隐私保护通话

开发指南

文档版本 3

发布日期 2026-03-12

华为云计算技术有限公司

商标声明


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云计算技术有限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，华为云计算技术有限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。

目 录

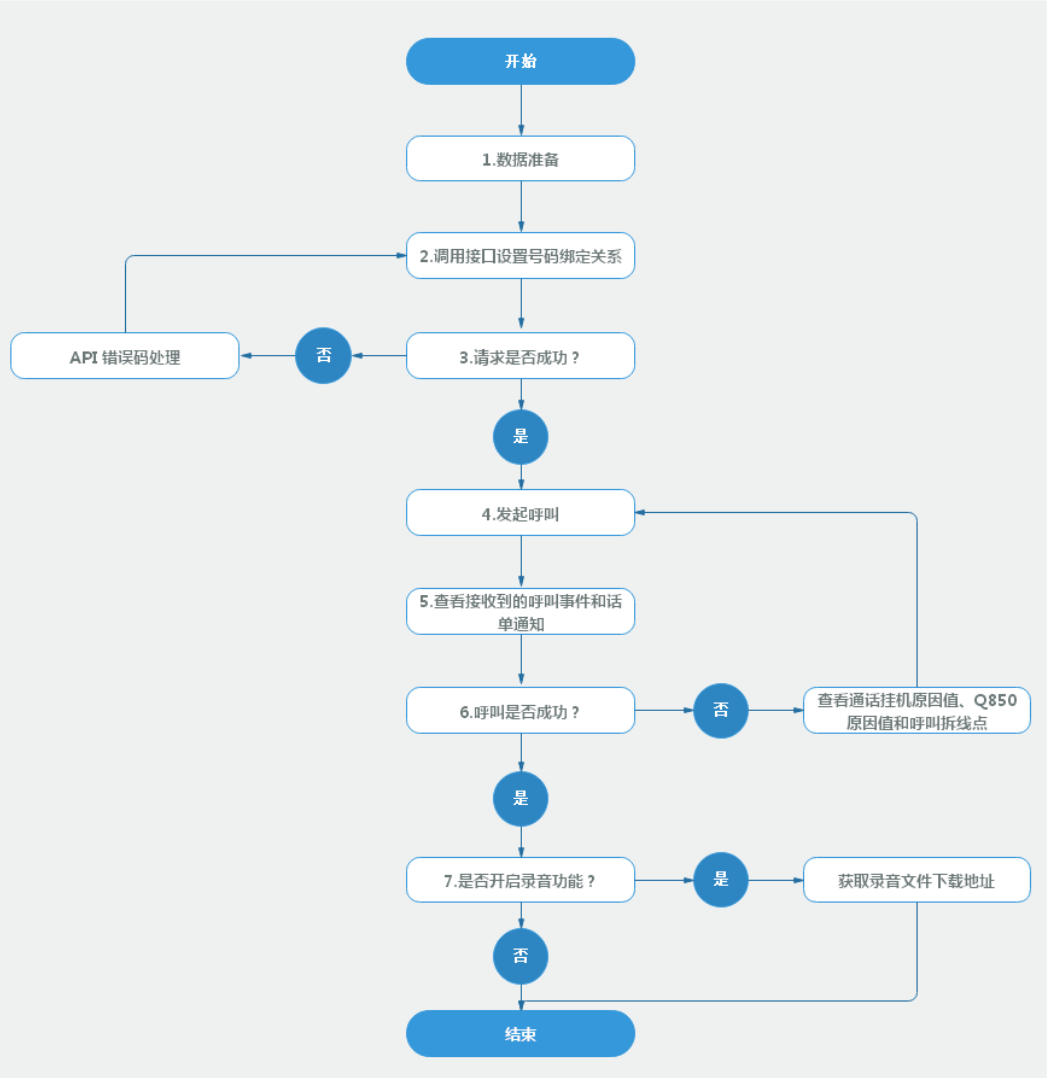
1 新手必读

业务流程

隐私保护通话二次开发业务整体流程如下：

说明

隐私保护通话各模式的二次开发业务流程不完全相同，以下流程以AXB模式为例。



1. 参考**开发准备**，获取调用隐私保护通话API的关联数据。
2. 参考**代码样例**调用AXB模式绑定接口，设置用户号码A、用户号码B和隐私号码X的绑定关系。
3. 根据请求响应消息，判断请求是否成功。
   - 请求成功 => **4**
   - 请求失败 => 参考**API错误码处理**，修正后重新执行**2**
4. 发起呼叫验证绑定关系：使用A或B号码直接呼叫X号码（**点击查看其他模式的呼叫验证方法**）。

   说明

   因运营商管控，固话号码只能接收来自X号码的呼叫，不能作为主叫呼叫X号码。若用户号码A为固话号码，请使用用户号码B（手机号码）呼叫X号码进行验证，反之亦然。
5. 查看接收到的呼叫事件通知和话单通知（若要接收呼叫事件通知和话单通知，需在**添加应用**时填写“呼叫状态接收地址”和“呼叫话单接收地址”）。
6. 根据呼叫是否接通（如A直接呼叫X，是否接通B），判断呼叫是否成功。
   - 呼叫成功 => **7**
   - 呼叫失败 => 参考**挂机原因值、Q850原因值、呼叫拆线点**分析呼叫未接通原因，如果重新验证呼叫，请重新执行**4**。

7. 是否开启了录音功能？（**点击查看如何开启录音功能**）
   - 是 =>**获取录音文件下载地址**=>结束
   - 否 => 结束

调测指引

隐私保护通话二次开发过程中，开发者需关注的业务调测点如下：

注：隐私保护通话各模式的二次开发业务流程不完全相同，以下流程以AXB模式为例。

- **Check 1**：发起AXB绑定请求前，对请求参数合法性做必要的检查，如：
  - 携带正确的APP接入地址。APP接入地址需登录隐私保护通话控制台，从“**应用管理**”页获取。
  - 携带的参数格式是否正确，无多余空格等。如：绑定接口携带的A号码需为全局号码格式（如：+86138****0021），其他参数详见**API参考文档**。
- **Check 2**：获取绑定请求结果时，请解析出响应结果码。若绑定失败，可参考**API错误码**中的处理建议进行修正。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
Content-Length: xx

{
"resultcode":"0",
```

"resultdesc":"Success" ,
"subscriptionId":"********",
"relationNum":"+867552****08",
"callDirection":0,
"duration":0,
"maxDuration":0
}

- **Check 3**：隐私保护通话平台会推送呼叫事件通知给客户服务器。若呼叫失败，请解析出挂机事件通知（disconnect）中的挂机原因值（stateCode）并参考**挂机原因值**排查失败原因。

  注：仅在**添加应用**时设置了呼叫状态接收地址时，隐私保护通话平台才会推送呼叫事件通知给客户服务器。

POST /status HTTP/1.1

{"eventType":"disconnect","statusInfo":
{"sessionId":"1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com","timestamp":"2019-01-23
09:16:41","caller":"+86138****0021","called":"+86138****7021",**"stateCode":0,"stateDesc":"The user releases the call.**","subscriptionId":"********"}}

- **Check 4**：通话结束后，隐私保护通话平台会推送话单通知给客户。若呼叫失败，请解析出转接呼叫操作失败的Q850原因值（fwdUnaswRsn）和通话失败的拆线点（ulFailReason），并参考**Q850原因值说明**和**呼叫拆线点说明**排查失败原因。

  注：仅在**添加应用**时设置了呼叫话单接收地址时，隐私保护通话平台才会推送话单通知给客户服务器。

POST /fee HTTP/1.1

{"eventType":"fee","feeLst":
[{"direction":1,"spId":"********","appKey":"************","icid":"ba171f34e6953fcd751edc77127748f4.3757223714.337238282.9","bindNum":"+86138****0022","sessionId":"1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com","subscriptionId":"********","callerNum":"+86138****0021","calleeNum":"+86138****0022","fwdDisplayNum":"+86138****0022","fwdDstNum":"+86138****7021","callInTime":"2019-01-23 09:15:14","fwdStartTime":"2019-01-23 09:15:15","fwdAlertingTime":"2019-01-23 09:15:21","fwdAnswerTime":"2019-01-23 09:15:36","callEndTime":"2019-01-23 09:16:41",**"fwdUnaswRsn":0,"ulFailReason":0,**"sipStatusCode":0,"callOutUnaswRsn":0,"recordFlag":1,"recordStartTime":"2019-01-23
09:15:37","recordDomain":"****.com","recordBucketName":"****","recordObjectName":"****.wav","ttsPlayTimes":0,"ttsTransDuration":0,"mptyId":"****","serviceType":"004","hostName":"callenabler246.huaweicaas.com"}]}

2 开发准备

启动开发前需要准备的数据如下：

注：X号码（隐私号码）、A号码和B号码的格式要求，请参考接口文档中的参数说明。

参数名	取值样例	获取方式	相关文档
APP_Key	a1**********	登录管理控制台，从“**应用管理**”页获取。 注：AXE模式、AXB模式和AX模式对应不同的应用，请按需获取。	**添加应用**
APP_Secret	cfc8**********		
APP接入地址	https://rtcpns.cn-north-1.myhuaweicloud.com		
X号码（隐私号码）	+86180****0001	登录管理控制台，根据“所属应用”从“**号码订购**”页下载号码表。	**订购号码**
区号（areaCode）	0755		
A号码	+86186****5678	AXE模式、AX模式和AXB模式下，A用户真实手机号或固定电话，请按需填写。 注：因运营商管控，当A号码为固话号码时，只能接收来自X号码的呼叫，不能作为主叫呼叫X号码。	-
B号码	+86186****5679	AXB模式下，B用户真实手机号或固定电话，请按需填写。 注：因运营商管控，当B号码为固话号码时，只能接收来自X号码的呼叫，不能作为主叫呼叫X号码。	-

参数名	取值样例	获取方式	相关文档
访问URI	/rest/provision/caas/privatenumber/v1.0	不同接口具有不同的URI，请从对应的接口文档中获取。	**AX模式绑定接口**

3 代码样例

3.1 Java 代码样例

环境要求

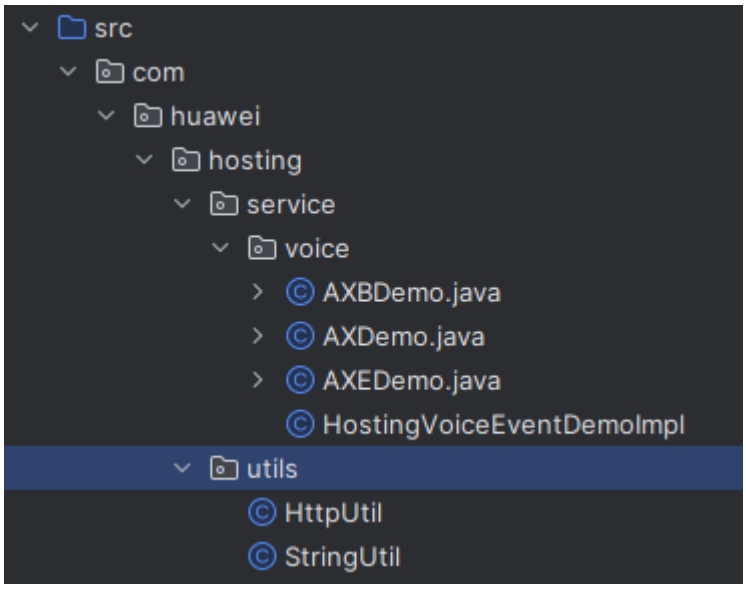
环境要求	基于JDK 1.8版本，要求JDK 1.6及以上版本。
引用库	**commons-io**、**commons-lang3**、**fastjson**、**log4j**、**sun.misc.base64decoder**

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

3.1.1 公共依赖

创建一个新的Java项目，并按照以下目录结构组织代码。



将以下代码复制到相应的目录中，保存后编译并运行项目。代码运行依赖于utils目录下的工具类文件，请确保这些文件存在于项目目录中。

HttpUtil

