

CEC  
3.6.0.0

# 座席集成——座席呼叫处理 (RESTful)

文档版本 01  
发布日期 2024-08-19



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <https://www.huawei.com>

客户服务邮箱： [support@huawei.com](mailto:support@huawei.com)

客户服务电话： 4008302118

# 安全声明

## 漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：

<https://www.huawei.com/cn/psirt/vul-response-process>

如企业客户须获取漏洞信息，请参见如下网址：

<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

# 目录

<b>1 简介</b>	<b>1</b>
1.1 概述	1
1.2 组网和开发模式	1
1.3 对接模式	2
1.3.1 座席事件获取方式	2
1.3.2 C4 座席操作类接口鉴权方式	3
1.3.3 座席连接超时检测	4
1.3.4 座席事件推送方式	4
<b>2 第一次应用开发</b>	<b>5</b>
2.1 实例简介	5
2.2 前提条件	8
2.3 实现过程	8
2.4 实现示例	10
2.5 返回结果	21
<b>3 基本应用开发</b>	<b>22</b>
3.1 如何查看接口调用日志	22
3.2 基本功能与设置	22
3.2.1 座席签入	22
3.2.2 查询座席的配置技能（可选）	23
3.2.3 签入座席的技能	23
3.2.4 获取座席事件	23
3.2.5 座席签出	23
3.3 基本语音呼叫	23
3.3.1 呼出	23
3.3.2 呼入	25
3.3.2.1 自动应答	25
3.3.2.2 手工应答	26
3.3.2.3 查询应答来话前呼叫信息	27
3.4 通话建立后	27
3.4.1 释放呼叫	27
3.4.2 保持	28
3.4.2.1 保持呼叫	28

3.4.2.2 取保持呼叫.....	29
3.4.3 转移.....	29
3.4.4 内部求助.....	30
3.4.5 三方通话.....	32
3.4.6 静音.....	33
3.4.6.1 开始静音.....	33
3.4.6.2 停止静音.....	33
3.5 实时质检.....	33
3.5.1 插入.....	33
3.5.2 侦听.....	34
3.5.3 拦截.....	35
3.6 放音.....	36
3.6.1 在非通话状态进行放音操作.....	36
3.6.2 在通话态进行放音操作.....	38
3.7 基本多媒体呼叫.....	39
3.7.1 收发消息.....	39

# 1 简介

- 1.1 概述
- 1.2 组网和开发模式
- 1.3 对接模式

## 1.1 概述

CC-Gateway提供基于REST风格的Web Service接口，供业务封装使用，提供给二次开发人员对座席业务功能进行开发。本文档描述的是基于REST（Representational State Transfer，表象状态转换）风格提供的Web服务。文档中所提供的接口，能够使您开发提供给座席话务员使用的音视频电话处理、文字类呼叫处理等业务能力。同时您也可以通过申请其它各类开发包，提供各类呼叫中心业务的集成能力。

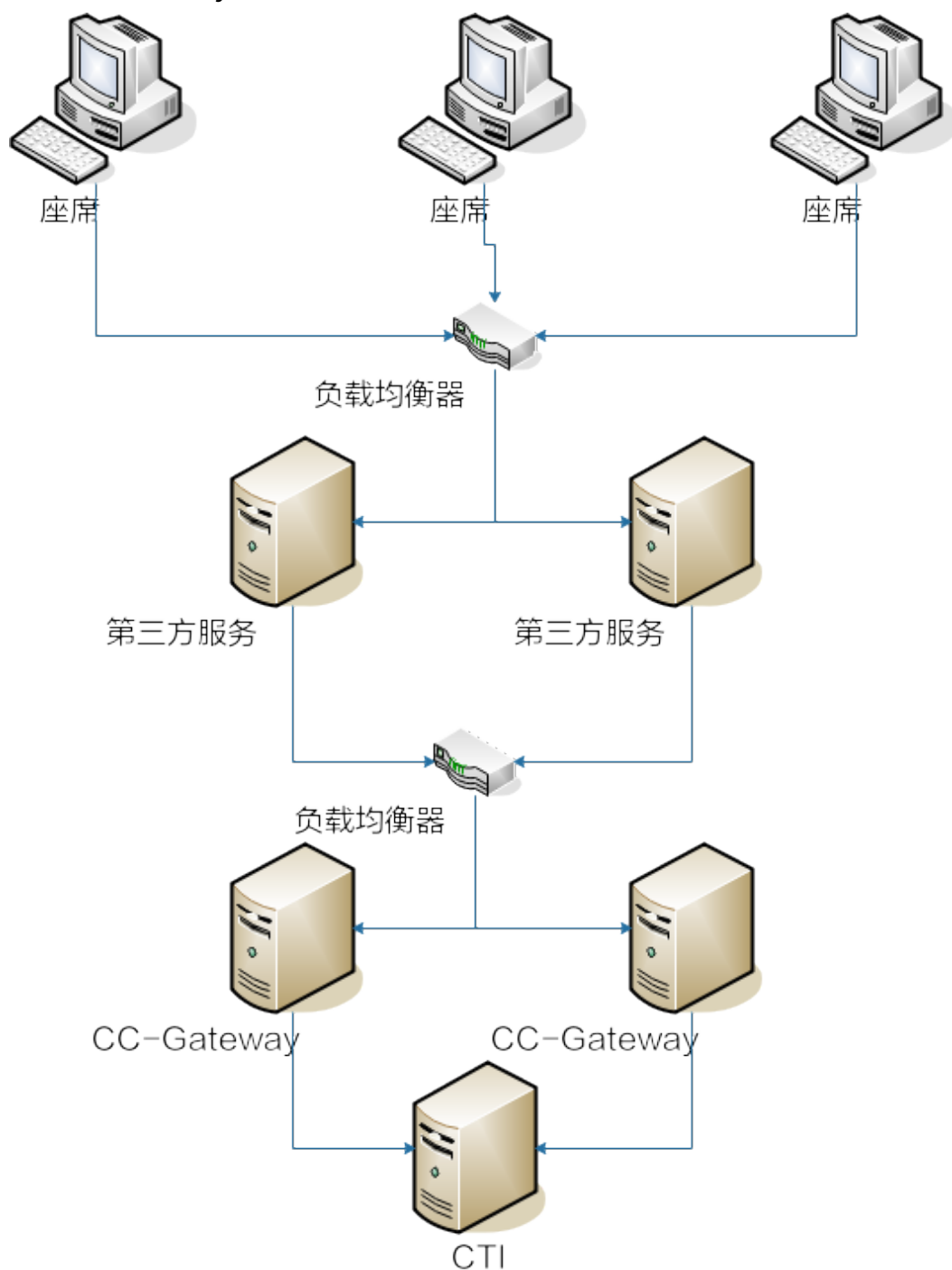
CC-Gateway接口能够提供如下功能：

- 更简易方便的电话呼叫处理，能够使您降低研发成本。
- 支持座席业务软件直接访问，您可以直接在座席话务员的电脑上直接访问接口。

## 1.2 组网和开发模式

CC-Gateway采用服务端对服务端的对接模式，如图1-1所示。

图 1-1 CC-Gateway 组网



## 1.3 对接模式

### 1.3.1 座席事件获取方式

座席事件获取方式有：轮询方式和推送方式

两种获取座席事件方式是根据座席签入接口中callBackUri和serviceToken这两个参数来区分的：

当座席签入时同时传递callBackUri和serviceToken这两个参数，说明选择事件推送方式；

只传其中一个参数或者两个参数都不传递，说明选择事件轮询方式。

请参考接口[签入](#)

事件轮询方式：

请参考接口[轮询方式获取单座席事件](#)。

事件推送方式：

请参考接口[座席心跳接口](#) 和 [推送事件回调接口](#)

### 1.3.2 C4 座席操作类接口鉴权方式

接口鉴权模式支持：静态鉴权和动态鉴权，默认使用动态鉴权模式。

#### 须知

- 静态鉴权模式只在历史局点升级兼容的场景下使用，新建局点应该使用动态鉴权模式。
- 使用静态鉴权模式时，存在Guid被猜测窃取后，仿冒使用的可能，安全性较低，应谨慎使用，历史局点应尽快升级第三方系统，使用动态鉴权模式。

#### 静态鉴权

- 在agentgateway/WEB-INF/config/basic.properties中修改配置：  
AUTHMODE = 1
- 座席调用登录接口登录成功后，可以从请求的响应头中获取到名称为Set-GUID的header，值例如为“JSESSIONID=27\*\*\*\*\*f5.AgentGateway0”，其中27\*\*\*\*\*f5.AgentGateway0为guid，这个guid就是这个座席的鉴权信息。座席调用其他接口时需要将该guid设置到http请求的header中，header的名称必须为guid。CC-Gateway会从请求的header中获取guid进行比较，判断是否合法用户，如果不是合法用户，则返回结果码为100-006或000-003。

#### 动态鉴权

动态鉴权是对静态鉴权功能的增强。guid会定时更新。

- 在agentgateway/WEB-INF/config/basic.properties中修改配置：  
AUTHMODE = 2
- 当guid改变时，从事件获取接口的HTTP请求的响应头中获取到名称为Set-GUID的header，值例如为“JSESSIONID=27\*\*\*\*\*f5.AgentGateway0”，其中27\*\*\*\*\*f5.AgentGateway0为guid，这个guid就是这个座席的鉴权信息。
- 相比静态鉴权basic.properties中新增以下两项配置  
GUIDINTERVAL = 60000 //guid更新周期  
GUIDTIMEOUT = 120000 //guid超时时间  
单位都为毫秒，并且GUIDINTERVAL必须小于GUIDTIMEOUT。



### 1.3.3 座席连接超时检测

在agentgateway/WEB-INF/config/basic.properties中修改配置:

```
TIMEOUT_FLAG = ON //是否启用检测座席连接超时  
MAXTIME = 120000 //事件轮询方式时的超时时间  
EVENT_PUSH_HANDSHAKE_MAXTIME = 120000 //事件推送方式时的超时时间
```

座席超过超时时间没有发送请求到服务端, 则座席会被强制签出。

### 1.3.4 座席事件推送方式

推送回调地址采用https方式时, 默认采用证书认证, 需要需要上传第三方提供的事件推送证书。

## 虚拟化 CC-Gateway

**步骤1** 将事件推送三方证书上传至/home/elpis/tomcat/keystore/3rdcert中。

```
chmod 600 /home/elpis/tomcat/keystore/3rdcert/*  
chown elpis:elpis /home/elpis/tomcat/keystore/3rdcert/*
```

**步骤2** 检查事件推送三方证书相关配置项, 打开配置文件/home/elpis/tomcat/webapps/agentgateway/WEB-INF/config/basic.properties, 查看以下配置项是否正确。

```
eventpush.ssl.cert.file =/tomcat/keystore/3rdcert //事件推送模式信任证书目录,只能配置用户home目录的相对路径  
eventpush.ssl.cert.type= //证书类型,示例: JKS
```

**步骤3** 将事件推送三方的证书密码, 使用DigitalWare OpenAS加密工具的AES128CBC加密算法进行加密, 得到加密后的证书密码和密钥。

**步骤4** 打开配置文件/home/elpis/tomcat/webapps/agentgateway/WEB-INF/config/servicecloud.security.properties, 填入加密后的证书密码和密钥。

```
#加密后的证书密钥  
eventpush.ssl.keyStorePwd.encryptedKey=***  
#加密后的证书密码  
eventpush.ssl.keyStorePwd=***
```

**步骤5** 保存并退出。

**步骤6** 重启CC-Gateway。

```
su - elpis  
cd /home/elpis/icdcomm/bin/  
stopcomm  
icdcomm  
cd /home/elpis/tomcat/bin/  
./shutdown.sh ;./startup.sh
```

----结束

# 2 第一次应用开发

---

- [2.1 实例简介](#)
- [2.2 前提条件](#)
- [2.3 实现过程](#)
- [2.4 实现示例](#)
- [2.5 返回结果](#)

