

呼叫标识

contactId : String

呼叫唯一标识(长位)

callingDeviceId : String

主叫

calledDeviceId : String

被叫

ociCallingDeviceId : String

原始主叫

ociCalledDeviceId : String

原始被叫

split : String

技能组

trunkGroup : String

中继组号

trunkMember : String

中继通道号

#### **Disconnected**

连接断开事件

#### **Connecting**

重连事件，每次尝试重连都会触发

**Parameters**

e : Object

事件对象，包含如下属性

reconnectAttempts : Number

重连次数

#### **Reconnected**

连接恢复事件

#### Recovered

重连后恢复状态恢复事件

**Parameters**

errors : Array

恢复分机和坐席状态过程中出现的错误对象数组，没有错误数组为空

#### AgentStateChange

坐席状态改变事件

#### StationStateChange

分机状态改变事件

#### **AgentLoggedOn**

坐席登录事件

#### **AgentNotReady**

坐席未就绪事件

#### **AgentReady**

坐席就绪事件

#### **AgentWorkingAfterCall**

坐席后处理事件

#### **AgentLoggedOff**

坐席退出事件

#### **ServiceInitiated**

摘机事件

#### **Originated**

外呼事件

#### **Delivered**

振铃事件

**Parameters**

alertingDevice : String

振铃的话机

#### **Established**

接通事件

**Parameters**

answeringDevice : String

应答的话机

#### **ConnectionCleared**

挂断事件

**Parameters**

releasingDevice : String

挂断的话机

#### **Held**

保持事件

**Parameters**

holdingDevice : String

保持的话机

#### **Retrieved**

恢复事件

**Parameters**

retrievingDevice : String

取回的话机

#### **Conferenced**

会议事件

**Parameters**

conferencingDevice : String

会议发起的话机

#### **Transferred**

转移事件

**Parameters**

transferringDevice : String

转移发起的话机

#### **Diverted**

代接事件

**Parameters**

divertingDevice : String

被代接的话机

#### **MonitorCancelled**

监控丢失

#### **MonitorRecovered**

监控恢复

#### **Snapshot**

技能组快照事件

**Parameters**

callsInQueue: Number

当前排队数

#### **failed**

呼叫失败事件

# 错误码

**[Error](#_说明) 对象包含如下属性**

code错误码

name错误名称

message错误信息

type错误类型

400 INVOKE\_TIMEOUT 请求超时

401 SOCKET\_NOT\_OPEN 连接未建立

402 SESSION\_NOT\_ALIVE 会话未建立

403 STATION\_NOT\_MONITORED 分机未监视

404 DCMP\_CONNECTION\_FAILED DCMP连接失败

405 SERVER\_CONNECTION\_FAILED 服务器连接失败

406 NO\_AVAILABLE\_SERVER 没有可用的服务器

407 INVALID\_STATE\_ERR 无效的状态

410 NO\_DEVICE\_INFO 未获取到分机信息

411 EXIST\_AGENT\_LOGIN 工号已登录

412 EXIST\_DEVICE\_LOGIN 分机已登录

413 NOT\_LOGGED\_IN 未登录

414 INVALID\_DEVICE\_ID 无效的分机号

415 INVALID\_AGENT\_ID 无效的工号

416 INVALID\_PASSWORD 无效的密码

426 UNKNOW\_ERROR 未知错误

499 Error 其他错误

33 RESOURCE\_BUSY 分机正在通话

GENERIC\_UNSPECIFIED 29 坐席已登录（登录）或未登录（退出）

GENERIC\_OPERATION 28 分机已被登录，或者分机号码未被注册（ONE-X或实体话机）

REQUEST\_INCOMPATIBLE\_WITH\_OBJECT 2 监控的设备在交换机中存在异常，可能原因ACD License过期

VALUE\_OUT\_OF\_RANGE 3 值超过范围

OBJECT\_NOT\_KNOWN 4 在单步会议中callid参数没有包含当前呼叫id

INVALID\_CALLING\_DEVICE 5 无效的主叫号码

INVALID\_CALLED\_DEVICE 6 无效的被叫号码

INVALID\_CSTA\_CALL\_IDENTIFIER 11 无效的呼叫标示

GENERIC\_STATE\_INCOMPATIBILITY 21 状态不兼容

INVALID\_OBJECT\_STATE 22 无效的对象状态

INVALID\_CONNECTION\_ID\_FOR\_ACTIVE\_CALL 23

NO\_ACTIVE\_CALL 24 当前没有可用的呼叫

NO\_HELD\_CALL 25 当前没有保持的呼叫

RESOURCE\_BUSY 33 资源忙，无效的分机状态，比如登录时分机可能正在通话