```
package com.huawei.hosting.utils;
import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;
import org.apache.log4j.Logger;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.net.URLEncoder;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Map;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
/**
 * HTTP协议传输工具类
 */
public class HttpUtil {
    private static Logger logger = Logger.getLogger(HttpUtil.class);
    private static final int HTTP_STATUS_OK = 200;
    /**
     * 向指定 URL发送POST方法的请求
     *
     * @param appKey
     * @param appSecret
     * @param url
     * @param jsonBody
     * @return
     */
    public static String sendPost(String appKey, String appSecret, String url, String jsonBody) {
        DataOutputStream out = null;
        BufferedReader in = null;
        StringBuffer result = new StringBuffer();
        HttpsURLConnection connection = null;
        InputStream is = null;
```



```
    HostnameVerifier hv = new HostnameVerifier() {
        @Override
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    };
    try {
        trustAllHttpsCertificates();
    } catch (Exception e1) {
        e1.printStackTrace();
    }
    try {
        URL realUrl = new URL(url);
        connection = (HttpsURLConnection) realUrl.openConnection();
        connection.setHostnameVerifier(hv);
        connection.setDoOutput(true);
        connection.setDoInput(true);
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Accept", "application/json");
        connection.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
        connection.setRequestProperty("Authorization",
                "AKSK realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"");
        connection.setRequestProperty("X-AKSK", StringUtil.buildAKSKHeader(appKey, appSecret));
        logger.info("RequestBody is : " + jsonBody);
        connection.connect();
        out = new DataOutputStream(connection.getOutputStream());
        out.writeBytes(jsonBody);
        out.flush();
        out.close();
        int status = connection.getResponseCode();
        if (HTTP_STATUS_OK == status) {
            is = connection.getInputStream();
        } else {
            is = connection.getErrorStream();
        }
        in = new BufferedReader(new InputStreamReader(is, "UTF-8"));
        String line = "";
        while ((line = in.readLine()) != null) {
            result.append(line);
        }
    } catch (Exception e) {
        logger.info("Send Post request catch exception: " + e.toString());
    }
    finally {
        IOUtils.closeQuietly(out);
        IOUtils.closeQuietly(is);
        IOUtils.closeQuietly(in);
        if (null != connection) {
            IOUtils.close(connection);
        }
    }
    return result.toString();
}
/**
 * 向指定 URL发送PUT方法的请求
 *
 * @param appKey
 * @param appSecret
 * @param url
 * @param jsonBody
 * @return
 */
public static String sendPut(String appKey, String appSecret, String url, String jsonBody) {
    DataOutputStream out = null;
    BufferedReader in = null;
    StringBuffer result = new StringBuffer();
    HttpsURLConnection connection = null;
    InputStream is = null;
    HostnameVerifier hv = new HostnameVerifier() {
```

```
        @Override
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    };
    try {
        trustAllHttpsCertificates();
    } catch (Exception e1) {
        e1.printStackTrace();
    }
    try {
        URL realUrl = new URL(url);
        connection = (HttpsURLConnection) realUrl.openConnection();
        connection.setHostnameVerifier(hv);
        connection.setDoOutput(true);
        connection.setDoInput(true);
        connection.setRequestMethod("PUT");
        connection.setRequestProperty("Accept", "application/json");
        connection.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
        connection.setRequestProperty("Authorization",
                "AKSK realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"");
        connection.setRequestProperty("X-AKSK", StringUtil.buildAKSKHeader(appKey, appSecret));
        logger.info("RequestBody is : " + jsonBody);
        connection.connect();
        out = new DataOutputStream(connection.getOutputStream());
        out.writeBytes(jsonBody);
        out.flush();
        out.close();
        int status = connection.getResponseCode();
        if (HTTP_STATUS_OK == status) {
            is = connection.getInputStream();
        } else {
            is = connection.getErrorStream();
        }
        in = new BufferedReader(new InputStreamReader(is));
        String line;
        while ((line = in.readLine()) != null) {
            result.append(line);
        }
    } catch (Exception e) {
        logger.info("Send Put request catch exception: " + e.toString());
        e.printStackTrace();
    }
    finally {
        IOUtils.closeQuietly(out);
        IOUtils.closeQuietly(is);
        IOUtils.closeQuietly(in);
        if (null != connection) {
            IOUtils.close(connection);
        }
    }
    return result.toString();
}
/**
 * 向指定 URL发送DELETE方法的请求
 *
 * @param appKey
 * @param appSecret
 * @param url
 * @param params
 * @return
 */
public static String sendDelete(String appKey, String appSecret, String url, String params) {
    BufferedReader in = null;
    StringBuffer result = new StringBuffer();
    HttpsURLConnection connection = null;
    InputStream is = null;
    HostnameVerifier hv = new HostnameVerifier() {
        @Override
```

```
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    };
    try {
        trustAllHttpsCertificates();
    } catch (Exception e1) {
        e1.printStackTrace();
    }
    try {
        String realPath = url + (StringUtils.isEmpty(params) ? "" : "?" + params);
        URL realUrl = new URL(realPath);
        connection = (HttpsURLConnection) realUrl.openConnection();
        connection.setHostnameVerifier(hv);
        connection.setDoInput(true);
        connection.setRequestMethod("DELETE");
        connection.setRequestProperty("Accept", "application/json");
        connection.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
        connection.setRequestProperty("Authorization",
                "AKSK realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"");
        connection.setRequestProperty("X-AKSK", StringUtil.buildAKSKHeader(appKey, appSecret));
        logger.info("RequestBody is : " + params);
        connection.connect();
        int status = connection.getResponseCode();
        if (HTTP_STATUS_OK == status) {
            is = connection.getInputStream();
        } else {
            is = connection.getErrorStream();
        }
        in = new BufferedReader(new InputStreamReader(is));
        String line;
        while ((line = in.readLine()) != null) {
            result.append(line);
        }
    } catch (Exception e) {
        logger.info("Send DELETE request catch exception: " + e.toString());
    }
    finally {
        IOUtils.closeQuietly(is);
        IOUtils.closeQuietly(in);
        if (null != connection) {
            IOUtils.close(connection);
        }
    }
    return result.toString();
}
/**
 * 向指定 URL发送GET方法的请求
 *
 * @param appKey
 * @param appSecret
 * @param url
 * @param params
 * @return
 */
public static String sendGet(String appKey, String appSecret, String url, String params) {
    BufferedReader in = null;
    StringBuffer result = new StringBuffer();
    HttpsURLConnection connection = null;
    InputStream is = null;
    HostnameVerifier hv = new HostnameVerifier() {
        @Override
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    };
    try {
        trustAllHttpsCertificates();
    } catch (Exception e1) {
```

```
            e1.printStackTrace();
        }
        try {
            String realPath = url + (StringUtils.isEmpty(params) ? "" : "?" + params);
            URL realUrl = new URL(realPath);
            connection = (HttpsURLConnection) realUrl.openConnection();
            connection.setHostnameVerifier(hv);
            connection.setDoInput(true);
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Accept", "application/json");
            connection.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
            connection.setRequestProperty("Authorization",
                    "AKSK realm=\"SDP\",profile=\"UsernameToken\",type=\"Appkey\"");
            connection.setRequestProperty("X-AKSK", StringUtil.buildAKSKHeader(appKey, appSecret));
            connection.setInstanceFollowRedirects(false); //设置本次连接不自动处理重定向
            logger.info("RequestBody is : " + params);
            connection.connect();
            int status = connection.getResponseCode();
            if (301 == status) { //获取录音文件下载地址
                return connection.getHeaderField("Location");
            }else if (HTTP_STATUS_OK == status) { //查询绑定信息
                is = connection.getInputStream();
            } else { //获取错误码
                is = connection.getErrorStream();
            }
            in = new BufferedReader(new InputStreamReader(is));
            String line;
            while ((line = in.readLine()) != null) {
                result.append(line);
            }
        } catch (Exception e) {
            logger.info("Send GET request catch exception: " + e.toString());
        }
        finally {
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(in);
            if (null != connection) {
                IOUtils.close(connection);
            }
        }
        return result.toString();
    }
    /**
     * 键值对转查询url
     *
     * @param map
     * @return
     */
    public static String map2UrlEncodeString(Map<String, Object> map) {
        if(null == map || map.isEmpty()) {
            return "";
        }
        StringBuilder sb = new StringBuilder();
        String temp = "";
        for (String s : map.keySet()) {
            try {
                temp = URLEncoder.encode(String.valueOf(map.get(s)), "UTF-8");
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
            sb.append(s).append("=").append(temp).append("&");
        }
        return sb.deleteCharAt(sb.length() - 1).toString();
    }
    /**
     * 忽略SSL证书校验
     *
     * @throws Exception
     */
```

```
    static void trustAllHttpsCertificates() throws Exception {
        TrustManager[] trustAllCerts = new TrustManager[] { new X509TrustManager() {
            public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException
{
                return;
            }
            public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
                return;
            }
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
        } };
        SSLContext sc = SSLContext.getInstance("SSL");
        sc.init(null, trustAllCerts, null);
        HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
    }
}
```

StringUtil

```
/*
 * Copyright Notice:
 *      Copyright  1998-2008, Huawei Technologies Co., Ltd.  ALL Rights Reserved.
 *
 *      Warning: This computer software sourcecode is protected by copyright law
 *      and international treaties. Unauthorized reproduction or distribution
 *      of this sourcecode, or any portion of it, may result in severe civil and
 *      criminal penalties, and will be prosecuted to the maximum extent
 *      possible under the law.
 */
package com.huawei.hosting.utils;
//import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Calendar;
import java.util.Locale;
import java.util.TimeZone;
import java.util.UUID;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
public class StringUtil {
    public static final String AKSK_HEADER_FORMAT = "UsernameToken Username=\"%s\",PasswordDigest=
\"%s\",Nonce=\"%s\",Created=\"%s\"";
    public static boolean strIsNullOrEmpty(String s) {
        return (null == s || s.trim().length() < 1);
    }
    public static String buildAKSKHeader(String appKey, String appSecret) throws Exception {
        if (StringUtil.strIsNullOrEmpty(appKey) || StringUtil.strIsNullOrEmpty(appSecret)) {
            return null;
        }
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
        format.setTimeZone(TimeZone.getTimeZone("UTC"));
        Calendar calendar = Calendar.getInstance();
        String time = format.format(calendar.getTime());
        String stNonce = UUID.randomUUID().toString().replace("-", "").toUpperCase(Locale.ROOT);
        String str = stNonce + time;
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(new SecretKeySpec(appSecret.getBytes(StandardCharsets.UTF_8), "HmacSHA256"));
        byte[] authBytes = mac.doFinal(str.getBytes(StandardCharsets.UTF_8));
        String passwordDigestBase64Str = encodeBase64(authBytes);
        return String.format(AKSK_HEADER_FORMAT, appKey, passwordDigestBase64Str, stNonce, time);
    }
    private static String encodeBase64(byte[] bytes) {
        if (bytes.length == 0) {
            return null;
        } else {
```