## 2.1 实例简介

当前实例只涉及座席的登录、强制登录、设置技能队里、获取事件四个接口。  
业务逻辑如下：

图 2-1 座席事件轮询方式

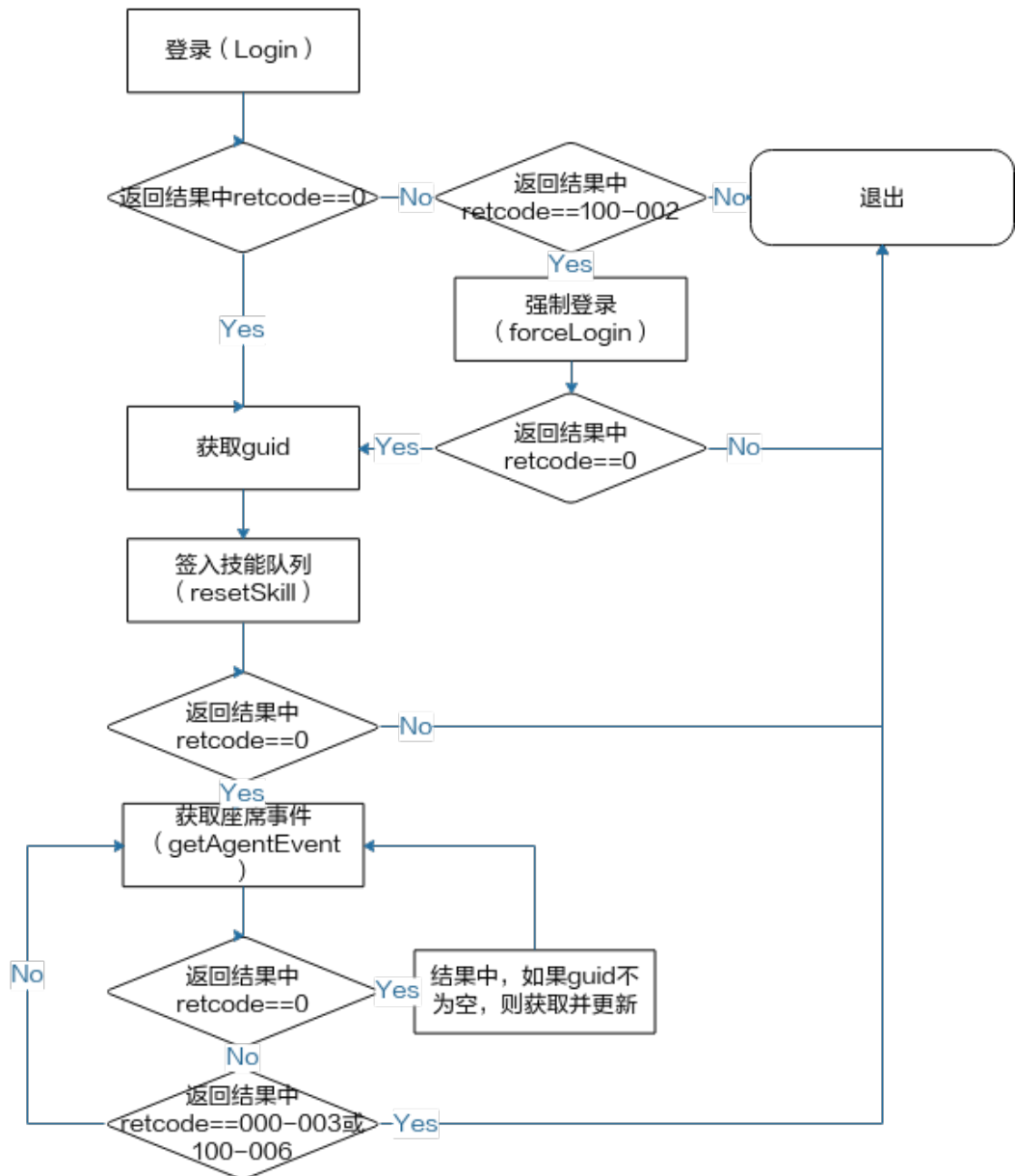
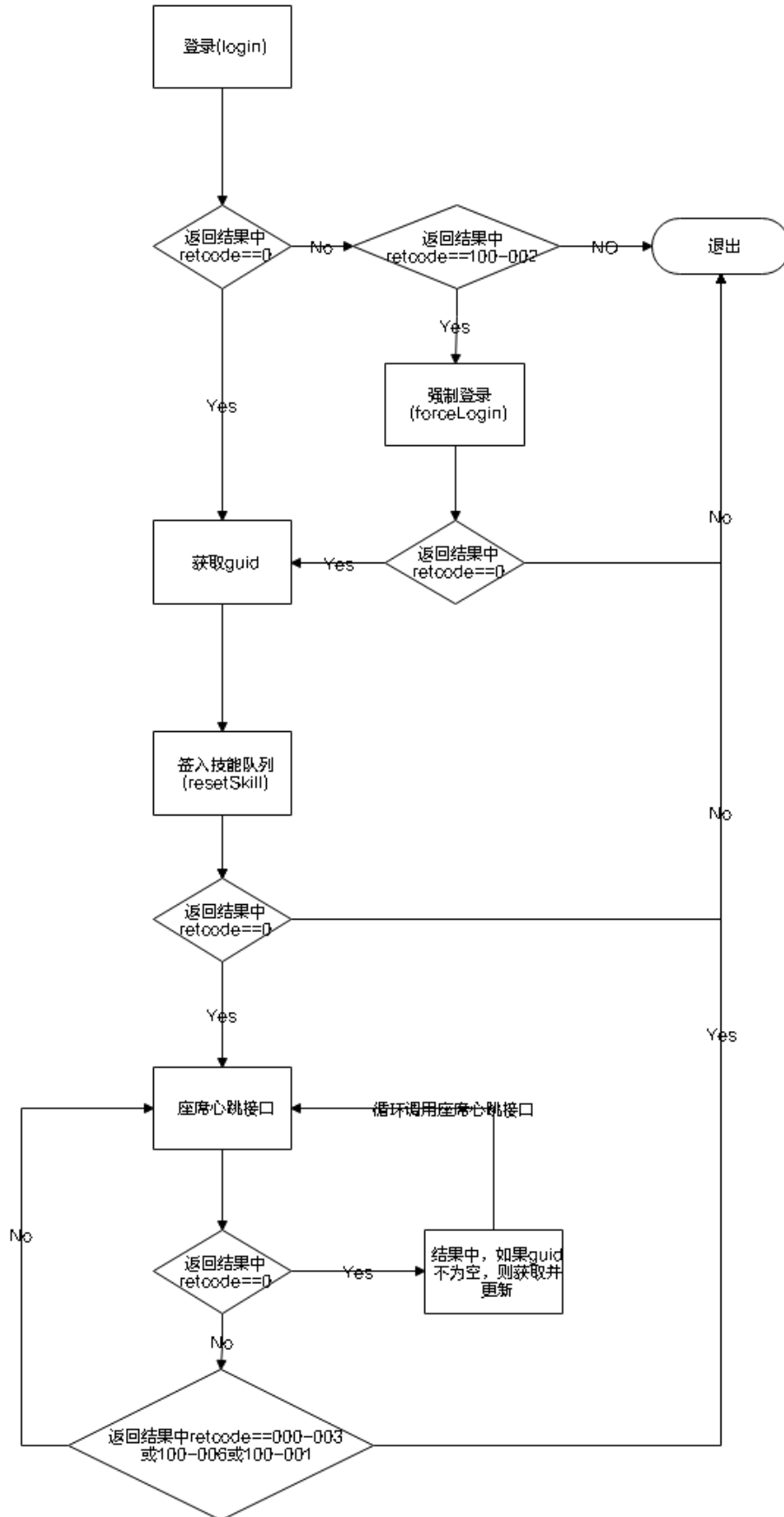


图 2-2 座席事件推送方式



1. 调用座席登录接口(login)进行登录, 如果返回结果中retcode为0, 则表示登录成功; 如果返回结果中retcode为100-002, 则表示当前座席已经登录, 可以调用强制登录接口进行登录 ( forceLogin ), 同样当返回结果中retcode为0, 表示登录成功。
2. 在调用登录或强制登录接口登录成功时, CC-Gateway会在HTTP请求的响应response中的headers中新增返回一个Set-GUID的header, 值例如为“JSESSIONID=279fe21d-2caa-4437-84c3-ab9e9dae20f5.AgentGateway0”, 其中279fe21d-2caa-4437-84c3-ab9e9dae20f5.AgentGateway0为guid, 是CC-Gateway对座席调用接口的鉴权信息。
3. 获取guid之后, 可以通过调用重置技能队列 ( resetSkill ) 接口, 签入指定的技能队列。
4. 如图一所示, 采用座席事件轮询方式, 完成签入技能后, 调用获取座席事件 ( getAgentEvent ) 接口定时轮询获取座席事件, 该接口不仅用于获取座席的事件, 同时也是座席与CC-Gateway之间的心跳线;

如图二所示, 采用座席事件推送方式, 签入成功和完成签入技能后, 事件会直接推送到签入时传入的回调地址, 只需调用座席心跳接口保持CC-Gateway与三方回调地址心跳。

当CC-Gateway采用动态鉴权时, guid会动态更新, 并且当guid更新后, 都会在轮询获取事件接口和座席心跳接口的HTTP请求的响应response中的headers中新增返回一个Set-GUID的header, 同样需要获取并更新guid, 并且在其他接口调用中用新获取的guid进行设置。

## 2.2 前提条件

1. 已经完成CC-Gateway部署和调试。
2. 本实例以java作为开发语言, 以eclipse作为开发工具。
3. 在eclipse中, File -->new -->Java Project新建java工程。
4. 新建java工程后, 选择右键, 新建包com.huawei.example。
5. 在com.huawei.example包下新建MainTest.java文件。
6. 将实现示例中的代码复制到MainTest.java文件中。

## 2.3 实现过程

### 涉及接口

1. 登录 ( login )  
请求方法: PUT  
请求的url: https://ip:port/agentgateway/resource/onlineagent/{agentid}  
请参考[签入](#)
2. 强制登录 ( forclogin )  
请求方法: PUT  
请求的url: https://ip:port/agentgateway/resource/onlineagent/{agentid}/forclogin  
请参考[强制签入](#)
3. 设置技能队列 ( resetSkill )

请求方法: POST

请求的url: <https://ip:port/agentgateway/resource/onlineagent/{agentid}/resetskill/{autoflag}?skillid={skillid}&phonelinkage={phonelinkage}>

请参考[重置技能队列](#)

#### 4. 获取座席事件 ( getAgentEvent )

请求方法: GET

请求的url: <https://ip:port/agentgateway/resource/agentevent/{agentid}>

请参考[轮询方式获取单座席事件](#)

## 如何发送 PUT 的 HTTP 请求

请参考MainTest.java中的

```
/**
 * Send http's PUT request
 * @param url the address of the request
 * @param entityParams the paramters of entity
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<string, string> put(string url, Map<string, Object> entityParams, Map<string, string> headers)
```

## 如何发送 POST 的 HTTP 请求

请参考MainTest.java中的

```
/**
 * Send http's POST request
 * @param url the address of the request
 * @param entityParams the paramters of entity
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<string, string> post(string url, Map<string, Object> entityParams, Map<string, string> headers)
```

## 如何发送 GET 的 HTTP 请求

请参考MainTest.java中的

```
/**
 * Send http's POST request
 * @param url the address of the request
 * @param entityParams the paramters of entity
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<string, string> get(string url, Map<string, Object> entityParams, Map<string, string> headers)
```

## 实现登录接口

请参考MainTest.java中的

```
/**
 * log in
 * @param workNo the work no of the agent
 * @param password the password of the agent
 * @param phoneNumber the phone number of the agent
```

```
* @return
*/
public Map<string, string> login(string workNo, string password, string phoneNumber)
```

## 实现强制登录接口

请参考MainTest.java中的

```
/**
 * When agent has logged in, call the interface to forcibly log in
 * @param workNo the work no of the agent
 * @param password the password of the agent
 * @param phoneNumber the phone number of the agent
 * @return
 */
public Map<string, string> forceLogin(string workNo, string password, string phoneNumber)
```