CONFERENCE\_MEMBER\_LIMIT\_EXCEEDED 38 会议成员超过限制

|  |
| --- |
| // 错误定义  Error.INVOKE\_TIMEOUT = {  code: 400,  name: 'INVOKE\_TIMEOUT',  message: '请求超时 - {method}',  type: 'client',  };  Error.SOCKET\_NOT\_OPEN = {  code: 401,  name: 'SOCKET\_NOT\_OPEN',  message: '连接未建立',  type: 'client'  };  Error.SESSION\_NOT\_ALIVE = {  code: 402,  name: 'SESSION\_NOT\_ALIVE',  message: '会话未建立',  type: 'client'  };  Error.STATION\_NOT\_MONITORED = {  code: 403,  name: 'STATION\_NOT\_MONITORED',  message: '分机未监视',  type: 'connector',  };  Error.DCMP\_CONNECTION\_FAILED = {  code: 404,  name: 'DCMP\_CONNECTION\_FAILED',  message: 'DCMP连接失败：{dcmp}',  type: 'connector',  };  Error.SERVER\_CONNECTION\_FAILED = {  code: 405,  name: 'SERVER\_CONNECTION\_FAILED',  message: '服务器连接失败：{server}',  type: 'client',  };  Error.NO\_AVAILABLE\_SERVER = {  code: 406,  name: 'NO\_AVAILABLE\_SERVER',  message: '没有可用的服务器',  type: 'client',  };  Error.INVALID\_STATE\_ERR = {  code:407,  name:'INVALID\_STATE\_ERR',  message: '无效的状态',  type:'client'  };  Error.NO\_DEVICE\_INFO = {  code: 410,  name: 'NO\_DEVICE\_INFO',  message: '未获取到分机信息',  type: 'client'  };  Error.UNKNOW\_ERROR = {  code: 426,  name: 'UNKNOW\_ERROR',  message: '未知错误',  type: 'connector',  };  Error.EXIST\_AGENT\_LOGIN = {  code: 411,  name: 'EXIST\_AGENT\_LOGIN',  message: '工号已登录，登录分机：{deviceId}',  type: 'connector',  };  Error.EXIST\_DEVICE\_LOGIN = {  code: 412,  name: 'EXIST\_DEVICE\_LOGIN',  message: '分机已登录，登录工号：{agentId}',  type: 'connector',  };  Error.NOT\_LOGGED\_IN = {  code: 413,  name: 'NOT\_LOGGED\_IN',  message: '未登录',  type: 'connector',  };  Error.INVALID\_DEVICE\_ID = {  code: 414,  name: 'INVALID\_DEVICE\_ID',  message: '无效的分机号：{deviceId}',  type: 'connector'  };  Error.INVALID\_AGENT\_ID = {  code: 415,  name: 'INVALID\_AGENT\_ID',  message: '无效的工号：{agentId}',  type: 'connector'  };  Error.INVALID\_PASSWORD = {  code: 416,  name: 'INVALID\_PASSWORD',  message: '无效的密码',  type: '{type}'  };  Error.RESOURCE\_BUSY = {  code: 33,  name: 'RESOURCE\_BUSY',  message: '分机正在通话',  type: '{type}'  }; |

# 配置

|  |
| --- |
| // 全局设置  **var** settings = {  // 超时时间  timeout:6000,  // 心跳间隔  heartbeatInterval:5000,  // 最大重连间隔时间  maxReconnectInterval:30000,  // 初始重连间隔时间  reconnectInterval:1000,  // 重连间隔时间衰减  reconnectDecay:1.5,  // 暂时无效  timeoutInterval:2000,  // 最大重试次数，null为无限重试  maxReconnectAttempts:**null**,  // 默认坐席登录状态  loginMode:'NotReady',  // 坐席工作模式  autoWorkMode:'ManualIn',//AutoIn ManualIn  // DCMP服务器地址，无需修改  dcmpAddr:'http://{0}/action/Dcmp.queryService/{1}',  // Connect服务器地址，无需修改  serverAddr:'ws://{0}/websocket',  // 日志服务器地址  logAddr:'ws://localhost:8080',  // 非多会话，刷新可以重连，通话也可以登录  multipleSession: **false**,  // 实时统计服务名称（查询技能组用到）  realtimeStat: 'RTStat',  debug:**false**  }; |