```
            return new String(Base64.getEncoder().encode(bytes), StandardCharsets.UTF_8);
        }
    }
}
```

3.1.2 AXB 模式

AXB模式代码样例：

```
package com.huawei.hosting.service.voice;
import com.huawei.hosting.utils.HttpUtil;
import com.alibaba.fastjson.JSONObject;
import org.apache.commons.lang3.StringUtils;
import org.apache.log4j.Logger;
import java.util.HashMap;
import java.util.Map;
/**
 * AXB模式(具体场景请参考接口说明)
 */
public class AXBDemo {
    /**
     * 必填,请登录管理控制台,从"应用管理"页获取
     */
    private final static String OMPAPPKEY = "a1d1f50cad21415fbdd13d8f53d36d60"; // APP_Key
    private final static String OMPAPPSECRET = "cfc881cc704c4fba8d8fef5788e03e6b"; // APP_Secret
    private final static String OMPDOMAINNAME = "https://rtcpns.cn-north-1.myhuaweicloud.com"; // APP接
入地址
    public static void main(String[] args) {
        IAXBInterfaceDemo axb = new AXBInterfaceDemoImpl(OMPAPPKEY, OMPAPPSECRET,
OMPDOMAINNAME);
        // 第一步: 号码绑定,即调用AXB模式绑定接口
        //         axb.axbBindNumber("+8618010000001", "+8618612345678", "+8618612345679");
        // 当用户发起通话时,隐私保护通话平台会将呼叫事件推送到商户应用,参考: HostingVoiceEventDemoImpl
        // 当用户使用短信功能,隐私保护通话平台将短信事件推送到商户应用,参考: HostingVoiceEventDemoImpl
        // 第二步: 用户通话结束,若设置录音,则商户可以获取录音文件下载地址,即调用获取录音文件下载地址接口
        //         axb.axbGetRecordDownloadLink("ostor.huawei.com", "1200_366_0_20161228102743.wav");
        // 第三步: 根据业务需求,可更改绑定关系,即调用AXB模式绑定信息修改接口
        //         axb.axbModifyNumber("efw89efwf7fea324252", "+8618612345679", null);
        // 第四步: 隐私号码循环使用,商户可将绑定关系解绑,即调用AXB模式解绑接口
        //         axb.axbUnbindNumber(null, "+8618010000001");
        // 第五步: 商户可查询已订购的隐私号码的绑定信息,即调用AXB模式绑定信息查询接口
        axb.axbQueryBindRelation(null, "+8618010000001");
    }
}
interface IAXBInterfaceDemo {
    /**
     * Set X number to be the privacy number between number a and number b |
     * 隐私号码AXB绑定
     *
     * @param relationNum 关系号码
     * @param callerNum   主叫号码
     * @param calleeNum   被叫号码
     */
    void axbBindNumber(String relationNum, String callerNum, String calleeNum);
    /**
     * Modify number a/b of the privacy relationship assigned by subscriptionId |
     * 隐私号码AXB绑定信息修改
     *
     * @param subscriptionId 绑定关系ID
     * @param callerNum      主叫号码
     * @param calleeNum      被叫号码
     */
    void axbModifyNumber(String subscriptionId, String callerNum, String calleeNum);
    /**
     * Unbind the privacy relationship between number a and number b | 隐私号码AXB解绑
     *
     * @param subscriptionId 绑定关系ID
     * @param relationNum    关系号码 都传时以subscriptionId优先
```

```
     */
    void axbUnbindNumber(String subscriptionId, String relationNum);
    /**
     * Query the privacy binding relationship on the X number | 查询AXB绑定信息
     *
     * @param subscriptionId 绑定关系ID
     * @param relationNum    关系号码 都传时以subscriptionId优先
     */
    void axbQueryBindRelation(String subscriptionId, String relationNum);
    /**
     * Get download link of the record file created in call | 获取录音文件下载地址
     *
     * @param recordDomain 录音文件存储的服务器域名
     * @param fileName     录音文件名
     */
    void axbGetRecordDownloadLink(String recordDomain, String fileName);
}
/**
 * AXB模式接口测试
 */
class AXBInterfaceDemoImpl implements IAXBInterfaceDemo {
    private Logger logger = Logger.getLogger(AXBInterfaceDemoImpl.class);
    private String appKey; // APP_Key
    private String appSecret; // APP_Secret
    private String ompDomainName; // APP接入地址
    public AXBInterfaceDemoImpl(String appKey, String appSecret, String ompDomainName) {
        this.appKey = appKey;
        this.appSecret = appSecret;
        this.ompDomainName = ompDomainName;
    }
    /**
     * Build the real url of https request | 构建隐私保护通话平台请求路径
     *
     * @param path 接口访问URI
     * @return
     */
    private String buildOmpUrl(String path) {
        return ompDomainName + path;
    }
    @Override
    public void axbBindNumber(String relationNum, String callerNum, String calleeNum) {
        if (StringUtils.isBlank(relationNum) || StringUtils.isBlank(callerNum) || StringUtils.isBlank(calleeNum)) {
            logger.info("axbBindNumber set params error");
            return;
        }
        // 必填,AXB模式绑定接口访问URI
        String url = "/rest/caas/relationnumber/partners/v1.0";
        String realUrl = buildOmpUrl(url);
        // 封装JOSN请求
        JSONObject json = new JSONObject();
        json.put("relationNum", relationNum); // X号码(关系号码)
        json.put("callerNum", callerNum); // A方真实号码(手机或固话)
        json.put("calleeNum", calleeNum); // B方真实号码(手机或固话)
        /**
         * 选填,各参数要求请参考"AXB模式绑定接口"
         */
        //        json.put("areaCode", "0755"); //城市码
        //        json.put("areaMatchMode", "1"); //号码筛选方式
        //        json.put("callDirection", 0); //允许呼叫的方向
        //        json.put("duration", 86400); //绑定关系保持时间
        //        json.put("recordFlag", false); //是否通话录音
        //        json.put("recordHintTone", "recordHintTone.wav"); //录音提示音
        //        json.put("maxDuration", 60); //单次通话最长时间
        //        json.put("privateSms", true); //是否支持短信功能
        //        JSONObject preVoice = new JSONObject();
        //        preVoice.put("callerHintTone", "callerHintTone.wav"); //设置A拨打X号码时的通话前等待音
        //        preVoice.put("calleeHintTone", "calleeHintTone.wav"); //设置B拨打X号码时的通话前等待音
        //        json.put("preVoice", preVoice); //个性化通话前等待音
        String result = HttpUtil.sendPost(appKey, appSecret, realUrl, json.toString());
```

```
    logger.info("Response is :" + result);
}
@Override
public void axbModifyNumber(String subscriptionId, String callerNum, String calleeNum) {
    if (StringUtils.isBlank(subscriptionId)) {
        logger.info("axbModifyNumber set params error");
        return;
    }
    // 必填,AXB模式绑定信息修改接口访问URI
    String url = "/rest/caas/relationnumber/partners/v1.0";
    String realUrl = buildOmpUrl(url);
    // 封装JOSN请求
    JSONObject json = new JSONObject();
    json.put("subscriptionId", subscriptionId); // 绑定关系ID
    if (StringUtils.isNotBlank(callerNum)) {
        json.put("callerNum", callerNum); // 将A方修改为新的号码(手机或固话)
    }
    if (StringUtils.isNotBlank(calleeNum)) {
        json.put("calleeNum", calleeNum); // 将B方修改为新的号码(手机或固话)
    }
    /**
     * 选填,各参数要求请参考"AXB模式绑定信息修改接口"
     */
    //        json.put("callDirection", 0); //允许呼叫的方向
    //        json.put("duration", 86400); //绑定关系保持时间
    //        json.put("maxDuration", 90); //单次通话最长时间
    //        json.put("privateSms", true); //是否支持短信功能
    //        json.put("recordFlag", false); //是否通话录音
    //        JSONObject preVoice = new JSONObject();
    //        preVoice.put("callerHintTone", "callerHintTone.wav"); //设置A拨打X号码时的通话前等待音
    //        preVoice.put("calleeHintTone", "calleeHintTone.wav"); //设置B拨打X号码时的通话前等待音
    //        json.put("preVoice", preVoice); //个性化通话前等待音
    String result = HttpUtil.sendPut(appKey, appSecret, realUrl, json.toString());
    logger.info("Response is :" + result);
}
@Override
public void axbUnbindNumber(String subscriptionId, String relationNum) {
    if (StringUtils.isBlank(subscriptionId) && StringUtils.isBlank(relationNum)) {
        logger.info("axbUnbindNumber set params error");
        return;
    }
    // 必填,AXB模式解绑接口访问URI
    String url = "/rest/caas/relationnumber/partners/v1.0";
    String realUrl = buildOmpUrl(url);
    // 申明对象
    Map<String, Object> map = new HashMap<String, Object>();
    if (StringUtils.isNotBlank(subscriptionId)) {
        map.put("subscriptionId", subscriptionId); // 绑定关系ID
    } else {
        map.put("relationNum", relationNum); // X号码(关系号码)
    }
    String result = HttpUtil.sendDelete(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
    logger.info("Response is :" + result);
}
@Override
public void axbQueryBindRelation(String subscriptionId, String relationNum) {
    if (StringUtils.isBlank(subscriptionId) && StringUtils.isBlank(relationNum)) {
        logger.info("axbQueryBindRelation set params error");
        return;
    }
    // 必填,AXB模式绑定信息查询接口访问URI
    String url = "/rest/caas/relationnumber/partners/v1.0";
    String realUrl = buildOmpUrl(url);
    // 申明对象
    Map<String, Object> map = new HashMap<String, Object>();
    if (StringUtils.isNotBlank(subscriptionId)) {
        map.put("subscriptionId", subscriptionId); // 绑定关系ID
    } else {
        map.put("relationNum", relationNum); // X号码(关系号码)
```

```
            /**
             * 选填,各参数要求请参考"AXB模式绑定信息查询接口"
             */
            //            map.put("pageIndex", 1); //查询的分页索引,从1开始编号
            //            map.put("pageSize", 20); //查询的分页大小,即每次查询返回多少条数据
        }
        String result = HttpUtil.sendGet(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
        logger.info("Response is :" + result);
    }
    @Override
    public void axbGetRecordDownloadLink(String recordDomain, String fileName) {
        if (StringUtils.isBlank(recordDomain) || StringUtils.isBlank(fileName)) {
            logger.info("axbGetRecordDownloadLink set params error");
            return;
        }
        // 必填,AXB模式获取录音文件下载地址接口访问URI
        String url = "/rest/provision/voice/record/v1.0";
        String realUrl = buildOmpUrl(url);
        // 申明对象
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("recordDomain", recordDomain); // 录音文件存储的服务器域名
        map.put("fileName", fileName); // 录音文件名
        String result = HttpUtil.sendGet(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
        logger.info("Response is :" + result);
    }
}
```

3.1.3 AX 模式

AX模式代码样例：