## 实现签入技能队列接口

请参考MainTest.java中的

```
/**
 * After log in, reset the skills
 * @param workNo the work no of the agent
 * @param autoFlag Is automatically signed into the skill queue
 * @param skillId the id of the skill. if has more than one skill that need to be sign, it's split by ;
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<string, string> resetSkill(string workNo, boolean autoFlag,
string skillId, Map<string, string> headers)
```

## 实现获取座席事件接口

请参考MainTest.java中的

```
/**
 * Get the agent's event
 * @param workNo workNo the work no of the agent
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<string, string> getAgentEvent(string workNo, Map<string, string> headers)
```

## 2.4 实现示例

### 📖 说明

调用时请按照实际的cc-gateway地址修改样例:

- https://ip:port/agentgateway  
其中, ip为CC-Gateway服务器地址, port为CC-Gateway服务器的HTTPS端口号。
- WORKNO为座席工号, PASSWORD为座席密码, PHONENUMBER为座席软电话号码。
- IF\_TRUST\_ALL为是否信任所有证书, 取值范围为false/true, 默认为false, 需要加载CC-Gateway客户端证书truststore.jks, 并配置客户端证书密码TRUSTSTORE\_PASSWORD。配置为true有安全风险, 请谨慎使用。
- truststore.jks证书需要跟编译后的MainTest.class放在同一层目录。

```
package com.huawei.example;

import com.alibaba.fastjson.JSON;
import lombok.extern.log4j.Log4j2;
```

```
import org.bouncycastle.crypto.BlockCipher;
import org.bouncycastle.crypto.engines.AESEngine;
import org.bouncycastle.crypto.prng.SP800SecureRandomBuilder;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import sun.security.provider.Sun;
import org.apache.commons.io.IOUtils;
import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.config.Registry;
import org.apache.http.config.RegistryBuilder;
import org.apache.http.conn.socket.ConnectionSocketFactory;
import org.apache.http.conn.socket.PlainConnectionSocketFactory;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.impl.conn.PoolingHttpClientConnectionManager;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.security.KeyStore;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.SecureRandom;
import java.security.Security;
import java.security.cert.CRL;
import java.security.cert.CertPathBuilder;
import java.security.cert.CertStore;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.CollectionCertStoreParameters;
import java.security.cert.PKIXBuilderParameters;
import java.security.cert.PKIXParameters;
import java.security.cert.PKIXRevocationChecker;
import java.security.cert.X509CertSelector;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.Collection;
import java.util.EnumSet;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;
import java.util.concurrent.TimeUnit;

import javax.net.ssl.CertPathTrustManagerParameters;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;

public class MainTest {
    private static final String AGENTGATEWAY_URL = "https://ip:port/agentgateway";

    private static final String WORKNO = "100";
```



```
private static final String PASSWORD = "xxxxxxx";

private static final String PHONENUMBER = "88880001";

private static final boolean IF_TRUST_ALL = false;

private static final String TRUSTSTORE_PASSWORD = "*****";

private static final long MAX_IDLE_TIME = 2L;

private static final String HTTP = "http";

private static final String HTTPS = "https";

private static SSLConnectionSocketFactory sslsf = null;

private static PoolingHttpClientConnectionManager cm = null;

private static CloseableHttpClient httpClient = null;

private static final int HTTPCLIENT_CM_MAXCONNECTION = 200;

private static final int HTTPCLIENT_CM_MAXPERROUTECONNECTION = 20;

private static final int CONNECT_TIMEOUT = 10000;

private static final int CONNECTION_REQUEST_TIMEOUT = 10000;

private static final int SOCKET_TIMEOUT = 20000;

private static final int AES_KEY_BIT_LENGTH = 256;

private static final int ENTROPY_SOURCE_BIT_LENGTH = 384;

private static final boolean FORCE_RESEED = false;

private static final boolean PREDICTION_RESISTANT = false;

private static final FastSecureRandomUtil FAST_SECURE_RANDOM = new FastSecureRandomUtil();

/**
 * @param args
 */
public static void main(String[] args) {
    MainTest test = new MainTest();
    Map<String, String> resultMap = test.login(WORKNO, PASSWORD, PHONENUMBER);
    if (resultMap == null) {
        System.out.println("Send http request to agentgateway failed");
        return;
    }
    String guid = resultMap.get("guid");
    String agwResultString = resultMap.get("result");
    HashMap<String, Object> agwResult = JSON.parseObject(agwResultString, HashMap.class);
    if (agwResult == null) {
        System.out.println("Parse json to map failed");
        return;
    }

    if ("0".equals(agwResult.get("retcode"))) {
        //log in successfully
        System.out.println("----login ok");
    } else if ("100-002".equals(agwResult.get("retcode"))) {
        //Agent has logged in, but Agent can forcibly log in
        System.out.println("----has login");
        resultMap = test.forceLogin(WORKNO, PASSWORD, PHONENUMBER);
        if (resultMap == null) {
            System.out.println("Send http request to agentgateway failed");
            return;
        }
    }
}
```

```
    }

    guid = resultMap.get("guid");
    agwResultString = resultMap.get("result");
    agwResult = JSON.parseObject(agwResultString, HashMap.class);
    if (agwResult == null) {
        System.out.println("Parse json to map failed");
        return;
    }

    if ("0".equals(agwResult.get("retcode"))) {
        //forcibly log in successfully
        System.out.println("----forceLogin ok");
    } else {
        System.out.println("----forceLogin failed");
        return;
    }
} else {
    System.out.println("----login failed");
    return;
}

//Need to add guid to http request for Authentication after log in
Map<String, String> headers = new HashMap<String, String>();
headers.put("guid", guid);

//After log in,when agent reset skills, agent can receive call from customer request
resultMap = test.resetSkill(WORKNO, true, "", headers);
if (resultMap == null) {
    System.out.println("Send http request to agentgateway failed");
    return;
}
agwResultString = resultMap.get("result");
agwResult = JSON.parseObject(agwResultString, HashMap.class);
if (agwResult == null) {
    System.out.println("Parse json to map failed");
    return;
}

if ("0".equals(agwResult.get("retcode"))) {
    //Reset skills successfully
    System.out.println("----resetSkill ok");
} else {
    System.out.println("----resetSkill failed");
    return;
}

/**
 *After log in and reset skill successfully. We need to start a thread to get the agent's event by interval.
 */
Map<String, String> event = null;
while (true) {
    event = null;
    resultMap = test.getAgentEvent(WORKNO, headers);
    if (resultMap == null) {
        System.out.println("Send http request to agentgateway failed");
        return;
    }

    //if agentgateway uses dynamic authentication mode, the guid will be updated by interval.
    //You can get the guid from the reponse of get the agent's event request
    guid = resultMap.get("guid");
    if (guid != null && guid != "") {
        headers = new HashMap<String, String>();
        headers.put("guid", guid);
    }

    agwResultString = resultMap.get("result");
    agwResult = JSON.parseObject(agwResultString, HashMap.class);
```

```

        if (agwResult == null) {
            System.out.println("Parse json to map failed");
            return;
        }

        if ("0".equals(agwResult.get("retcode"))) {
            //Get the agent's event successfully
            event = (Map<String, String>) agwResult.get("event");
            if (event != null) {
                System.out.println("----getAgentEvent ok:" + agwResultString);
            } else {
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    System.out.println("getAgentEvent InterruptedException ");
                }
            }
        } else if ("000-003".equals(agwResult.get("retcode")) ||
"100-006".equals(agwResult.get("retcode"))) {
            //No right to visit the interface
            break;
        } else {
            System.out.println("----getAgentEvent failed");
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                System.out.println("getAgentEvent InterruptedException ");
            }
        }
    }
}

private static TrustManager[] trustManagers = new TrustManager[]{
    new X509TrustManager() {
        @Override
        public void checkClientTrusted(X509Certificate[] x509Certificates, String s) throws
CertificateException {
        }

        @Override
        public void checkServerTrusted(X509Certificate[] x509Certificates, String s) throws
CertificateException {
        }

        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[0];
        }
    }
};

static {
    InputStream inStream = null;
    InputStream crlInputStream = null;
    SSLContext context = null;
    try {
        context = SSLContext.getInstance("TLSv1.2");

        if (IF_TRUST_ALL) {
            context.init(null, trustManagers, getSecurityRandomInstance());
        } else {

            CertificateFactory certificateFactory = CertificateFactory.getInstance("X.509");
            inStream = MainTest.class.getResourceAsStream("truststore.jks");

            KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
            keyStore.load(inStream, TRUSTSTORE_PASSWORD.toCharArray());
        }
    }
}

```

```

        String crlFile = "";
        if (!crlFile.isEmpty()) {
            crlInputStream = new FileInputStream(crlFile);
        }
        loadTrustCertificate(context, certificateFactory, keyStore, crlInputStream);
    }

    sslsf = new SSLConnectionSocketFactory(context, new String[]{"TLSv1.2"}, null,
        NoopHostnameVerifier.INSTANCE);
    httpClient = createHttpClient();
} catch (Exception e) {
    System.out.println("init occur Exception: " + e.getMessage());
} finally {
    IOUtils.closeQuietly(inStream, null);
    IOUtils.closeQuietly(crlInputStream, null);
}
}

public static SecureRandom getSecurityRandomInstance() {
    try {
        SecureRandom secureRandom = FAST_SECURE_RANDOM.getSecureRandom();
        secureRandom.setSeed(secureRandom.generateSeed(64));
        return secureRandom;
    } catch (Exception e) {
        System.out.println("get SecureRandom instance failed" + e);
        throw e;
    }
}

private static void loadTrustCertificate(SSLContext context, CertificateFactory certificateFactory,
    KeyStore keyStore, InputStream crlInputStream) {
    try {
        Collection<CRL> crls = null;
        if (null != crlInputStream) {
            crls = new HashSet();
            crls.add(certificateFactory.generateCRL(crlInputStream));
        }
        TrustManagerFactory tmf = TrustManagerFactory.getInstance("PKIX");
        PKIXParameters pkixParams = new PKIXBuilderParameters(keyStore, new X509CertSelector());
        if (null != crls) {
            List<CertStore> certStores = new ArrayList();
            certStores.add(CertStore.getInstance("Collection", new CollectionCertStoreParameters(crls)));
            CertPathBuilder certPathBuilder = CertPathBuilder.getInstance("PKIX", new Sun());
            PKIXRevocationChecker revokeChecker = (PKIXRevocationChecker)
certPathBuilder.getRevocationChecker();
            revokeChecker.setOptions(EnumSet.of(PKIXRevocationChecker.Option.PREFER_CRLS,
                PKIXRevocationChecker.Option.NO_FALLBACK));
            pkixParams.setRevocationEnabled(true);
            pkixParams.setCertStores(certStores);
            pkixParams.addCertPathChecker(revokeChecker);
        } else {
            pkixParams.setRevocationEnabled(false);
        }

        tmf.init(new CertPathTrustManagerParameters(pkixParams));
        context.init(null, tmf.getTrustManagers(), getSecurityRandomInstance());
    } catch (Exception e) {
        System.out.println("loadTrustCertificate occur exception: " + e.getMessage());
    }
}

/**
 * createHttpClient
 *
 * @return CloseableHttpClient
 * @throws Exception Exception
 */
private static CloseableHttpClient createHttpClient() {
    Registry<ConnectionSocketFactory> registry =