# 报工号示例

1、接通客户电话时，磋商IVR号码

2、IVR振铃，发起三方会议

        程序实现：订阅接通事件(Established)，判断主叫号码是否为客户号码，调用consultation接口，成功后调用conference接口

|  |
| --- |
| // 报工号  event.on('user.Established', **function**(e) {  // 判断号码  **if** (playAgentNo && e && e.srcDeviceId && e.answeringDevice && e.srcDeviceId == station.deviceId && e.answeringDevice == station.deviceId && e.callingDevice && e.callingDevice.length > 6) {  // 外线->调用consultation接口，参数填IVR号码  station.consultation(ivrNo, {  userData: {  map: {  agentid: agent.agentId,  'function': 'playAgentNO'  }  }  }).then(**function**(e) {  console.info("磋商IVR成功");  **return** station.conference();  }).then(**function**(e) {  console.info("加入IVR成功");  })['catch'](**function**(e) {  // 错误处理  widget.error(e.message);  });  }  }); |

# 满意度示例

转满意度流程：

       准备结束与客户通话时，坐席点击转满意度的按钮

       程序实现：订阅按钮点击事件，调用singleStepTransfer接口

|  |
| --- |
| // 满意度  $('#degree').on('click', **function**() {  station.singleStepTransfer(degree, {  // 随路数据  userData: {  map: {  VDN: degree,  transfer: "3"  }  }  }).then(**function**(e) {  console.info('满意度');  })['catch'](**function**(e) {  checkButton();  widget.error(e.message);  });  }); |

# 查询排队数示例

查询排队数步骤：

       监听技能组Snapshot事件

       登录成功后取得坐席所属技能组

       订阅技能组快照事件

|  |
| --- |
| // 监听组状态改变事件  event.on('user.Snapshot', **function**(e) {  **if** (e.srcDeviceId != group.deviceId) **return**;  $('#callsInQueue').html(group.callsInQueue);  });  // 点击签入按钮先连接后登录  $('#login').on('click', **function**() {  // 调用session的start接口连接服务器  session.start(dcmpAddr, connector).then(**function**() {  // 调用agent的signIn接口登录  **return** station.signIn($('#deviceId').val(), $('#agentId').val(), $('#password').val());  // 登录成功的处理函数  }).then(**function**(e) {  checkButton();  widget.alert('签入成功');  **if** (callsWaiting && group.state == Station.PENDING && agent.queues[0]) {  // 订阅技能组快照事件，如果单技能组，取第一个  group.monitor(agent.queues[0].queueId);  }  }  // 登录失败的处理函数  })['catch'](**function**(e) {  session.end();  checkButton();  widget.error(e.message);  });  }); |

# 使用示例

|  |
| --- |
| **var** Error = H5Client.Error;  // 自定义错误消息，可覆盖  Error.GENERIC\_OPERATION = {  code: 498,// 自定义未使用的错误号  name: 'GENERIC\_OPERATION',// 异常名称等于错误属性名称  message: '操作错误，可能分机已被登录，或者分机号码未被注册（ONE-X或实体话机）',// 自定义错误消息  type: 'user' // 自定义错误类型  };  // 获得会话  **var** Session = H5Client.Session;  **var** session = **new** Session();  // 获得分机对象  **var** Station = H5Client.Station;  **var** station = **new** Station(session);    **var** Agent = H5Client.Agent;  // 修改全局配置  H5Client.settings.timeout = 6000;  H5Client.settings.heartbeatInterval = 60000;  // 连接服务器，开始会话  $('#open').on('click',**function**(){  session.start($('#server').val())  .then(**function**(e){  console.info('连接成功');  })['catch'](**function**(e){  console.warn(e.message);  });  });  // 关闭会话  $('#close').on('click',**function**(){  session.end()  .then(**function**(e){  console.info('关闭成功');  })['catch'](**function**(e){  console.warn(e.message);  });  });  // 登录  $('#signIn').on('click',**function**(){  station.signIn($('#deviceId').val(),$('#agentId').val(),$('#password').val())  .then(**function**(e){  console.info('签入成功');  })['catch'](**function**(e){  console.warn(e.message);  });  });  // 退出  $('#signOut').on('click',**function**(){  station.signOut()  .then(**function**(e){  console.info('签出成功');  })['catch'](**function**(e){  console.warn(e.message);  });  }); |