```
package com.huawei.hosting.service.voice;
import com.huawei.hosting.utils.HttpUtil;
import com.alibaba.fastjson.JSONObject;
import org.apache.commons.lang3.StringUtils;
import org.apache.log4j.Logger;
import java.util.HashMap;
import java.util.Map;
/**
 * AX模式(具体场景请参考接口说明)
 */
public class AXDemo {
    /**
     * 必填,请登录管理控制台,从"应用管理"页获取
     */
    private final static String OMPAPPKEY = "a1d1f50cad21415fbdd13d8f53d36d60"; // APP_Key
    private final static String OMPAPPSECRET = "cfc881cc704c4fba8d8fef5788e03e6b"; // APP_Secret
    private final static String OMPDOMAINNAME = "https://rtcpns.cn-north-1.myhuaweicloud.com"; // APP接
入地址
    public static void main(String[] args) {
        IAXInterfaceDemo ax = new AXInterfaceDemoImpl(OMPAPPKEY, OMPAPPSECRET,
OMPDOMAINNAME);
        // 第一步: 号码绑定,即调用AX模式绑定接口
//        ax.axBindNumber("+8618612345678", "+8618010000001", "0");
        // 当用户发起通话时,隐私保护通话平台会将呼叫事件推送到商户应用,参考: HostingVoiceEventDemoImpl
        // 当用户使用短信功能,隐私保护通话平台将短信事件推送到商户应用,参考: HostingVoiceEventDemoImpl
        // 第二步: 用户通话结束,若设置录音,则商户可以获取录音文件下载地址,即调用获取录音文件下载地址接口
//        ax.axGetRecordDownloadLink("ostor.huawei.com", "1200_366_0_20161228102743.wav");
        // 第三步: 根据业务需求,可更改绑定关系,即调用AX模式绑定信息修改接口
//        ax.axModifyNumber("efw89efwf7fea324252", null, null, true);
        // 第四步: 根据业务需求,设置临时被叫,即调用AX模式设置临时被叫接口
//        ax.axSetCalleeNumber(null, "+8618010000001", "+8618612345679");
        // 第五步: 隐私号码循环使用,商户可将绑定关系解绑,即调用AX模式解绑接口
//        ax.axUnbindNumber(null, "+8618010000001");
        // 第六步: 商户可查询已订购的隐私号码的绑定信息,即调用AX模式绑定信息查询接口
        ax.axQueryBindRelation(null, "+8618612345678", null);
    }
```

```
}
/**
 * AX模式接口
 */
interface IAXInterfaceDemo {
    /**
     * Set the X number to the user's privacy number | 隐私号码AX绑定
     *
     * @param origNum          用户号码
     * @param privateNum       隐私号码
     * @param calleeNumDisplay 是否显示用户号码
     */
    void axBindNumber(String origNum, String privateNum, String calleeNumDisplay);
    /**
     * Modify sms function of the user's privacy number | 隐私号码AX绑定信息修改
     *
     * @param subscriptionId 绑定关系ID
     * @param origNum        用户号码,如果需要修改绑定关系中的用户号码,请指定新的用户号码.不携带时表示
不修改该参数值
     * @param privateNum     隐私号码
     * @param privateSms     是否支持短信
     *                       subscriptionId和privateNum二选一即可,当都传入时,优先选用subscriptionId
     */
    void axModifyNumber(String subscriptionId, String origNum, String privateNum, boolean privateSms);
    /**
     * Unbind the privacy number from number a | 隐私号码AX解绑
     *
     * @param subscriptionId 绑定关系ID
     * @param privateNum     隐私号码
     *                       subscriptionId和privateNum二选一即可,当都传入时,优先选用subscriptionId
     */
    void axUnbindNumber(String subscriptionId, String privateNum);
    /**
     * Query the privacy binding numbers on the X number | 查询AX绑定信息
     *
     * @param subscriptionId 绑定关系ID
     * @param origNum        用户号码
     * @param privateNum     隐私号码
     *                       subscriptionId,origNum和privateNum三选一即可,当都传入时,subscriptionId > origNum >
privateNum
     */
    void axQueryBindRelation(String subscriptionId, String origNum, String privateNum);
    /**
     * Set the callee number to the ax bind relation | 设置AX临时被叫
     *
     * @param subscriptionId 绑定关系ID
     * @param privateNum     隐私号码
     * @param calleeNum      被叫号码
     *                       subscriptionId和privateNum二选一即可,当都传入时,优先选用subscriptionId
     */
    void axSetCalleeNumber(String subscriptionId, String privateNum, String calleeNum);
    /**
     * Get download link of the record file created in call | 获取录音文件下载地址
     *
     * @param recordDomain 录音文件存储的服务器域名
     * @param fileName     录音文件名
     */
    void axGetRecordDownloadLink(String recordDomain, String fileName);
}
/**
 * AX模式接口测试
 */
class AXInterfaceDemoImpl implements IAXInterfaceDemo {
    private Logger logger = Logger.getLogger(AXInterfaceDemoImpl.class);
    private String appKey; // APP_Key
    private String appSecret; // APP_Secret
    private String ompDomainName; // APP接入地址
    public AXInterfaceDemoImpl(String appKey, String appSecret, String ompDomainName) {
        this.appKey = appKey;
```



```
        this.appSecret = appSecret;
        this.ompDomainName = ompDomainName;
    }
    /**
     * Build the real url of https request | 构建隐私保护通话平台请求路径
     *
     * @param path 接口访问URI
     * @return
     */
    private String buildOmpUrl(String path) {
        return ompDomainName + path;
    }
    @Override
    public void axBindNumber(String origNum, String privateNum, String calleeNumDisplay) {
        if (StringUtils.isBlank(origNum) || StringUtils.isBlank(privateNum) ||
StringUtils.isBlank(calleeNumDisplay)) {
            logger.info("axBindNumber set params error");
            return;
        }
        // 必填,AX模式绑定接口访问URI
        String url = "/rest/provision/caas/privatenumber/v1.0";
        String realUrl = buildOmpUrl(url);
        // 封装JOSN请求
        JSONObject json = new JSONObject();
        json.put("origNum", origNum); // A方真实号码(手机或固话)
        json.put("privateNum", privateNum); // 已订购的隐私号码(X号码)
        json.put("calleeNumDisplay", calleeNumDisplay); // 设置非A用户呼叫X时,A接到呼叫时的主显号码
        /**
         * 选填,各参数要求请参考"AX模式绑定接口"
         */
        //        json.put("privateNumType", "mobile-virtual"); //固定为mobile-virtual
        //        json.put("areaCode", "0755"); //城市码
        //        json.put("areaMatchMode", "1"); //号码筛选方式
        //        json.put("recordFlag", true); //是否通话录音
        //        json.put("recordHintTone", "recordHintTone.wav"); //录音提示音
        //        json.put("privateSms", true); //是否支持短信功能
        //        JSONObject preVoice = new JSONObject();
        //        preVoice.put("callerHintTone", "callerHintTone.wav"); //设置A拨打X号码时的通话前等待音
        //        preVoice.put("calleeHintTone", "calleeHintTone.wav"); //设置非A用户拨打X号码时的通话前等待
音
        //        json.put("preVoice", preVoice); //个性化通话前等待音
        //        json.put("maxDuration", 120); //设置允许单次通话进行的最长时间，单位为分钟
        //        json.put("userData", "test123"); //用户自定义数据
        //        json.put("callDirection", 0); //呼叫方向控制
        String result = HttpUtil.sendPost(appKey, appSecret, realUrl, json.toString());
        logger.info("Response is :" + result);
    }
    @Override
    public void axModifyNumber(String subscriptionId, String origNum, String privateNum, boolean
privateSms) {
        if (StringUtils.isBlank(subscriptionId) && StringUtils.isBlank(privateNum)) {
            logger.info("axModifyNumber set params error");
            return;
        }
        // 必填,AX模式绑定信息修改接口访问URI
        String url = "/rest/provision/caas/privatenumber/v1.0";
        String realUrl = buildOmpUrl(url);
        // 封装JOSN请求
        JSONObject json = new JSONObject();
        if (StringUtils.isNotBlank(subscriptionId)) {
            json.put("subscriptionId", subscriptionId); // 绑定关系ID
        } else {
            json.put("privateNum", privateNum); // AX中的X号码
        }
        if (StringUtils.isNotBlank(origNum)) {
            json.put("origNum", origNum); // AX中的A号码
        }
        json.put("privateSms", privateSms); // 是否支持短信功能
        /**
```

```
     * 选填,各参数要求请参考"AX模式绑定信息修改接口"
     */
    //        json.put("recordFlag", true); //是否通话录音
    //        json.put("maxDuration", 120); //修改允许单次通话进行的最长时间，单位为分钟
    //        json.put("userData", "test123"); //用户自定义数据
    //        json.put("callDirection", 0); //呼叫方向控制
    String result = HttpUtil.sendPut(appKey, appSecret, realUrl, json.toString());
    logger.info("Response is :" + result);
  }
  @Override
  public void axUnbindNumber(String subscriptionId, String privateNum) {
    if (StringUtils.isBlank(subscriptionId) && StringUtils.isBlank(privateNum)) {
      logger.info("axUnbindNumber set params error");
      return;
    }
    // 必填,AX模式解绑接口访问URI
    String url = "/rest/provision/caas/privatenumber/v1.0";
    String realUrl = buildOmpUrl(url);
    // 申明对象
    Map<String, Object> map = new HashMap<String, Object>();
    if (StringUtils.isNotBlank(subscriptionId)) {
      map.put("subscriptionId", subscriptionId); // 绑定关系ID
    } else {
      map.put("privateNum", privateNum); // AX中的X号码
    }
    String result = HttpUtil.sendDelete(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
    logger.info("Response is :" + result);
  }
  @Override
  public void axQueryBindRelation(String subscriptionId, String origNum, String privateNum) {
    if (StringUtils.isBlank(subscriptionId) && StringUtils.isBlank(origNum) &&
StringUtils.isBlank(privateNum)) {
      logger.error("axQueryBindRelation set parsms error");
      return;
    }
    // 必填,AX模式绑定信息查询接口访问URI
    String url = "/rest/provision/caas/privatenumber/v1.0";
    String realUrl = buildOmpUrl(url);
    // 申明对象
    Map<String, Object> map = new HashMap<String, Object>();
    if (StringUtils.isNotBlank(subscriptionId)) {
      map.put("subscriptionId", subscriptionId); // 指定绑定关系ID进行查询
    } else {
      if (StringUtils.isNotBlank(origNum)) {
        map.put("origNum", origNum); // 指定A号码进行查询
      } else {
        map.put("privateNum", privateNum); // 指定X号码进行查询
      }
    }
    String result = HttpUtil.sendGet(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
    logger.info("Response is :" + result);
  }
  @Override
  public void axSetCalleeNumber(String subscriptionId, String privateNum, String calleeNum) {
    if (StringUtils.isBlank(calleeNum)
        || (StringUtils.isBlank(subscriptionId) && StringUtils.isBlank(privateNum))) {
      logger.info("axSetCalleeNumber set params error");
      return;
    }
    // 必填,AX模式设置临时被叫接口
    String url = "/rest/caas/privatenumber/calleenumber/v1.0";
    String realUrl = buildOmpUrl(url);
    // 封装JOSN请求
    JSONObject json = new JSONObject();
    if (StringUtils.isNotBlank(subscriptionId)) {
      json.put("subscriptionId", subscriptionId); // 绑定关系ID
    } else {
      json.put("privateNum", privateNum); // AX中的X号码
    }
```