```

```

RegistryBuilder.<ConnectionSocketFactory>create().register(
    HTTP, new PlainConnectionSocketFactory()).register(HTTPS, sslsf).build();

cm = new PoolingHttpClientConnectionManager(registry);

int maxConnect;
int maxPerRouteConnect;
String maxConnectNum = "10";
String maxPreConnectNum = "100";
try {
    maxConnect = Integer.parseInt(maxConnectNum);
} catch (NumberFormatException e) {
    maxConnect = HTTPCLIENT_CM_MAXCONNECTION;
}

try {
    maxPerRouteConnect = Integer.parseInt(maxPreConnectNum);
} catch (NumberFormatException e) {
    maxPerRouteConnect = HTTPCLIENT_CM_MAXPERROUTECONNECTION;
}

cm.setMaxTotal(maxConnect);
cm.setDefaultMaxPerRoute(maxPerRouteConnect);

RequestConfig requestConfig = RequestConfig.custom()
    .setConnectTimeout(CONNECT_TIMEOUT)
    .setConnectionRequestTimeout(CONNECTION_REQUEST_TIMEOUT)
    .setSocketTimeout(SOCKET_TIMEOUT)
    .build();

httpClient = HttpClients.custom()
    .setDefaultRequestConfig(requestConfig)
    .evictExpiredConnections()
    .evictIdleConnections(MAX_IDLE_TIME, TimeUnit.SECONDS)
    .disableAutomaticRetries()
    .setConnectionManager(cm)
    .build();
return httpClient;
}

/**
 * log in
 *
 * @param workNo    the work no of the agent
 * @param password  the password of the agent
 * @param phoneNumber the phone number of the agent
 * @return
 */
public Map<String, String> login(String workNo, String password, String phoneNumber) {
    String loginUrl = AGENTGATEWAY_URL + "/resource/onlineagent/" + workNo;
    Map<String, Object> loginParam = new HashMap<String, Object>();
    loginParam.put("password", password);
    loginParam.put("phonenum", phoneNumber);
    return put(loginUrl, loginParam, null);
}

/**
 * When agent has logged in, call the interface to forcibly log in
 *
 * @param workNo    the work no of the agent
 * @param password  the password of the agent
 * @param phoneNumber the phone number of the agent
 * @return
 */
public Map<String, String> forceLogin(String workNo, String password, String phoneNumber) {
    String loginUrl = AGENTGATEWAY_URL + "/resource/onlineagent/" + workNo + "/forcelogin";
    Map<String, Object> loginParam = new HashMap<String, Object>();
    loginParam.put("password", password);
    loginParam.put("phonenum", phoneNumber);
}

```

```

    return put(loginUrl, loginParam, null);
}

/**
 * After log in, reset the skills
 *
 * @param workNo the work no of the agent
 * @param autoFlag Is automatically signed into the skill queue
 * @param skillId the id of the skill. if has more than one skill that need to be sign, it is split by ;
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<String, String> resetSkill(String workNo, boolean autoFlag,
                                     String skillId, Map<String, String> headers) {
    String url = AGENTGATEWAY_URL + "/resource/onlineagent/" + workNo + "/resetskill/" + autoFlag;
    if (skillId != null && skillId != "") {
        url = url + "?skillid=" + skillId;
    }
    return post(url, null, headers);
}

/**
 * Get the agent's event
 *
 * @param workNo workNo the work no of the agent
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<String, String> getAgentEvent(String workNo, Map<String, String> headers) {
    String url = AGENTGATEWAY_URL + "/resource/agentevent/" + workNo;
    return get(url, headers);
}

/**
 * Send http's PUT request
 *
 * @param url the address of the request
 * @param entityParams the paramters of entity
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<String, String> put(String url, Map<String, Object> entityParams, Map<String, String>
headers) {
    CloseableHttpClient httpClient = null;
    Map<String, String> resultMap = null;
    try {
        httpClient = createHttpClient();

        HttpPut httpPut = new HttpPut(url);
        if (entityParams != null) {
            String jsonString = JSON.toJSONString(entityParams);
            HttpEntity entity = new StringEntity(jsonString);
            httpPut.setEntity(entity);
        }

        if (headers != null) {
            Set<Entry<String, String>> headersSet = headers.entrySet();
            for (Entry<String, String> entry : headersSet) {
                httpPut.setHeader(entry.getKey(), entry.getValue());
            }
        }
        httpPut.setHeader("Content-Type", "application/json");
        HttpResponse response = httpClient.execute(httpPut);
        int returnCode = response.getStatusLine().getStatusCode();
        if (returnCode == HttpStatus.SC_OK) {
            InputStream is = response.getEntity().getContent();
            BufferedReader in = new BufferedReader(new InputStreamReader(is));
            StringBuffer buffer = new StringBuffer();

```

```

        String line = "";
        while ((line = in.readLine()) != null) {
            buffer.append(line);
        }

        Header[] allHeaders = response.getAllHeaders();
        String guid = "";
        if (allHeaders != null && allHeaders.length > 0) {
            for (Header header : allHeaders) {
                if (header.getName().equals("Set-GUID")) {
                    String setGuid = header.getValue();
                    if (setGuid != null) {
                        guid = setGuid.replace("JSESSIONID=", "");
                    }
                    break;
                }
            }
        }
        resultMap = new HashMap<String, String>();
        resultMap.put("guid", guid);
        resultMap.put("result", buffer.toString());
    } else {
        System.out.println(returnCode);
    }
    return resultMap;
} catch (ClientProtocolException e) {
    System.out.println("HttpPut ClientProtocolException ");
} catch (IOException e) {
    System.out.println("HttpPut IOException ");
} catch (Exception e) {
    System.out.println("HttpPut Exception ");
} finally {
    httpClient.getConnectionManager().shutdown();
}
return resultMap;
}

/**
 * Send http's POST request
 *
 * @param url      the address of the request
 * @param entityParams the paramters of entity
 * @param headers  the field is used to set the header of http request
 * @return
 */
public Map<String, String> post(String url, Map<String, Object> entityParams, Map<String, String>
headers) {
    CloseableHttpClient httpClient = null;
    Map<String, String> resultMap = null;
    try {
        httpClient = createHttpClient();

        HttpPost httpPost = new HttpPost(url);
        if (entityParams != null) {
            String jsonString = JSON.toJSONString(entityParams);
            HttpEntity entity = new StringEntity(jsonString);
            httpPost.setEntity(entity);
        }

        if (headers != null) {
            Set<Entry<String, String>> headersSet = headers.entrySet();
            for (Entry<String, String> entry : headersSet) {
                httpPost.setHeader(entry.getKey(), entry.getValue());
            }
        }
        httpPost.setHeader("Content-Type", "application/json");
        HttpResponse response = httpClient.execute(httpPost);
    }
}

```

```
int returnCode = response.getStatusLine().getStatusCode();
if (returnCode == HttpStatus.SC_OK) {
    InputStream is = response.getEntity().getContent();
    BufferedReader in = new BufferedReader(new InputStreamReader(is));
    StringBuffer buffer = new StringBuffer();
    String line = "";
    while ((line = in.readLine()) != null) {
        buffer.append(line);
    }
    resultMap = new HashMap<String, String>();
    resultMap.put("result", buffer.toString());
} else {
    System.out.println(returnCode);
}
return resultMap;
} catch (ClientProtocolException e) {
    System.out.println("HttpPost ClientProtocolException");
} catch (IOException e) {
    System.out.println("HttpPost IOException");
} catch (Exception e) {
    System.out.println("HttpPost Exception");
} finally {
    httpClient.getConnectionManager().shutdown();
}
return resultMap;
}

/**
 * Send http's GET request
 *
 * @param url the address of the request
 * @param headers the field is used to set the header of http request
 * @return
 */
public Map<String, String> get(String url, Map<String, String> headers) {
    CloseableHttpClient httpClient = null;
    Map<String, String> resultMap = null;
    try {
        httpClient = createHttpClient();

        HttpGet httpGet = new HttpGet(url);
        if (headers != null) {
            Set<Entry<String, String>> headersSet = headers.entrySet();
            for (Entry<String, String> entry : headersSet) {
                httpGet.setHeader(entry.getKey(), entry.getValue());
            }
        }
        httpGet.setHeader("Content-Type", "application/json");
        HttpResponse response = httpClient.execute(httpGet);
        int returnCode = response.getStatusLine().getStatusCode();
        if (returnCode == HttpStatus.SC_OK) {
            InputStream is = response.getEntity().getContent();
            BufferedReader in = new BufferedReader(new InputStreamReader(is));
            StringBuffer buffer = new StringBuffer();
            String line = "";
            while ((line = in.readLine()) != null) {
                buffer.append(line);
            }

            Header[] allHeaders = response.getAllHeaders();
            String guid = "";
            if (allHeaders != null && allHeaders.length > 0) {
                for (Header header : allHeaders) {
                    if (header.getName().equals("Set-GUID")) {
                        String setGuid = header.getValue();
                        if (setGuid != null) {
                            guid = setGuid.replace("JSESSIONID=", "");
                        }
                    }
                }
            }
        }
    }
}
```



```
        break;
    }
}
}
resultMap = new HashMap<String, String>();
resultMap.put("guid", guid);
resultMap.put("result", buffer.toString());
} else {
    System.out.println(returnCode);
}
return resultMap;
} catch (ClientProtocolException e) {
    System.out.println("HttpGet ClientProtocolException");
} catch (IOException e) {
    System.out.println("HttpGet IOException");
} catch (Exception e) {
    System.out.println("HttpGet Exception");
} finally {
    httpClient.getConnectionManager().shutdown();
}
return resultMap;
}
}
}

package com.huawei.example;

import lombok.extern.log4j.Log4j2;

import org.bouncycastle.crypto.BlockCipher;
import org.bouncycastle.crypto.engines.AESEngine;
import org.bouncycastle.crypto.prng.SP800SecureRandomBuilder;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.Security;

@Log4j2
public class FastSecureRandomUtil {
    private static final int AES_KEY_BIT_LENGTH = 256;

    private static final int ENTROPY_SOURCE_BIT_LENGTH = 384;

    private static final boolean FORCE_RESEED = false;

    private static final boolean PREDICTION_RESISTANT = false;

    static {
        if (Security.getProvider(BouncyCastleProvider.PROVIDER_NAME) == null) {
            Security.addProvider(new BouncyCastleProvider());
        }
    }

    private final SecureRandom fastRandom;

    public FastSecureRandomUtil() throws Exception {
        SecureRandom source = null;
        try {
            source = SecureRandom.getInstanceStrong();
        } catch (NoSuchAlgorithmException e) {
            log.error("FastSecureRandom NoSuchAlgorithmException: {}. ", e.getMessage());
            throw new Exception("FastSecureRandom NoSuchAlgorithmException.", e);
        }

        BlockCipher cipher = new AESEngine();

        byte[] nonce = new byte[ENTROPY_SOURCE_BIT_LENGTH / Byte.SIZE];
        source.nextBytes(nonce);
    }
}
```

```
        fastRandom = new SP800SecureRandomBuilder(source,
PREDICTION_RESISTANT).setEntropyBitsRequired(
        ENTROPY_SOURCE_BIT_LENGTH).buildCTR(cipher, AES_KEY_BIT_LENGTH, nonce,
FORCE_RESEED);
    }

    public byte[] getRandomBytes(int byteSize) {
        byte[] randomBytes = new byte[byteSize];
        fastRandom.nextBytes(randomBytes);
        return randomBytes;
    }

    public SecureRandom getSecureRandom() {
        return fastRandom;
    }
}
```

## 2.5 返回结果

座席未登录的情况下，输出结果如下：

```
----login ok
----resetSkill ok
----getAgentEvent ok:{"message":"","retcode":"0","event":
{"content":null,"eventType":"AgentOther_InService","workNo":"291"}}
----getAgentEvent ok:{"message":"","retcode":"0","event":
{"content":null,"eventType":"AgentState_Ready","workNo":"291"}}
```

座席已登录的情况下，输出结果如下：

```
----has login
----forceLogin ok
----resetSkill ok
----getAgentEvent ok:{"message":"","retcode":"0","event":
{"content":null,"eventType":"AgentOther_InService","workNo":"291"}}
----getAgentEvent ok:{"message":"","retcode":"0","event":
{"content":null,"eventType":"AgentState_Ready","workNo":"291"}}
```

# 3 基本应用开发

[3.1 如何查看接口调用日志](#)

[3.2 基本功能与设置](#)

[3.3 基本语音呼叫](#)

[3.4 通话建立后](#)

[3.5 实时质检](#)

[3.6 放音](#)

[3.7 基本多媒体呼叫](#)