```
        json.put("calleeNum", calleeNum); // 本次呼叫的真实被叫号码
        /**
         * 选填,各参数要求请参考"AX模式设置临时被叫接口"
         */
        //        json.put("duration", 120); //临时被叫关系保持时间,单位为秒
        //        json.put("userData", "test123"); //用户自定义数据
        String result = HttpUtil.sendPut(appKey, appSecret, realUrl, json.toString());
        logger.info("Response is :" + result);
    }
    @Override
    public void axGetRecordDownloadLink(String recordDomain, String fileName) {
        if (StringUtils.isBlank(recordDomain) || StringUtils.isBlank(fileName)) {
            logger.info("axGetRecordDownloadLink set params error");
            return;
        }
        // 必填,AX模式获取录音文件下载地址接口访问URI
        String url = "/rest/provision/voice/record/v1.0";
        String realUrl = buildOmpUrl(url);
        // 申明对象
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("recordDomain", recordDomain); // 录音文件存储的服务器域名
        map.put("fileName", fileName); // 录音文件名
        String result = HttpUtil.sendGet(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
        logger.info("Response is :" + result);
    }
}
```

3.1.4 AXE 模式

AXE模式代码样例：

```
package com.huawei.hosting.service.voice;
import com.huawei.hosting.utils.HttpUtil;
import com.alibaba.fastjson.JSONObject;
import org.apache.commons.lang3.StringUtils;
import org.apache.log4j.Logger;
import java.util.HashMap;
import java.util.Map;
/**
 * AXE模式(具体场景请参考接口说明)
 */
public class AXEDemo {
    /**
     * 必填,请登录管理控制台,从"应用管理"页获取
     */
    private final static String OMPAPPKEY = "a1d1f50cad21415fbdd13d8f53d36d60"; // APP_Key
    private final static String OMPAPPSECRET = "cfc881cc704c4fba8d8fef5788e03e6b"; // APP_Secret
    private final static String OMPDOMAINNAME = "https://rtcpns.cn-north-1.myhuaweicloud.com"; // APP接
入地址
    public static void main(String[] args) {
        IAXEInterfaceDemo axe = new AXEInterfaceDemoImpl(OMPAPPKEY, OMPAPPSECRET,
OMPDOMAINNAME);
        // 第一步: 设置绑定关系,即调用AXE模式绑定接口,绑定成功获取分机号E,如: 1234
        //        axe.axeBindNumber("+8618010000001", "+8618612345678");
        // 当用户发起通话时,隐私保护通话平台会将呼叫事件推送到商户应用,参考: HostingVoiceEventDemoImpl
        // 第二步: 用户通话结束,若设置录音,则商户可以获取录音文件下载地址,即调用获取录音文件下载地址接口
        //        axe.axeGetRecordDownloadLink("ostor.huawei.com", "1200_366_0_20161228102743.wav");
        // 第三步: 隐私号码循环使用,商户可将绑定关系解绑,即调用AXE模式解绑接口
        //        axe.axeUnbindNumber(null, "+8618010000001", "1234");
        // 第四步: 商户可查询已订购的隐私号码的绑定信息,即调用AXE模式绑定信息查询接口
        axe.axeQueryBindRelation(null, "+8618010000001", "1234");
    }
}
/**
 * AXE模式接口
 */
interface IAXEInterfaceDemo {
```

```
    /**
     * Set the X number to the user's virtual number | 隐私号码AXE绑定
     *
     * @param virtualNum X号码
     * @param bindNum    AXE中的A号码
     */
    void axeBindNumber(String virtualNum, String bindNum);
    /**
     * Unbind the virtual number and extend number from number a | 隐私号码AXE解绑
     *
     * @param subscriptionId 绑定关系ID
     * @param virtualNum     X号码
     * @param extendNum      分机号E
     *                       subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
     */
    void axeUnbindNumber(String subscriptionId, String virtualNum, String extendNum);
    /**
     * Query the binding information for virtual number and extend number |
     * 查询AXE绑定信息
     *
     * @param subscriptionId 绑定关系ID
     * @param virtualNum     X号码
     * @param extendNum      分机号E
     *                       subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
     */
    void axeQueryBindRelation(String subscriptionId, String virtualNum, String extendNum);
    /**
     * Get download link of the record file created in call | 获取录音文件下载地址
     *
     * @param recordDomain 录音文件存储的服务器域名
     * @param fileName     录音文件名
     */
    void axeGetRecordDownloadLink(String recordDomain, String fileName);
}
/**
 * AXE模式接口测试
 */
class AXEInterfaceDemoImpl implements IAXEInterfaceDemo {
    private Logger logger = Logger.getLogger(AXEInterfaceDemoImpl.class);
    private String appKey; // APP_Key
    private String appSecret; // APP_Secret
    private String ompDomainName; // APP接入地址
    public AXEInterfaceDemoImpl(String appKey, String appSecret, String ompDomainName) {
        this.appKey = appKey;
        this.appSecret = appSecret;
        this.ompDomainName = ompDomainName;
    }
    /**
     * Build the real url of https request | 构建隐私保护通话平台请求路径
     *
     * @param path 接口访问URI
     * @return
     */
    private String buildOmpUrl(String path) {
        return ompDomainName + path;
    }
    @Override
    public void axeBindNumber(String virtualNum, String bindNum) {
        if (StringUtils.isBlank(virtualNum) || StringUtils.isBlank(bindNum)) {
            logger.info("axeBindNumber set params error");
            return;
        }
        // 必填,AXE模式绑定接口访问URI
        String url = "/rest/caas/extendnumber/v1.0";
        String realUrl = buildOmpUrl(url);
        // 封装JOSN请求
        JSONObject json = new JSONObject();
        json.put("virtualNum", virtualNum); // X号码
        json.put("bindNum", bindNum); // A方真实号码(手机或固话)
```



```
    /**
     * 选填,各参数要求请参考"AXE模式绑定接口"
     */
    //        json.put("areaCode", "0755"); // 城市码
    //        json.put("areaMatchMode", "1"); // 号码筛选方式
    //        json.put("extendNum", "1234"); // 分机号,即AXE中的E
    //        json.put("displayNumMode", "0"); // 非A用户呼叫X号码时,A看到的主显号码
    //        json.put("recordFlag", "false"); // 是否通话录音
    //        json.put("recordHintTone", "recordHintTone.wav"); // 录音提示音
    //        json.put("callbackTone", "callbackTone.wav"); // A呼叫X不存在回呼记录的提示音
    //        json.put("callbackNum", "+8618000000021"); // A呼叫X不存在回呼记录的转接号码
    //        json.put("timeUnit", "0"); // 绑定关系的有效时间
    //        json.put("bindExpiredTime", 24); // 绑定关系的有效时间
    //        json.put("callbackExpiredTime", 24); // 回呼记录有效时间
    //        json.put("userData", "abcdefg"); // 用户自定义数据
    String result = HttpUtil.sendPost(appKey, appSecret, realUrl, json.toString());
    logger.info("Response is :" + result);
}
@Override
public void axeUnbindNumber(String subscriptionId, String virtualNum, String extendNum) {
    if (StringUtils.isBlank(subscriptionId)
        && (StringUtils.isBlank(virtualNum) || StringUtils.isBlank(extendNum))) {
        logger.info("axeUnbindNumber set params error");
        return;
    }
    // 必填,AXE模式解绑接口访问URI
    String url = "/rest/caas/extendnumber/v1.0";
    String realUrl = buildOmpUrl(url);
    // 申明对象
    Map<String, Object> map = new HashMap<String, Object>();
    if (StringUtils.isNotBlank(subscriptionId)) {
        map.put("subscriptionId", subscriptionId); // 绑定关系ID
    } else {
        map.put("virtualNum", virtualNum); // AXE中的X号码
        map.put("extendNum", extendNum); // AXE中的E号码
    }
    String result = HttpUtil.sendDelete(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
    logger.info("Response is :" + result);
}
@Override
public void axeQueryBindRelation(String subscriptionId, String virtualNum, String extendNum) {
    if (StringUtils.isBlank(subscriptionId)
        && (StringUtils.isBlank(virtualNum) || StringUtils.isBlank(extendNum))) {
        logger.info("axeQueryBindRelation set params error");
        return;
    }
    // 必填,AXE模式绑定信息查询接口访问URI
    String url = "/rest/caas/extendnumber/v1.0";
    String realUrl = buildOmpUrl(url);
    // 申明对象
    Map<String, Object> map = new HashMap<String, Object>();
    if (StringUtils.isNotBlank(subscriptionId)) {
        map.put("subscriptionId", subscriptionId); // 绑定关系ID
    } else {
        map.put("virtualNum", virtualNum); // AXE中的X号码
        map.put("extendNum", extendNum); // AXE中的E号码
    }
    String result = HttpUtil.sendGet(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
    logger.info("Response is :" + result);
}
@Override
public void axeGetRecordDownloadLink(String recordDomain, String fileName) {
    if (StringUtils.isBlank(recordDomain) || StringUtils.isBlank(fileName)) {
        logger.info("axeGetRecordDownloadLink set params error");
        return;
    }
    // 必填,AXE模式获取录音文件下载地址接口访问URI
    String url = "/rest/provision/voice/record/v1.0";
    String realUrl = buildOmpUrl(url);
```

```
        // 申明对象
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("recordDomain", recordDomain); // 录音文件存储的服务器域名
        map.put("fileName", fileName); // 录音文件名
        String result = HttpUtil.sendGet(appKey, appSecret, realUrl, HttpUtil.map2UrlEncodeString(map));
        logger.info("Response is :" + result);
    }
}
```

3.1.5 呼叫事件与话单通知

呼叫事件与话单通知代码样例：

```
package com.huawei.hosting.service.voice;
import org.apache.log4j.Logger;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
/**
 * 呼叫事件通知/话单通知/短信通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知/话单通知/短信通知,可按如下样例解析处理
 */
public class HostingVoiceEventDemoImpl {
    private Logger logger = Logger.getLogger(HostingVoiceEventDemoImpl.class);
    /**
     * 呼叫事件 for AXB/AX/AXE
     *
     * @param jsonBody
     * @breif 详细内容以接口文档为准
     */
    public void onCallEvent(String jsonBody) {
        // 封装JOSN请求
        JSONObject json = JSON.parseObject(jsonBody);
        String eventType = json.getString("eventType"); // 通知事件类型
        if ("fee".equalsIgnoreCase(eventType)) {
            logger.info("EventType error: " + eventType);
            return;
        }
        JSONObject statusInfo = json.getJSONObject("statusInfo"); // 呼叫状态事件信息
        logger.info("eventType: " + eventType); // 打印通知事件类型
        //callin：呼入事件
        if ("callin".equalsIgnoreCase(eventType)) {
            /**
             * Example: 此处以解析notifyMode为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'subscriptionId': 绑定关系ID,AXE模式不携带该参数
             */
            String sessionId = statusInfo.getString("sessionId");
            logger.info("sessionId: " + sessionId);
            return;
        }
        //collectInfo：放音收号结果事件,仅AXE模式下的A被叫场景携带
        if ("collectInfo".equalsIgnoreCase(eventType)) {
            /**
             * Example: 此处以解析digitInfo为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'digitInfo': AXE场景中携带收号结果(即用户输入的数字)
             */
            String digitInfo = statusInfo.getString("digitInfo");
            logger.info("digitInfo: " + digitInfo);
```