## 3.1 如何查看接口调用日志

- 用户的所有接口请求日志会打印在/home/elpis/tomcat/logs/agentgateway/agentgateway-rest.log。

## 3.2 基本功能与设置

### 3.2.1 座席签入

1. 通过调用[签入](#)进行登录操作，登录操作的相关参数信息放置在请求内容中，以JSON格式进行标识，例子如下：

```
{"password":"xxxxxxx","phonenum":"6001"}
```

password代表座席的签入密码，phonenum代表座席的电话号码。
2. 座席签入调用成功后，会返回结果信息，如下：

```
{"retcode":"0","msg":"","result":{"workno":"6001","mediatype":"TTT"}}
```

mediatype字段表示签入媒体服务器是否成功，T表示成功，F表示失败，三位分别表示CTIServer、WebM、MailM。只有签入CTIServer成功，才能进行音视频呼叫业务处理；只有签入WebM成功，才能进行文字交谈业务处理；当前不支持签入MailM。
3. 如果座席签入失败，并且retcode返回为“100-002”，表示座席已经签入，可以调用[强制签入](#)进行强制签入操作。

## 3.2.2 查询座席的配置技能（可选）

座席签入后，可通过调用[查询配置技能队列](#)，获取到当前座席可以签入的技能队列。

## 3.2.3 签入座席的技能

座席使用[重置技能队列](#)签入座席所配置的所有技能队列或签入指定的技能队列。

## 3.2.4 获取座席事件

签入技能队列成功，表示座席可以正式开始提供服务，此时座席客户端需要启动一个定时器（建议200ms）定时轮询获取这个当前这个座席自己的事件。

- 通过调用[轮询方式获取单座席事件](#)接口获取事件并进行处理。

## 3.2.5 座席签出

座席已经签入系统后，如果需要签出，可以调用[签出](#)进行签出操作，返回的调用成功结果信息如下：

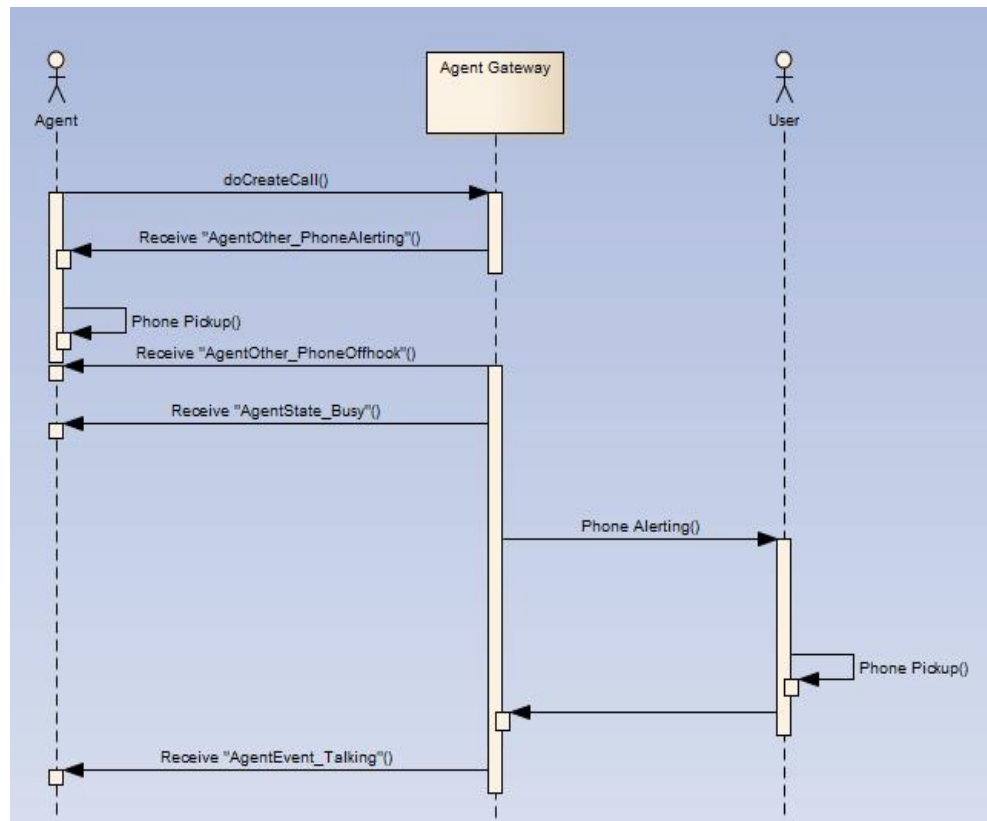
```
{"retcode":"0","msg":"","result:null"}
```

# 3.3 基本语音呼叫

## 3.3.1 呼出

- 应用场景  
座席主动发起呼叫

图 3-1 呼出流程



- 前提条件
  - 座席已经签入系统.
  - 座席当前无正在接听的呼叫（不包括保持后的呼叫）。
- 实现过程  
请参考 [普通外呼](#)
- 触发事件
  - 非长通座席  
座席物理话机振铃事件（AgentOther\_PhoneAlerting）
  - 座席摘机后  
座席物理话机摘机事件（AgentOther\_PhoneOffhook）  
座席忙事件（AgentState\_Busy）  
对方振铃（AgentEvent\_Customer\_Alerting）
  - 如果座席未摘机挂机  
外呼失败事件（AgentEvent\_Call\_Out\_Fail）  
连接失败事件（AgentEvent\_Connect\_Fail）  
座席物理话机挂机（AgentOther\_PhoneRelease）
  - 被叫摘机后  
座席通话事件（AgentEvent\_Talking），表示通话成功
  - 如果被叫未摘机、挂机  
座席物理话机挂机（AgentOther\_PhoneRelease）  
外呼失败事件（AgentEvent\_Call\_Out\_Fail）

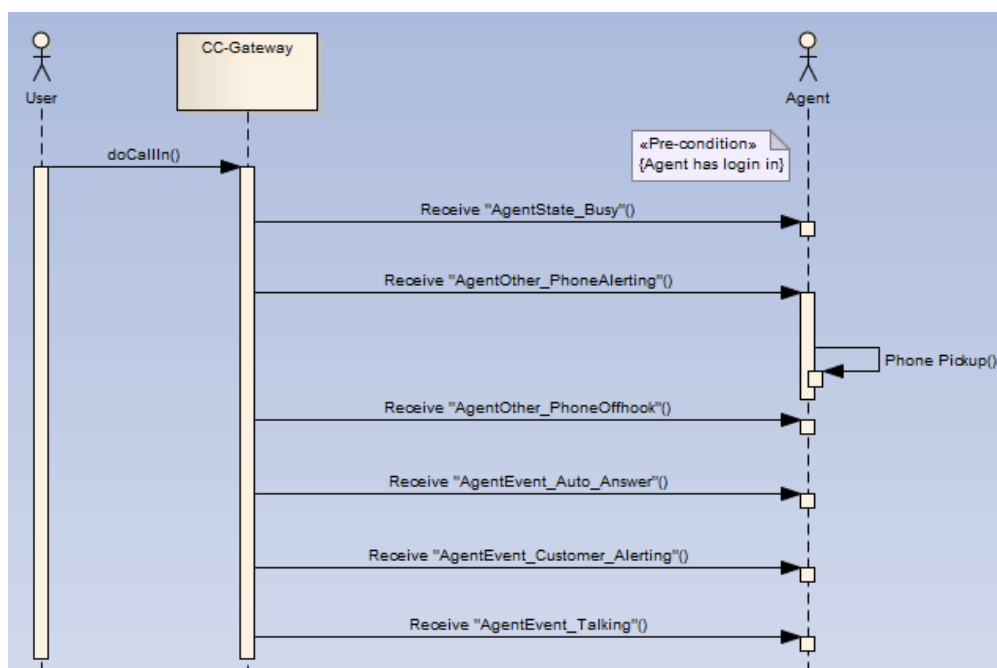
- 连接失败事件 ( AgentEvent\_Connect\_Fail )
- 示闲事件 ( AgentState\_Ready )

## 3.3.2 呼入

### 3.3.2.1 自动应答

- 应用场景  
座席在接到来话后，并且话机已经摘机的情况下，座席不需要在座席业务软件上点击应答。

图 3-2 呼入流程——自动应答

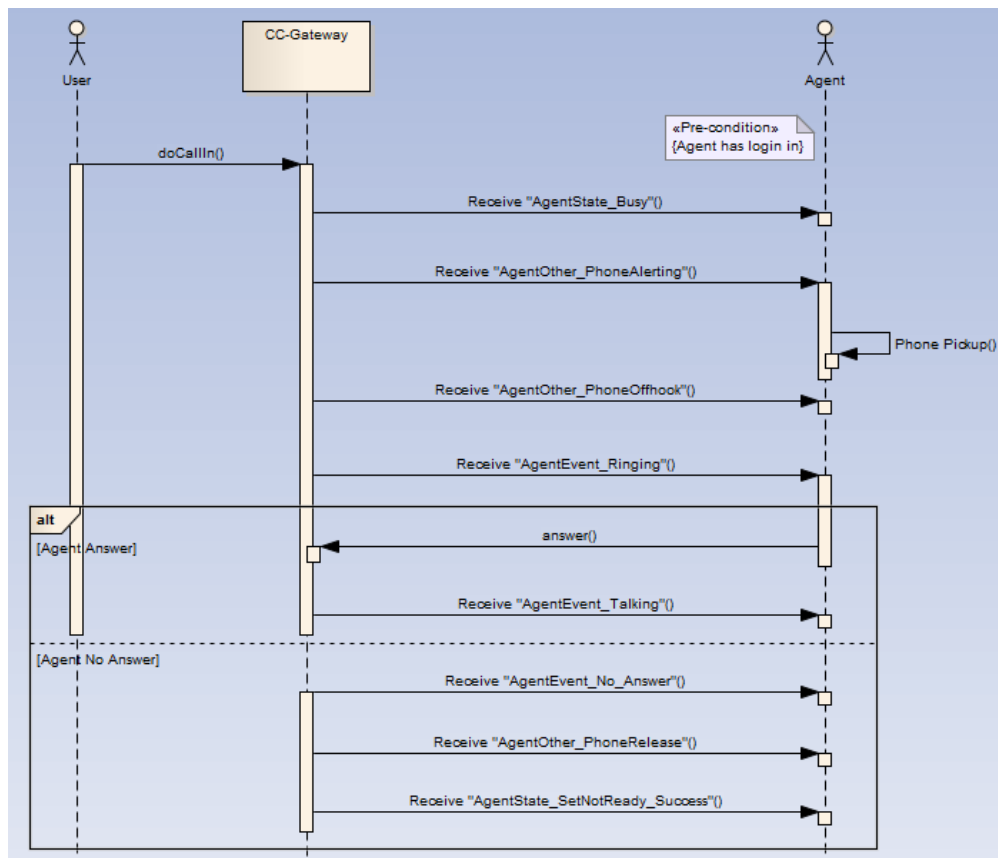


- 前提条件
  - 座席已经签入系统。
- 实现过程  
请参考[签入](#)，在签入时设置“是否自动应答”为true或者调用[设置是否自应答](#)接口设置为自动应答。
- 触发事件  
座席忙事件 ( AgentState\_Busy )，表示座席被预占
  - 非长通座席  
座席物理话机振铃事件 ( AgentOther\_PhoneAlerting )
  - 座席摘机后  
座席物理话机摘机事件 ( AgentOther\_PhoneOffhook )  
座席自动应答事件 ( AgentEvent\_Auto\_Answer )  
对方振铃 ( AgentEvent\_Customer\_Alerting )  
座席进入Talking事件 ( AgentEvent\_Talking )，表示通话建立成功。

### 3.3.2.2 手工应答

- 应用场景  
座席在接到来话后，并且话机已经摘机的情况下，座席还需要在座席业务软件上点击应答。

图 3-3 呼入流程——手工应答



- 前提条件
  - 座席已签入系统。
- 实现过程  
请参考[签入](#)，在签入时设置“是否自动应答”为false或者调用[设置是否自应答](#)接口设置为手工应答，收到用户呼入请求后，调用[呼叫应答](#)接口进行应答。
- 触发事件
  - 座席忙事件（AgentState\_Busy），表示座席被预占
    - 非长通座席
    - 座席物理话机振铃事件（AgentOther\_PhoneAlerting）
  - 座席摘机后
    - 座席物理话机摘机事件（AgentOther\_PhoneOffhook）
    - 座席来电提醒事件（AgentEvent\_Ringing），表示收到用户呼入请求。
    - 对方振铃事件（AgentEvent\_Customer\_Alerting）
  - 座席调用呼叫应答接口后
    - 座席进入Talking事件（AgentEvent\_Talking），表示通话建立成功。