```
            return;
        }
        //callout：呼出事件
        if ("callout".equalsIgnoreCase(eventType)) {
            /**
             * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'subscriptionId': 绑定关系ID
             */
            String sessionId = statusInfo.getString("sessionId");
            logger.info("sessionId: " + sessionId);
            return;
        }
        //alerting：振铃事件
        if ("alerting".equalsIgnoreCase(eventType)) {
            /**
             * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'subscriptionId': 绑定关系ID
             */
            String sessionId = statusInfo.getString("sessionId");
            logger.info("sessionId: " + sessionId);
            return;
        }
        //answer：应答事件
        if ("answer".equalsIgnoreCase(eventType)) {
            /**
             * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'subscriptionId': 绑定关系ID
             */
            String sessionId = statusInfo.getString("sessionId");
            logger.info("sessionId: " + sessionId);
            return;
        }
        //disconnect：挂机事件
        if ("disconnect".equalsIgnoreCase(eventType)) {
            /**
             * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'stateCode': 通话挂机的原因值
             * 'stateDesc': 通话挂机的原因值的描述
             * 'subscriptionId': 绑定关系ID
             */
            String sessionId = statusInfo.getString("sessionId");
            logger.info("sessionId: " + sessionId);
            return;
        }
    }
    /**
     * 话单通知 for AXB/AX/AXE/
     *
     * @param jsonBody
```

```
 * @breif 详细内容以接口文档为准
 */
public void onFeeEvent(String jsonBody) {
    // 封装JSON请求
    JSONObject json = JSON.parseObject(jsonBody);
    String eventType = json.getString("eventType"); // 通知事件类型
    if (!("fee".equalsIgnoreCase(eventType))) {
        logger.info("EventType error: " + eventType);
        return;
    }
    JSONArray feeLst = json.getJSONArray("feeLst"); // 呼叫话单事件信息
    /**
     * Example: 此处以解析notifyMode为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
     * 'sipStatusCode': 呼入,呼出的失败SIP状态码
     * 'recordFlag': 录音标识
     * 'recordStartTime': 录音开始时间
     * 'recordObjectName': 录音文件名
     * 'recordBucketName': 录音文件所在的目录名
     * 'recordDomain': 存放录音文件的域名
     * 'serviceType': 携带呼叫的业务类型信息
     * 'hostName': 话单生成的服务器设备对应的主机名
     * 'subscriptionId': 绑定关系ID
     * 'extendNum': 分机号E,该参数仅在AXE模式场景携带
     */
    // 短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (feeLst.size() > 1) {
        for (Object loop : feeLst) {
            if (((JSONObject) loop).containsKey("sessionId")) {
                logger.info("sessionId: " + ((JSONObject) loop).getString("sessionId"));
            }
        }
    } else if (feeLst.size() == 1) {
        if (feeLst.getJSONObject(0).containsKey("sessionId")) {
            logger.info("sessionId: " + feeLst.getJSONObject(0).getString("sessionId"));
        }
    } else {
        logger.info("feeLst error: no element.");
    }
}
/**
 * 短信通知 for AXB/AX
 *
 * @param jsonBody
 * @breif 详细内容以接口文档为准
 */
public void onSmsEvent(String jsonBody) {
    // 封装JOSN请求
    JSONObject json = JSON.parseObject(jsonBody);
    // String appKey = json.getString("appKey"); // 商户应用的AppKey
    JSONObject smsEvent = json.getJSONObject("smsEvent"); // 短信通知信息
```

```
        /**
         * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
         *
         * 'smsIdentifier': 短信唯一标识
         * 'notificationMode': 通知模式
         * 'calling': 真实发送方号码
         * 'called': 真实接收方号码
         * 'virtualNumber': 隐私号码(X号码)
         * 'event': 短信状态事件
         * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
         * 'subscriptionId': 绑定ID
         * 'smsContent': 用户发送的短信内容
         */
        String notificationMode = smsEvent.getString("notificationMode"); // 通知模式
        // 如果是Block模式,则要按接口文档进行回复响应
        if ("Block".equalsIgnoreCase(notificationMode)) {
            JSONObject resp = new JSONObject();
            JSONArray actions = new JSONArray();
            JSONObject action = new JSONObject();
            action.put("operation", "vNumberRoute"); // 操作类型
            actions.add(action);
            resp.put("actions", actions); // Block模式响应消息
            logger.info(resp);
        }
    }
}
```

3.2 PHP 代码样例

3.2.1 AXB 模式

样例	**AXB模式绑定接口** **AXB模式解绑接口** **AXB模式绑定信息修改接口** **AXB模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于PHP 7.2.9版本，要求PHP 5.6及以上版本。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXB 模式绑定接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; // APP接入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret
$relationNum = '+86180****0001'; // X号码(隐私号码)
$callerNum = '+86186****5678'; // A号码
$calleeNum = '+86186****5679'; // B号码

/*
 * 选填,各参数要求请参考"AXB模式绑定接口"
 */
// $areaCode = '0755'; // 需要绑定的X号码对应的城市码
// $callDirection = 0; // 允许呼叫的方向
// $duration = 86400; // 绑定关系保持时间,单位为秒。到期后会被系统自动解除绑定关系
// $recordFlag = 'false'; // 是否需要针对该绑定关系产生的所有通话录音
// $recordHintTone = 'recordHintTone.wav'; // 设置录音提示音
// $maxDuration = 60; // 设置允许单次通话进行的最长时间,单位为分钟。通话时间从接通被叫的时刻开始计算
// $privateSms = 'true'; // 设置该绑定关系是否支持短信功能

// $callerHintTone = 'callerHintTone.wav'; // 设置A拨打X号码时的通话前等待音
// $calleeHintTone = 'calleeHintTone.wav'; // 设置B拨打X号码时的通话前等待音
// $preVoice = [
//     'callerHintTone' => $callerHintTone,
//     'calleeHintTone' => $calleeHintTone
// ];

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求Body,可按需删除选填参数
$data = json_encode([
    'relationNum' => $relationNum,
    // 'areaCode' => $areaCode,
    'callerNum' => $callerNum,
    'calleeNum' => $calleeNum,
    // 'callDirection' => $callDirection,
    // 'duration' => $duration,
    // 'recordFlag' => $recordFlag,
    // 'recordHintTone' => $recordHintTone,
    // 'maxDuration' => $maxDuration,
    // 'privateSms' => $privateSms,
    // 'preVoice' => $preVoice
]);

$context_options = [
    'http' => [
        'method' => 'POST', // 请求方法为POST
        'header' => $headers,
        'content' => $data,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    print_r($data . PHP_EOL); // 打印请求数据
    fwrite($file, '绑定请求数据：' . $data . PHP_EOL); //绑定请求参数记录到本地文件,方便定位问题
```

```
    $response = file_get_contents($realUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
    fwrite($file, '绑定结果：' . $response . PHP_EOL); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"", $appKey, $base64, $nonce, $Created);
}

?>
```

AXB 模式解绑接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; // APP接入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AXB模式解绑接口"
 * subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
 */
$subscriptionId = '********';
$relationNum = '+86180****0001';

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'subscriptionId' => $subscriptionId,
    'relationNum' => $relationNum
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'DELETE', // 请求方法为DELETE
        'header' => $headers,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
```

```
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    print_r($data . PHP_EOL); // 打印请求数据
    $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
} catch (Exception $e) {
    echo $e->getMessage();
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AXB 模式绑定信息修改接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; // APP接
入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

$subscriptionId = '****'; // 必填,指定"AXB模式绑定接口"返回的绑定ID进行修改

/*
 * 选填,各参数要求请参考"AXB模式绑定信息修改接口"
 */
$callerNum = '+86186****5678'; // A号码
$calleeNum = '+86186****5679'; // B号码
// $callDirection = 0; //允许呼叫的方向
// $duration = 86400; //绑定关系保持时间,单位为秒。到期后会被系统自动解除绑定关系
// $maxDuration = 60; //设置允许单次通话进行的最长时间,单位为分钟。通话时间从接通被叫的时刻开始计算
// $privateSms = 'true'; //设置该绑定关系是否支持短信功能

// $callerHintTone = 'callerHintTone.wav'; // 设置A拨打X号码时的通话前等待音
// $calleeHintTone = 'calleeHintTone.wav'; // 设置B拨打X号码时的通话前等待音
// $preVoice = [
//     'callerHintTone' => $callerHintTone,
//     'calleeHintTone' => $calleeHintTone
// ];

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求Body,可按需删除选填参数
$data = json_encode([
    'subscriptionId' => $subscriptionId,
```

```
    'callerNum' => $callerNum,
    'calleeNum' => $calleeNum,
    // 'callDirection' => $callDirection,
    // 'duration' => $duration,
    // 'maxDuration' => $maxDuration,

    // 'privateSms' => $privateSms,
    // 'preVoice' => $preVoice
]);

$context_options = [
    'http' => [
        'method' => 'PUT', // 请求方法为PUT
        'header' => $headers,
        'content' => $data,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    print_r($data . PHP_EOL); // 打印请求数据
    $response = file_get_contents($realUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
} catch (Exception $e) {
    echo $e->getMessage();
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AXB 模式绑定信息查询接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; // APP接
入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AXB模式绑定信息查询接口"
 * subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
 */
$subscriptionId = '********'; // 指定"AXB模式绑定接口"返回的绑定ID进行查询
$relationNum = '+86180****0001'; // 指定X号码(隐私号码)进行查询
// $pageIndex = 1; //查询的分页索引,从1开始编号
// $pageSize = 20; //查询的分页大小,即每次查询返回多少条数据

// 请求Headers
```

```
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'subscriptionId' => $subscriptionId,
    'relationNum' => $relationNum
    // 'pageIndex' => $pageIndex,
    // 'pageSize' => $pageSize
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'GET', // 请求方法为GET
        'header' => $headers,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
    fwrite($file, '绑定查询结果： ' . $response . PHP_EOL); //查询结果,记录到本地文件
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

获取录音文件下载地址接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0'; // APP接入地址
+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

// 必填,通过"话单通知接口"获取
$recordDomain = '****.com';
```

```
$fileName = '****.wav';

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'recordDomain' => $recordDomain,
    'fileName' => $fileName
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'GET', // 请求方法为GET
        'header' => $headers,
        'max_redirects' => '0', // 关闭重定向
        'ignore_errors' => true // 获取错误码,方便调测
    ],// 为防止因代码工具重定向导致获取内容乱码,需要关闭重定向.若PHP是7.1.0及以上版本,若不需要可注释掉
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];
//若PHP版本介于5.1.3和7.1.0之间,请使用如下代码.若PHP是7.1.0及以上版本,请注释掉如下代码
// stream_context_set_default([
//      'ssl' => [
//          'verify_peer' => false,
//          'verify_peer_name' => false
//     ]
// ]);