- 座席不调用接口超时  
座席久不应答事件 ( AgentEvent\_No\_Answer )

### 3.3.2.3 查询应答来话前呼叫信息

- 应用场景  
座席设置非自动应答时，用户呼入来话后在座席应答前，需要知道呼叫来自哪个技能队列，用户的号码是什么。
- 前提条件
  - 座席已签入系统。
  - 平台有呼叫分配到此座席（话机振铃或摘机后）。
- 实现过程  
请参考[查询应答来话前呼叫信息](#)

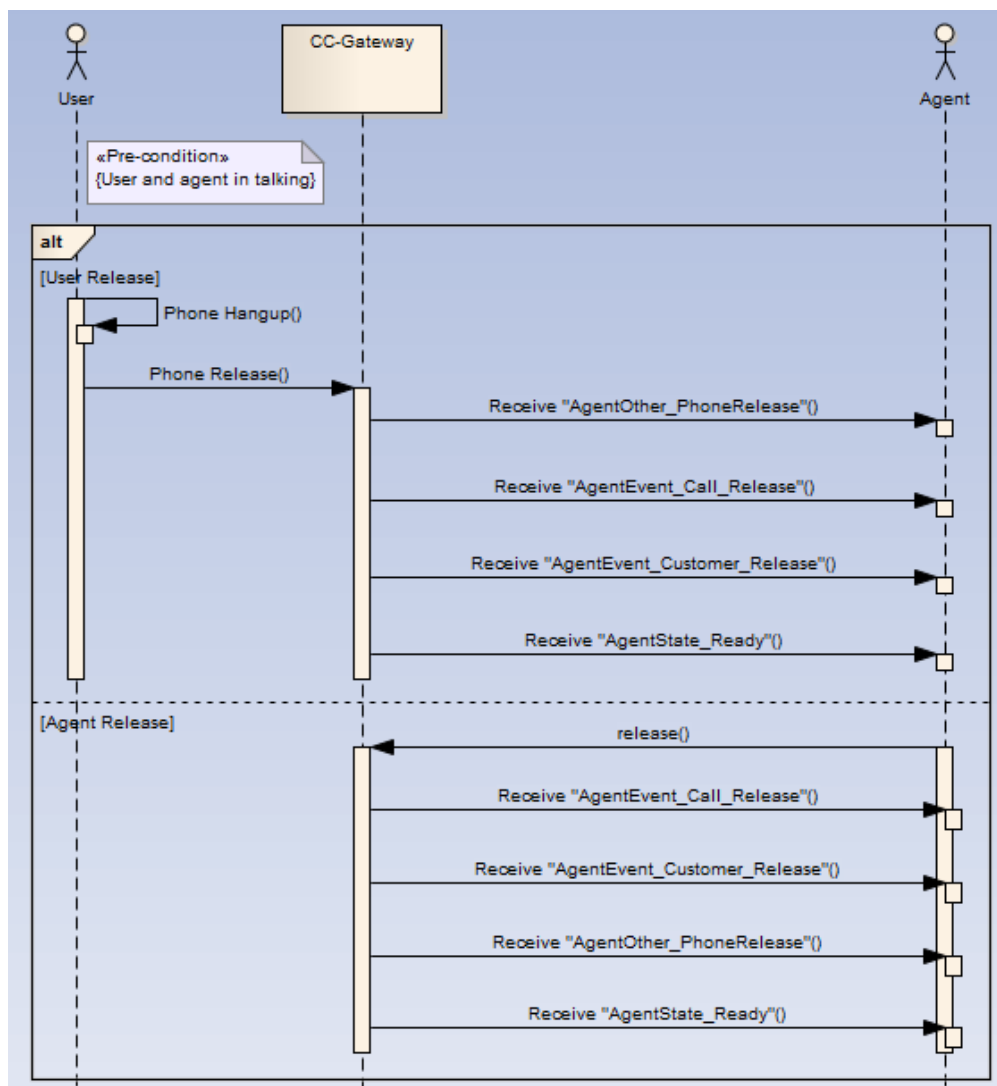
## 3.4 通话建立后

### 3.4.1 释放呼叫

- 应用场景  
座席需要释放呼叫。



图 3-4 释放呼叫流程



- 前提条件  
座席已经处于通话状态。
- 实现过程  
请参考[挂断呼叫](#)
- 触发事件
  - 座席挂断呼叫事件 ( AgentEvent\_Call\_Release )
  - 客户挂断呼叫事件 ( AgentEvent\_Customer\_Release )
  - 座席物理话机挂断事件 ( AgentOther\_PhoneRelease )
  - 示闲事件 ( AgentState\_Ready )

## 3.4.2 保持

### 3.4.2.1 保持呼叫

- 应用场景  
座席需要保持呼入或呼出呼叫，保持的过程中用户听到保持音。

- 前提条件  
座席已经处于通话状态。
- 实现过程  
[呼叫保持](#)
- 触发事件  
座席保持成功事件 ( AgentEvent\_Hold )

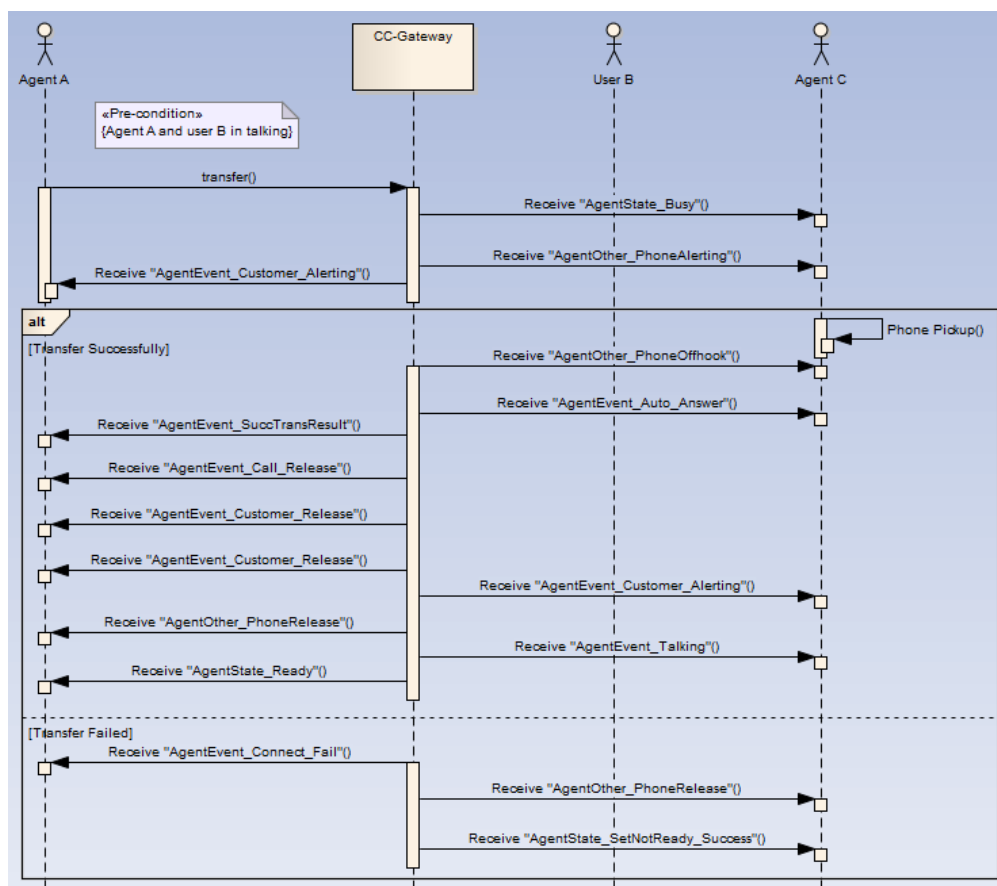
### 3.4.2.2 取保持呼叫

- 应用场景  
座席需要取保持呼叫。
- 前提条件  
座席已经处于保持状态。
- 实现过程  
请参考[取消保持](#)
- 触发事件  
座席通话事件 ( AgentEvent\_Talking )

### 3.4.3 转移

- 应用场景  
座席A与用户B建立通话后，希望将呼叫转移给其他座席或外部电话处理。转移分为内部转和外部转。
  - 内部转是指将转移呼叫给内部设备，如座席、IVR、技能队列、接入码。
    - 转座席时支持成功转和释放转；
    - 转技能队列时支持成功转；
    - 转IVR时支持释放转和挂起转；
    - 转接入码时支持成功转和释放转。
  - 外部转是指将呼叫转移给外部用户，即电话号码。外部转支持通话转和三方转，即座席A与用户B和用户C形成三方通话（通话转形成的三方通话中，用户B实际是被保持的），通话转或三方转成功后：
    - 如果座席A先释放，则用户B与用户C建立呼叫；
    - 如果用户B先释放，则用户C与座席A建立呼叫；
    - 如果用户C先释放，则用户B与座席A建立呼叫；
    - 如果座席A想要取回与用户B的呼叫，则调用[取消转移](#)接口取回呼叫。此时会挂断与用户C的通话，座席A与用户B进行通话。

图 3-5 转座席—成功转流程图



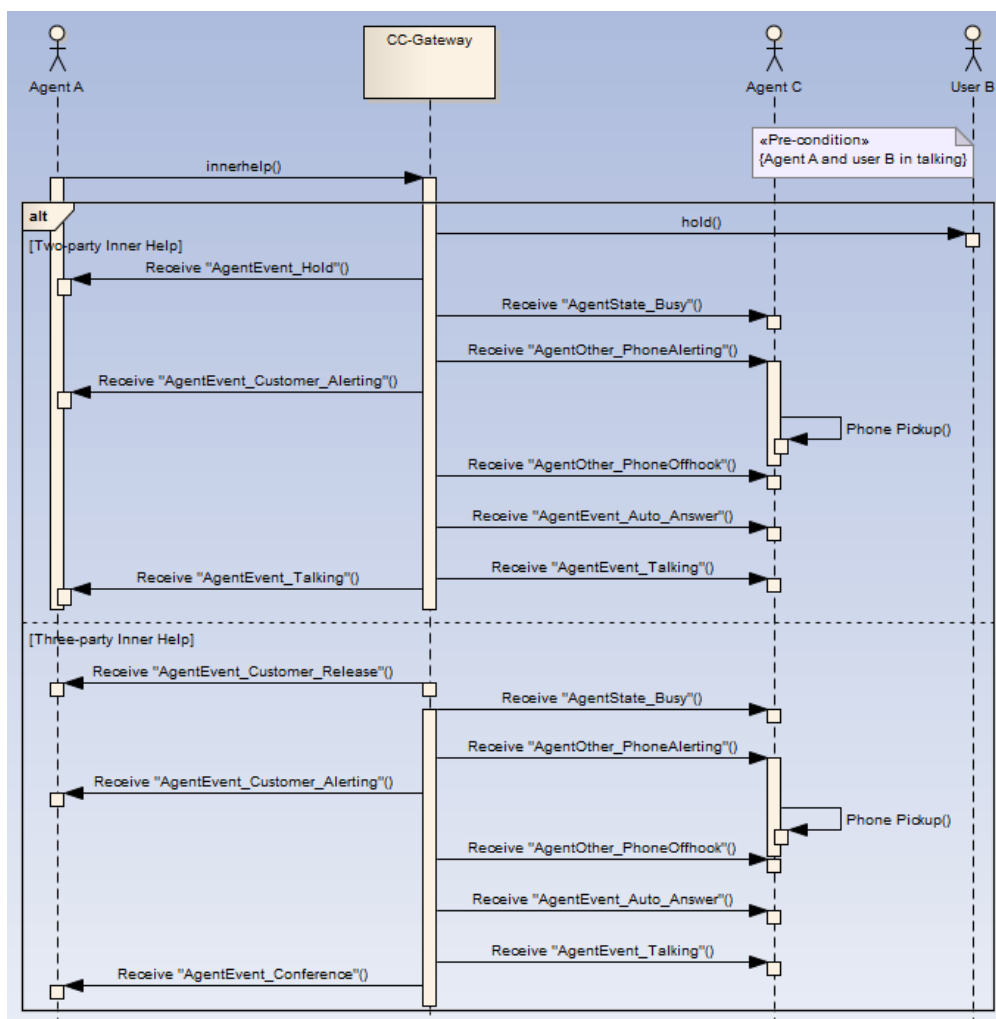
- 前提条件  
座席已经处于语音通话状态。
- 实现过程  
**呼叫转移**
- 触发事件  
对方振铃事件 ( AgentEvent\_Customer\_Alerting )  
座席挂断呼叫事件 ( AgentEvent\_Call\_Release )  
客户挂断呼叫事件 ( AgentEvent\_Customer\_Release )  
座席物理话机挂断事件 ( AgentOther\_PhoneRelease )  
示闲事件 ( AgentState\_Ready )  
转移模式为成功转时：成功转结果通知事件 ( AgentEvent\_SuccTransResult )  
转移模式为通话转或三方转时：三方通话成功事件 ( AgentEvent\_Conference )