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    //若PHP版本介于5.1.3和7.1.0之间,请使用如下代码获取响应消息头域信息
    //$http_response_header = get_headers($fullUrl, 1); //获取响应消息头域信息
    //若PHP是7.1.0及以上版本,请使用如下代码获取响应消息头域信息
    $http_response_header = get_headers($fullUrl, 1, stream_context_create($context_options)); //获取响应消
息头域信息
    if(strpos($http_response_header[0], '301') !== false){
        foreach ($http_response_header as $loop){
            if(strpos($loop, "Location") !== false){
                $fileUrl = trim(substr($loop, 10));
                print_r($fileUrl); //获取录音文件下载地址
                fwrite($file, '获取录音文件下载地址：' . $fileUrl . PHP_EOL); //查询结果,记录到本地文件
            }
        }
    }else{
        print_r($http_response_header[0] . PHP_EOL); //打印响应码400/401/403/500
//        $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); //获取响应消
息数据信息
//        print_r($response . PHP_EOL); // 打印响应结果
    }
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
```

```
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

呼叫事件通知接口

```
<?php
/**
 * 呼叫事件通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
 */

//呼叫事件通知样例
$jsonBody = json_encode([
    'eventType' => 'disconnect',
    'statusInfo' => [
        'sessionId' => '1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com',
        'timestamp' => '2019-01-23 09:16:41',
        'caller' => '+86138****0021',
        'called' => '+86138****7021',
        'stateCode' => 0,
        'stateDesc' => 'The user releases the call.',
        'subscriptionId' => '********'
    ]
]);

print_r($jsonBody . PHP_EOL);

//呼叫事件处理
onCallEvent($jsonBody);

/**
 * 呼叫事件通知
 * @desc 详细内容以接口文档为准
 * @param jsonArr
 */
function onCallEvent($jsonBody) {
    $jsonArr = json_decode($jsonBody, true); //将通知消息解析为关联数组
    $eventType = $jsonArr['eventType']; //通知事件类型

    if (strcasecmp($eventType, 'fee') == 0) {
        print_r('EventType error: ' . $eventType);
        return;
    }

    if (!array_key_exists('statusInfo', $jsonArr)) {
        print_r('param error: no statusInfo.');
        return;
    }
    $statusInfo = $jsonArr['statusInfo']; //呼叫状态事件信息

    print_r('eventType: ' . $eventType . PHP_EOL); //打印通知事件类型
    //callin：呼入事件
    if (strcasecmp($eventType, 'callin') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
```

```
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (array_key_exists('sessionId', $statusInfo)) {
        print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
    }
    return;
    ;
}
//callout：呼出事件
if (strcasecmp($eventType, 'callout') == 0) {
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (array_key_exists('sessionId', $statusInfo)) {
        print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
    }
    return;
}
//alerting：振铃事件
if (strcasecmp($eventType, 'alerting') == 0) {
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (array_key_exists('sessionId', $statusInfo)) {
        print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
    }
    return;
}
//answer：应答事件
if (strcasecmp($eventType, 'answer') == 0) {
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (array_key_exists('sessionId', $statusInfo)) {
        print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
    }
    return;
}
//disconnect：挂机事件
if (strcasecmp($eventType, 'disconnect') == 0) {
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'stateCode': 通话挂机的原因值
     * 'stateDesc': 通话挂机的原因值的描述
     * 'subscriptionId': 绑定关系ID
```

```
        */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
}
?>
```

话单通知接口

```
<?php
/**
 * 话单通知
 * 客户平台收到隐私保护通话平台的话单通知的接口通知
 */

//话单通知样例
$jsonBody = json_encode([
    'eventType' => 'fee',
    'feeLst' => [
        [
            'direction' => 1,
            'spId' => '********',
            'appKey' => '************',
            'icid' => 'ba171f34e6953fcd751edc77127748f4.3757223714.337238282.9',
            'bindNum' => '+86138****0022',
            'sessionId' => '1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com',
            'subscriptionId' => '********',
            'callerNum' => '+86138****0021',
            'calleeNum' => '+86138****0022',
            'fwdDisplayNum' => '+86138****0022',
            'fwdDstNum' => '+86138****7021',
            'callInTime' => '2019-01-23 09:15:14',
            'fwdStartTime' => '2019-01-23 09:15:15',
            'fwdAlertingTime' => '2019-01-23 09:15:21',
            'fwdAnswerTime' => '2019-01-23 09:15:36',
            'callEndTime' => '2019-01-23 09:16:41',
            'fwdUnaswRsn' => 0,
            'ulFailReason' => 0,
            'sipStatusCode' => 0,
            'callOutUnaswRsn' => 0,
            'recordFlag' => 1,
            'recordStartTime' => '2019-01-23 09:15:37',
            'recordDomain' => '****.com',
            'recordBucketName' => '****',
            'recordObjectName' => '****.wav',
            'ttsPlayTimes' => 0,
            'ttsTransDuration' => 0,
            'mptyId' => '****',
            'serviceType' => '004',
            'hostName' => 'callenabler246.huaweicaas.com'
        ]
    ]
]);

print_r($jsonBody . PHP_EOL);

//话单处理
onFeeEvent($jsonBody);

/**
 * 话单通知
 * @desc 详细内容以接口文档为准
 * @param jsonArr
 */
function onFeeEvent($jsonBody) {
    $jsonArr = json_decode($jsonBody, true); //将通知消息解析为关联数组
    $eventType = $jsonArr['eventType']; //通知事件类型
```

```
    if (strcasecmp($eventType, 'fee') != 0) {
        print_r('EventType error: ' . $eventType);
        return;
    }

    if (!array_key_exists('feeLst', $jsonArr)) {
        print_r('param error: no feeLst.');
        return;
    }
    $feeLst = $jsonArr['feeLst']; //呼叫话单事件信息

    print_r('eventType: ' . $eventType . PHP_EOL); //打印通知事件类型
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
     * 'sipStatusCode': 呼入,呼出的失败SIP状态码
     * 'recordFlag': 录音标识
     * 'recordStartTime': 录音开始时间
     * 'recordObjectName': 录音文件名
     * 'recordBucketName': 录音文件所在的目录名
     * 'recordDomain': 存放录音文件的域名
     * 'serviceType': 携带呼叫的业务类型信息
     * 'hostName': 话单生成的服务器设备对应的主机名
     * 'subscriptionId': 绑定关系ID
     */
    //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (sizeof($feeLst) > 1) {
        foreach ($feeLst as $loop){
            if (array_key_exists('sessionId', $loop)) {
                print_r('sessionId: ' . $loop['sessionId'] . PHP_EOL);
            }
        }
    } else if(sizeof($feeLst) == 1) {
        if (array_key_exists('sessionId', $feeLst[0])) {
            print_r('sessionId: ' . $feeLst[0]['sessionId'] . PHP_EOL);
        }
    } else {
        print_r('feeLst error: no element.');
    }
}
?>
```

短信通知接口

```
<?php
/**
 * 短信通知
 * 客户平台收到隐私保护通话平台的短信通知的接口通知
 */
```

```
//短信通知样例
$jsonBody = json_encode([
    'appKey' => '********',
    'smsEvent' => [
        'smsIdentifier' => '********',
        'notificationMode' => 'Block',
        'calling' => '+86138****0001',
        'virtualNumber' => '+86138****0000',
        'event' => 'TextSMS',
        'timeStamp' => '2018-09-13T09:46:16.023Z'
    ]
]);

print_r($jsonBody . PHP_EOL);

/*
 * 短信通知处理
 */
onSmsEvent($jsonBody);

/**
 * 短信通知
 * @desc 详细内容以接口文档为准
 * @param jsonBody
 */
function onSmsEvent($jsonBody) {
    //将通知消息解析为关联数组
    $jsonArr = json_decode($jsonBody, true);

    if (!array_key_exists('smsEvent', $jsonArr)) {
        print_r('param error: no smsEvent.');
        return;
    }

    //print_r($jsonArr['appKey'] . PHP_EOL); //商户应用的AppKey

    $smsEvent = $jsonArr['smsEvent']; //短信通知信息

    /**
     * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
     *
     * 'smsIdentifier': 短信唯一标识
     * 'notificationMode': 通知模式
     * 'calling': 真实发送方号码
     * 'called': 真实接收方号码
     * 'virtualNumber': 隐私号码(X号码)
     * 'event': 短信状态事件
     * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
     * 'subscriptionId': 绑定ID
     * 'smsContent': 用户发送的短信内容
     */
    if (array_key_exists('notificationMode', $smsEvent)) {
        if (strcasecmp($smsEvent['notificationMode'], 'Block') == 0) {
            //收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
            $actions = [
                'operation' => 'vNumberRoute', //操作类型:转发短信/丢弃短信
                'message' => [
                    'called' => '+86138****7022', //真实接收方号码
                    'calling' => '+86138****7021' //真实发送方号码
                ]
            ];
            $resp = json_encode(['actions' => [$actions]]); //Block模式响应消息
            print_r($resp . PHP_EOL);

        } elseif (strcasecmp($smsEvent['notificationMode'], 'Notify') == 0) {
            //收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
            //http_response_code(200);
            print_r('This is AXB sms Notify mode.');
        } else {
```

```
            print_r('notificationMode param error.');
        }
    }
}
?>
```

3.2.2 AX 模式

样例	**AX模式绑定接口** **AX模式解绑接口** **AX模式绑定信息修改接口** **AX模式绑定信息查询接口** **AX模式设置临时被叫接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于PHP 7.2.9版本，要求PHP 5.6及以上版本。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AX 模式绑定接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; // APP接入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret
$origNum = '+86186****5678'; // A号码
$privateNum = '+86180****0001'; // X号码(隐私号码)

/*
 * 选填,各参数要求请参考"AX模式绑定接口"
 */
// $privateNumType = 'mobile-virtual'; //固定为mobile-virtual
// $areaCode = '0755'; //需要绑定的X号码对应的城市码
// $recordFlag = 'false'; //是否需要针对该绑定关系产生的所有通话录音
// $recordHintTone = 'recordHintTone.wav'; //设置录音提示音
$calleeNumDisplay = '0'; // 设置非A用户呼叫X时,A接到呼叫时的主显号码
// $privateSms = 'true'; //设置该绑定关系是否支持短信功能

// $callerHintTone = 'callerHintTone.wav'; //设置A拨打X号码时的通话前等待音
// $calleeHintTone = 'calleeHintTone.wav'; //设置非A用户拨打X号码时的通话前等待音
```

```
// $preVoice = [
//     'callerHintTone' => $callerHintTone,
//     'calleeHintTone' => $calleeHintTone
// ];

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求Body,可按需删除选填参数
$data = json_encode([
    'origNum' => $origNum,
    'privateNum' => $privateNum,
    // 'privateNumType' => $privateNumType,
    // 'areaCode' => $areaCode,
    // 'recordFlag' => $recordFlag,
    // 'recordHintTone' => $recordHintTone,
    'calleeNumDisplay' => $calleeNumDisplay,
    // 'privateSms' => $privateSms,
    // 'preVoice' => $preVoice
]);