### 3.4.4 内部求助

- 应用场景  
座席A已经与用户B建立通话，此时座席A需要向其他座席（例如座席C）求助，则采用内部求助。内部求助分为两方求助和三方求助。  
- 两方求助：被求助的座席C应答座席A的求助呼叫后，用户B的呼叫被保持，座席A与座席C进行通话。两方求助成功后：

- 如果座席C先释放通话后，则座席A与用户B恢复通话；
- 如果座席A先释放通话，则用户B的呼叫被转移给座席C处理；
- 如果座席A想要取回与用户B的呼叫，则根据座席A与座席C形成的呼叫的 callId，调用[拆除指定callid呼叫](#)接口取回呼叫。
- 三方求助：被求助的座席C应答座席A的求助呼叫后，座席A、用户B、座席C三方形成通话。三方求助成功后：
  - 如果座席A先释放通话，则用户B与座席C形成两方通话；
  - 如果座席C先释放通话，则用户B与座席A形成两方通话；
  - 如果用户B先释放通话，则座席A与座席C形成内部呼叫；
  - 如果座席A想要取回与用户B的通话，则调用[释放指定号码连接](#)接口(接口中输入的被释放号码为座席C的工号)取回呼叫。

图 3-6 内部求助流程图



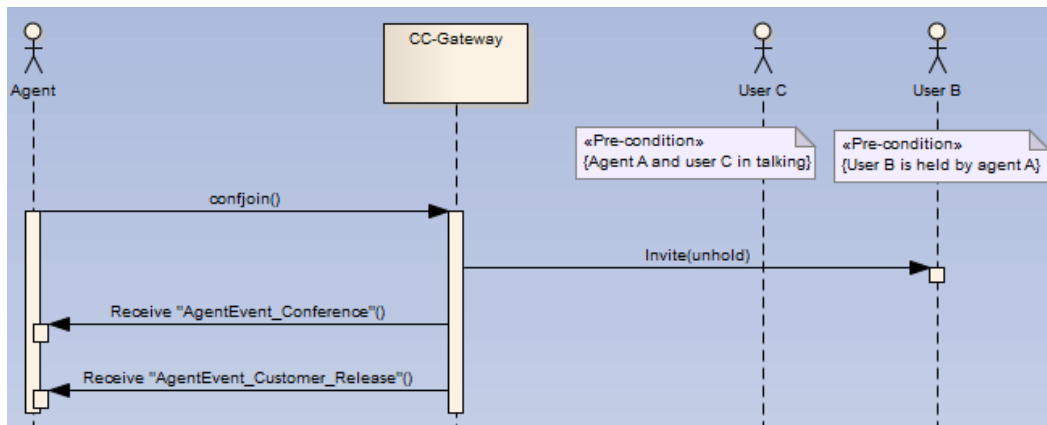
- 前提条件
  - 座席已经与客户正在通话中。

- 实现过程  
请参考[内部求助](#)
- 触发事件
  - 两方求助  
保持原有客户呼叫(AgentEvent\_Hold)  
被求助座席振铃(AgentEvent\_Customer\_Alerting)  
被求助座席接听来话(AgentEvent\_Talking)
  - 三方求助  
被求助座席振铃(AgentEvent\_Customer\_Alerting)  
释放与原客户的通话(AgentEvent\_Customer\_Release)  
被求助座席接听来话(AgentEvent\_Conference)

### 3.4.5 三方通话

- 应用场景  
座席A在保持用户B的呼叫后，外呼用户C，并且与用户C建立呼叫后，需要将保持的用户B加入到用户C与座席A的通话中，形成三方通话。三方通话成功后：
  - 如果座席A先释放呼叫，则用户B与用户C建立呼叫；
  - 如果用户B先释放呼叫，则座席A与用户C建立呼叫；
  - 如果用户C先释放呼叫，则座席A与用户B建立呼叫；
  - 如果座席A想要取回与用户B的呼叫，则调用[释放指定号码连接](#)接口(接口中输入的被释放号码为用户C的号码)取回呼叫。

图 3-7 三方通话流程图



- 前提条件
  - 座席存在被保持的呼叫。
  - 座席已经处于通话状态。
- 实现过程  
请参考[三方通话](#)
- 触发事件  
座席三方通话事件 ( AgentEvent\_Conference )

释放与原客户的通话 (AgentEvent\_Customer\_Release)

## 3.4.6 静音

### 3.4.6.1 开始静音

- 应用场景  
座席需要禁止客户听到座席的声音，但座席可以听到用户的声音。
- 前提条件  
座席存在一个与客户正在通话的呼叫。  
不支持内部呼叫、三方通话、三方求助的静音。
- 实现过程  
请参考[静音](#)
- 触发事件  
无

### 3.4.6.2 停止静音

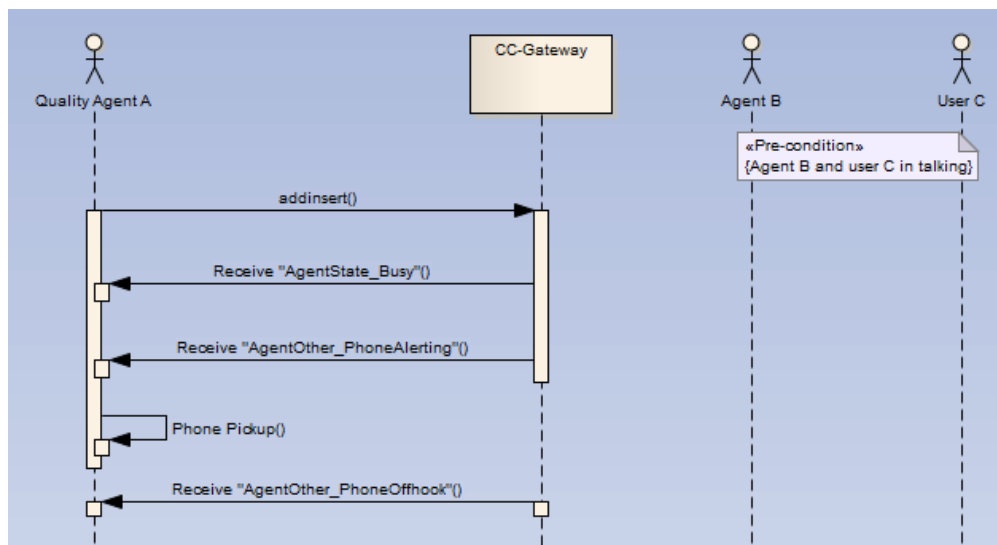
- 应用场景  
座席需要恢复客户能听到座席的声音。
- 前提条件  
座席存在一个与客户已静音的通话。
- 实现过程  
请参考[取消静音](#)
- 触发事件  
无

## 3.5 实时质检

### 3.5.1 插入

- 应用场景  
质检座席A签入后，插入指定座席B与客户C的语音通话中，进行实时质检，形成客户、座席和质检员在同一会场中的三方通话。

图 3-8 插入流程图

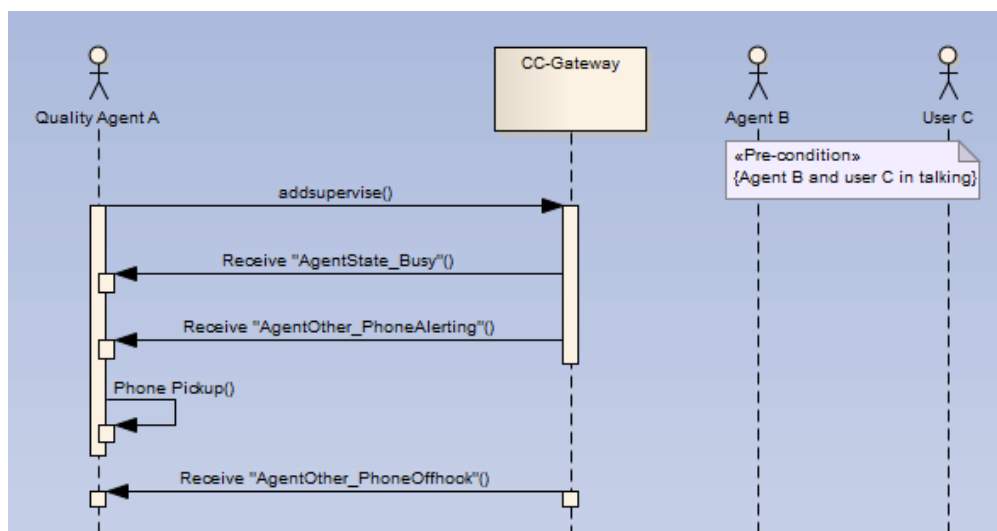


- 前提条件  
质检座席已签入。  
指定座席存在一个与客户正在通话的呼叫。
- 实现过程  
请参考[插入](#)。
- 触发事件  
质检座席收到事件：  
座席忙事件（AgentState\_Busy），表示座席被预占。
  - 非长通座席  
座席物理话机振铃事件（AgentOther\_PhoneAlerting）
  - 座席摘机后  
座席物理话机摘机事件（AgentOther\_PhoneOffhook）

### 3.5.2 侦听

- 应用场景  
质检座席A签入后，侦听指定座席B与客户C的语音通话，进行实时质检。质检座席A能够听到座席B和客户C之间的声音，但座席B和客户C听不到质检座席A的声音。

图 3-9 侦听流程图



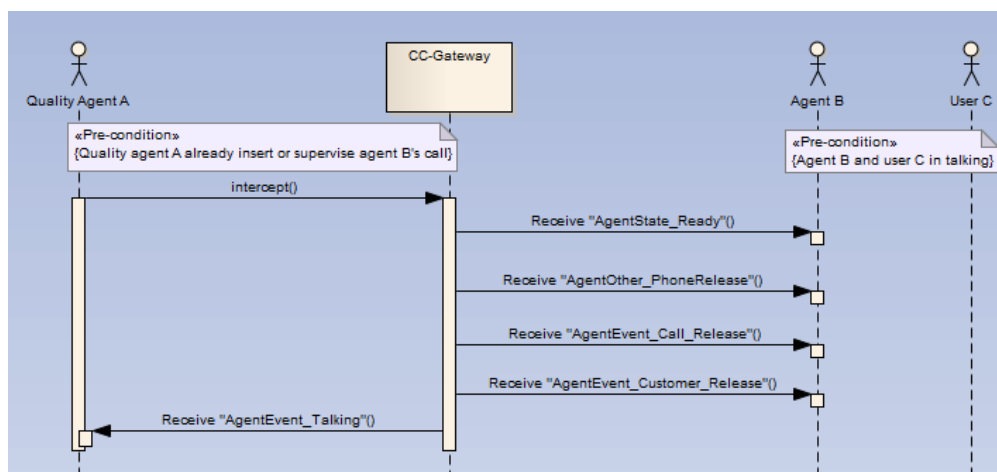
- 前提条件  
质检座席已签入。  
指定座席存在一个与客户正在通话的呼叫。
- 实现过程  
请参考[侦听](#)
- 触发事件  
质检座席收到事件：  
座席忙事件（AgentState\_Busy），表示座席被预占
  - 非长通座席  
座席物理话机振铃事件（AgentOther\_PhoneAlerting）
  - 座席摘机后  
座席物理话机摘机事件（AgentOther\_PhoneOffhook）

### 3.5.3 拦截

- 应用场景  
质检座席A签入后，拦截指定座席B与客户C的语音通话，直接和客户C进行语音通话，同时释放座席B。



图 3-10 拦截流程图

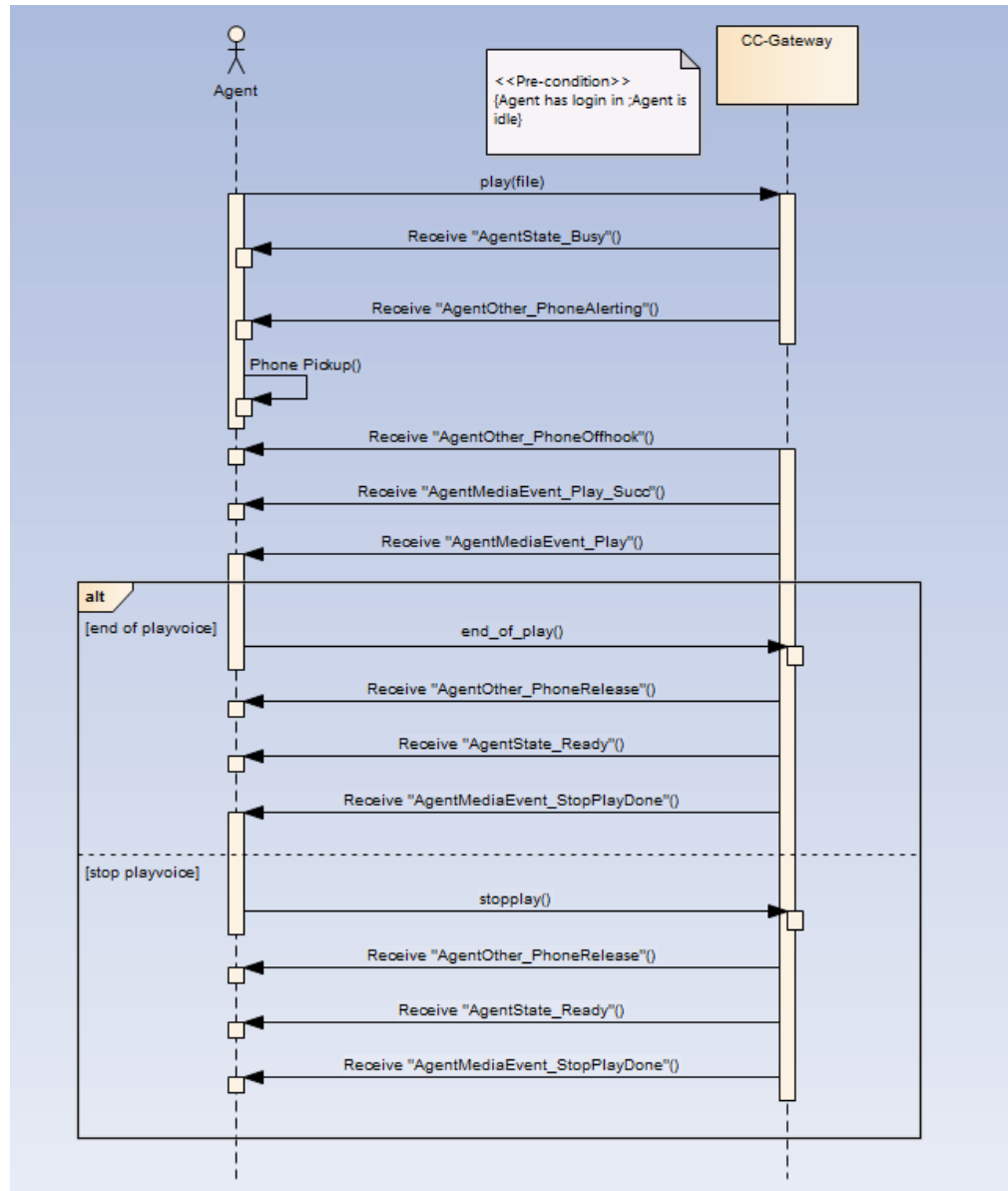


- 前提条件  
质检座席已签入。  
质检座席已对指定座席进行了插入或侦听或耳语。  
指定座席存在一个与客户正在通话的呼叫。
- 实现过程  
请参考[拦截](#)。
- 触发事件  
被质检座席收到事件：  
示闲事件（AgentState\_Ready）  
座席物理话机挂断事件（AgentOther\_PhoneRelease）  
座席挂断呼叫事件（AgentEvent\_Call\_Release）  
客户挂断呼叫事件（AgentEvent\_Customer\_Release）  
质检座席收到事件：  
座席通话事件（AgentEvent\_Talking）

## 3.6 放音

### 3.6.1 在非通话状态进行放音操作

- 应用场景  
座席需要播放录音。

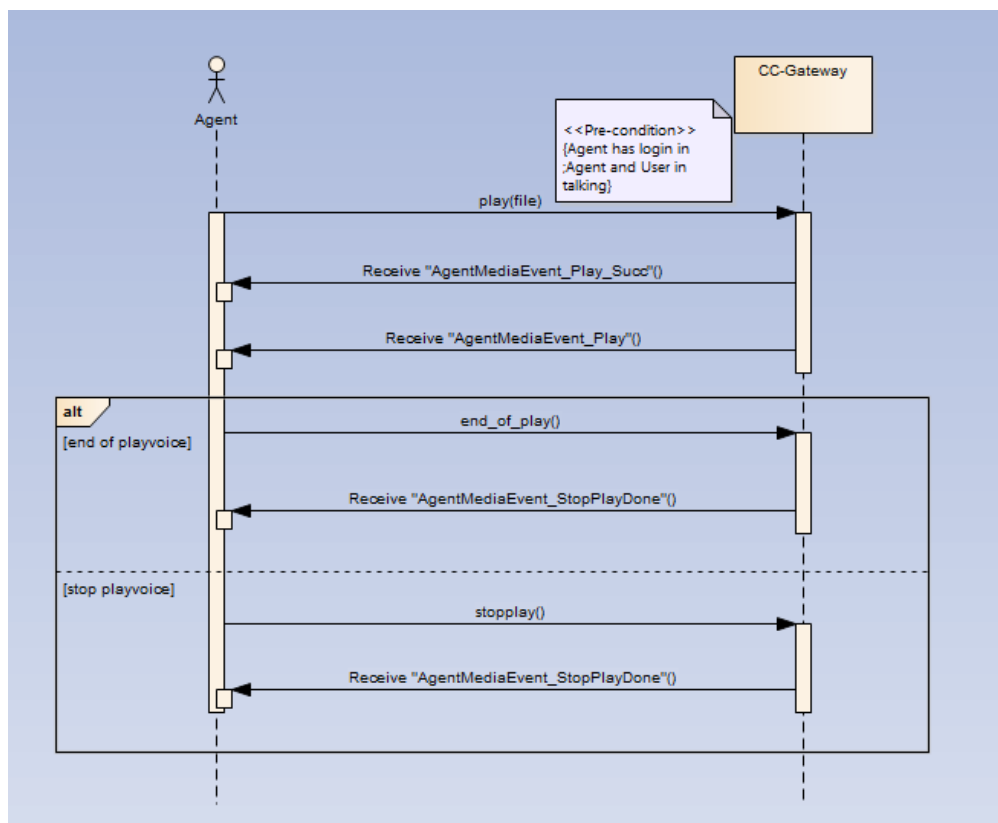


- 前提条件  
座席已签入  
座席不在通话中
- 实现过程  
请参考[录音回放:recordplay](#)
- 触发事件  
放音：
  - 座席忙事件 ( AgentState\_Busy )
  - 座席物理话机振铃事件 ( AgentOther\_PhoneAlerting )
  - 座席物理话机摘机事件 ( AgentOther\_PhoneOffhook )
  - 录音播放成功 ( AgentMediaEvent\_Play\_Succ )
  - 录音播放开始 ( AgentMediaEvent\_Play )放音失败：

- 座席忙事件 ( AgentState\_Busy )
  - 座席物理话机振铃事件 ( AgentOther\_PhoneAlerting )
  - 座席物理话机摘机事件 ( AgentOther\_PhoneOffhook )
  - 座席物理话机挂断事件 ( AgentOther\_PhoneRelease )
  - 示闲事件 ( AgentState\_Ready )
  - 录音播放失败 ( AgentMediaEvent\_Play\_Fail )
  - 停止录音播放 ( AgentMediaEvent\_StopPlayDone )
- 暂停放音:
- 暂停录音播放成功 ( AgentMediaEvent\_PausePlayDone )
- 恢复放音:
- 恢复录音播放成功 ( AgentMediaEvent\_ResumePlayDone )
- 放音快进:
- 录音播放快进或快退成功 ( AgentMediaEvent\_JumpPlayDone )
- 放音快退:
- 录音播放快进或快退成功 ( AgentMediaEvent\_JumpPlayDone )
- 停止放音或录音播放完成:
- 座席物理话机挂断事件 ( AgentOther\_PhoneRelease )
  - 示闲事件 ( AgentState\_Ready )
  - 停止录音播放成功 ( AgentMediaEvent\_StopPlayDone )

### 3.6.2 在通话态进行放音操作

- 应用场景  
座席需要播放录音。

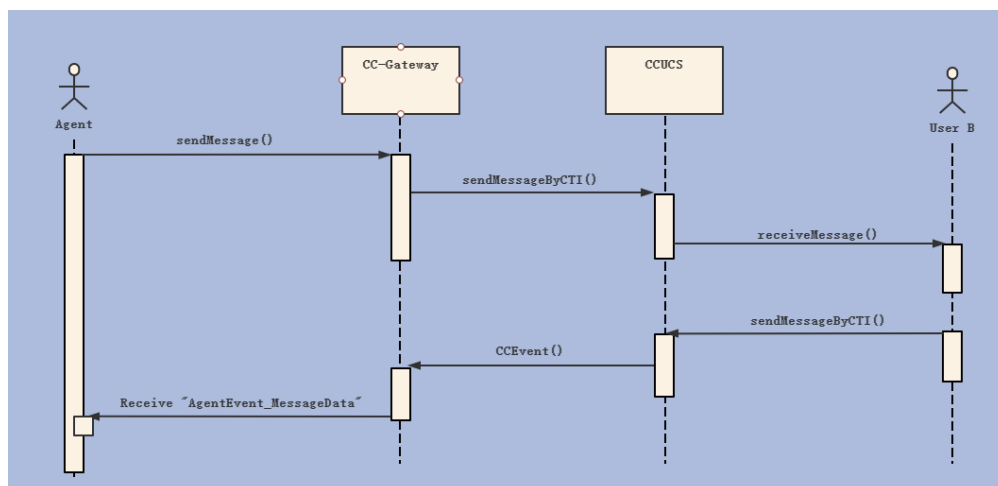


- 前提条件
  - 座席已签入
  - 座席在通话中
- 实现过程
  - 请参考[录音回放:recordplay](#)
- 触发事件
  - 放音：
    - 录音播放成功 ( AgentMediaEvent\_Play\_Succ )
    - 录音播放开始 ( AgentMediaEvent\_Play )
  - 放音失败：
    - 录音播放失败 ( AgentMediaEvent\_Play\_Fail )
    - 停止录音播放 ( AgentMediaEvent\_StopPlayDone )
  - 放音快进：
    - 录音播放快进或快退成功 ( AgentMediaEvent\_JumpPlayDone )
  - 放音快退：
    - 录音播放快进或快退成功 ( AgentMediaEvent\_JumpPlayDone )
  - 停止放音或录音播放完成：
    - 停止录音播放成功 ( AgentMediaEvent\_StopPlayDone )

## 3.7 基本多媒体呼叫

### 3.7.1 收发消息

- 应用场景
  - 座席与用户多媒体文字交谈。



- 前提条件
  - 座席已签入系统，签入平台多媒体服务器。
  - 平台有多媒体呼叫分配到此座席。
- 实现过程

座席发送消息请参考[发送消息:sendmessage](#)

- 接收消息事件

多媒体消息事件 ( AgentEvent\_MessageData )