$context_options = [
    'http' => [
        'method' => 'POST', // 请求方法为POST
        'header' => $headers,
        'content' => $data,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    print_r($data . PHP_EOL); // 打印请求数据
    fwrite($file, '绑定请求数据：' . $data . PHP_EOL); //绑定请求参数记录到本地文件,方便定位问题
    $response = file_get_contents($realUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
    fwrite($file, '绑定结果：' . $response . PHP_EOL); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系
及解绑
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AX 模式解绑接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; // APP接入
地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AX模式解绑接口"
 * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
 */
$origNum = '+86186****5678'; // A号码
$privateNum = '+86180****0001'; // X号码
// $subscriptionId = '********'; // 指定"AX模式绑定接口"返回的绑定ID进行解绑

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'origNum' => $origNum,
    'privateNum' => $privateNum
    // 'subscriptionId' => $subscriptionId
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'DELETE', // 请求方法为DELETE
        'header' => $headers,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    print_r($data . PHP_EOL); // 打印请求数据
    $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
} catch (Exception $e) {
    echo $e->getMessage();
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
```

```
}
?>
```

AX 模式绑定信息修改接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; // APP接入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AX模式绑定信息修改接口"
 * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
 */
$origNum = '+86186****5678'; // A号码
$privateNum = '+86180****0001'; // X号码
$subscriptionId = '********'; // 指定"AX模式绑定接口"返回的绑定ID进行修改

$privateSms = 'true'; // 必填,修改该绑定关系是否支持短信功能

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求Body,可按需删除选填参数
$data = json_encode([
    'origNum' => $origNum,
    'privateNum' => $privateNum,
    'subscriptionId' => $subscriptionId,
    'privateSms' => $privateSms
]);

$context_options = [
    'http' => [
        'method' => 'PUT', // 请求方法为PUT
        'header' => $headers,
        'content' => $data,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    print_r($data . PHP_EOL); // 打印请求数据
    $response = file_get_contents($realUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
} catch (Exception $e) {
    echo $e->getMessage();
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
```

```
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AX 模式绑定信息查询接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; // APP接入
地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AX模式绑定信息查询接口"
 * subscriptionId和origNum为二选一关系,两者都携带时以subscriptionId为准
 */
$origNum = '+86186****5678'; // 指定A号码进行查询
$subscriptionId = '********'; // 指定"AX模式绑定接口"返回的绑定ID进行查询

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'origNum' => $origNum,
    'subscriptionId' => $subscriptionId
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'GET', // 请求方法为GET
        'header' => $headers,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
    fwrite($file, '绑定查询结果： ' . $response . PHP_EOL); //查询结果,记录到本地文件
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
```

```
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AX 模式设置临时被叫接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/privatenumber/calleenumber/v1.0'; //
APP接入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AX模式设置临时被叫接口"
 * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
 */
$origNum = '+86186****5678'; // A号码
$privateNum = '+86180****0001'; // X号码
// $subscriptionId = '********'; // 指定"AX模式绑定接口"返回的绑定ID设置临时被叫

$calleeNum = '+86186****5679'; // 必填,本次呼叫的真实被叫号码，origNum和calleeNum不能相同

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求Body,可按需删除选填参数
$data = json_encode([
    'origNum' => $origNum,
    'privateNum' => $privateNum,
    // 'subscriptionId' => $subscriptionId,
    'calleeNum' => $calleeNum
]);

$context_options = [
    'http' => [
        'method' => 'PUT', // 请求方法为PUT
        'header' => $headers,
        'content' => $data,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    print_r($data . PHP_EOL); // 打印请求数据
    $response = file_get_contents($realUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
} catch (Exception $e) {
    echo $e->getMessage();
}

/**
```

```
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

获取录音文件下载地址接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0'; // APP接入地址
+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

// 必填,通过"话单通知接口"获取
$recordDomain = '****.com';
$fileName = '****.wav';

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'recordDomain' => $recordDomain,
    'fileName' => $fileName
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'GET', // 请求方法为GET
        'header' => $headers,
        'max_redirects' => '0', // 关闭重定向
        'ignore_errors' => true // 获取错误码,方便调测
    ],// 为防止因代码工具重定向导致获取内容乱码,需要关闭重定向.若PHP是7.1.0及以上版本,若不需要可注释掉
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];
//若PHP版本介于5.1.3和7.1.0之间,请使用如下代码.若PHP是7.1.0及以上版本,请注释掉如下代码
// stream_context_set_default([
//     'ssl' => [
//         'verify_peer' => false,
//         'verify_peer_name' => false
//     ]
// ]);

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    //若PHP版本介于5.1.3和7.1.0之间,请使用如下代码获取响应消息头域信息
```

```
    //$http_response_header = get_headers($fullUrl, 1); //获取响应消息头域信息
    //若PHP是7.1.0及以上版本,请使用如下代码获取响应消息头域信息
    $http_response_header = get_headers($fullUrl, 1, stream_context_create($context_options)); //获取响应消息头域信息
    if(strpos($http_response_header[0], '301') !== false){
        foreach ($http_response_header as $loop){
            if(strpos($loop, "Location") !== false){
                $fileUrl = trim(substr($loop, 10));
                print_r($fileUrl); //获取录音文件下载地址
                fwrite($file, '获取录音文件下载地址：' . $fileUrl . PHP_EOL); //查询结果,记录到本地文件
            }
        }
    }else{
        print_r($http_response_header[0] . PHP_EOL); //打印响应码400/401/403/500
//        $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); //获取响应消息数据信息
//        print_r($response . PHP_EOL); // 打印响应结果
    }
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"", $appKey, $base64, $nonce, $Created);
}
?>
```

呼叫事件通知接口

```
<?php
/**
 * 呼叫事件通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
 */

//呼叫事件通知样例
$jsonBody = json_encode([
    'eventType' => 'disconnect',
    'statusInfo' => [
        'sessionId' => '1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com',
        'timestamp' => '2019-01-24 02:30:22',
        'caller' => '+86138****0022',
        'called' => '+86138****7021',
        'stateCode' => 0,
        'stateDesc' => 'The user releases the call.',
        'subscriptionId' => '********'
    ]
]);

print_r($jsonBody . PHP_EOL);

//呼叫事件处理
onCallEvent($jsonBody);
```

```
/**
 * 呼叫事件通知
 * @desc 详细内容以接口文档为准
 * @param jsonArr
 */
function onCallEvent($jsonBody) {
    $jsonArr = json_decode($jsonBody, true); //将通知消息解析为关联数组
    $eventType = $jsonArr['eventType']; //通知事件类型

    if (strcasecmp($eventType, 'fee') == 0) {
        print_r('EventType error: ' . $eventType);
        return;
    }

    if (!array_key_exists('statusInfo', $jsonArr)) {
        print_r('param error: no statusInfo.');
        return;
    }
    $statusInfo = $jsonArr['statusInfo']; //呼叫状态事件信息

    print_r('eventType: ' . $eventType . PHP_EOL); //打印通知事件类型
    //callin：呼入事件
    if (strcasecmp($eventType, 'callin') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
    //callout：呼出事件
    if (strcasecmp($eventType, 'callout') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
    //alerting：振铃事件
    if (strcasecmp($eventType, 'alerting') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
```

```
    //answer：应答事件
    if (strcasecmp($eventType, 'answer') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
    //disconnect：挂机事件
    if (strcasecmp($eventType, 'disconnect') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'stateCode': 通话挂机的原因值
         * 'stateDesc': 通话挂机的原因值的描述
         * 'subscriptionId': 绑定关系ID
         */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
}
?>
```

话单通知接口

```
<?php
/**
 * 话单通知
 * 客户平台收到隐私保护通话平台的话单通知的接口通知
 */

//话单通知样例
$jsonBody = json_encode([
    'eventType' => 'fee',
    'feeLst' => [
        [
            'direction' => 1,
            'spId' => '********',
            'appKey' => '********',
            'icid' => 'ba171f34e6953fcd751edc77127748f4.3757285803.338666679.5',
            'bindNum' => '+86138****0022',
            'sessionId' => '1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com',
            'subscriptionId' => '********',
            'callerNum' => '+86138****0021',
            'calleeNum' => '+86138****0022',
            'fwdDisplayNum' => '+86138****0022',
            'fwdDstNum' => '+86138****7021',
            'callInTime' => '2019-01-24 02:30:03',
            'fwdStartTime' => '2019-01-24 02:30:03',
            'fwdAlertingTime' => '2019-01-24 02:30:04',
            'fwdAnswerTime' => '2019-01-24 02:30:06',
            'callEndTime' => '2019-01-24 02:30:22',
            'fwdUnaswRsn' => 0,
            'ulFailReason' => 0,
            'sipStatusCode' => 0,
```

```
            'callOutUnaswRsn' => 0,
            'recordFlag' => 1,
            'recordStartTime' => '2019-01-24 02:30:06',
            'recordDomain' => '****.com',
            'recordBucketName' => '********',
            'recordObjectName' => '****.wav',
            'ttsPlayTimes' => 0,
            'ttsTransDuration' => 0,
            'mptyId' => '****',
            'serviceType' => '003',
            'hostName' => 'callenabler246.huaweicaas.com'
        ]
    ]
]);

print_r($jsonBody . PHP_EOL);

//话单处理
onFeeEvent($jsonBody);

/**
 * 话单通知
 * @desc 详细内容以接口文档为准
 * @param jsonArr
 */
function onFeeEvent($jsonBody) {
    $jsonArr = json_decode($jsonBody, true); //将通知消息解析为关联数组
    $eventType = $jsonArr['eventType']; //通知事件类型

    if (strcasecmp($eventType, 'fee') != 0) {
        print_r('EventType error: ' . $eventType);
        return;
    }

    if (!array_key_exists('feeLst', $jsonArr)) {
        print_r('param error: no feeLst.');
        return;
    }
    $feeLst = $jsonArr['feeLst']; //呼叫话单事件信息

    print_r('eventType: ' . $eventType . PHP_EOL); //打印通知事件类型
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
     * 'sipStatusCode': 呼入,呼出的失败SIP状态码
     * 'recordFlag': 录音标识
     * 'recordStartTime': 录音开始时间
     * 'recordObjectName': 录音文件名
     * 'recordBucketName': 录音文件所在的目录名
     * 'recordDomain': 存放录音文件的域名
     * 'serviceType': 携带呼叫的业务类型信息
```

```
     * 'hostName': 话单生成的服务器设备对应的主机名
     * 'subscriptionId': 绑定关系ID
     */
    //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (sizeof($feeLst) > 1) {
        foreach ($feeLst as $loop){
            if (array_key_exists('sessionId', $loop)) {
                print_r('sessionId: ' . $loop['sessionId'] . PHP_EOL);
            }
        }
    } else if(sizeof($feeLst) == 1) {
        if (array_key_exists('sessionId', $feeLst[0])) {
            print_r('sessionId: ' . $feeLst[0]['sessionId'] . PHP_EOL);
        }
    } else {
        print_r('feeLst error: no element.');
    }
}
?>
```

短信通知接口

```
<?php
/**
 * 短信通知
 * 客户平台收到隐私保护通话平台的短信通知的接口通知
 */

//短信通知样例
$jsonBody = json_encode([
    'appKey' => '********',
    'smsEvent' => [
        'smsIdentifier' => '********',
        'notificationMode' => 'Block',
        'calling' => '+86138****0001',
        'virtualNumber' => '+86138****0000',
        'event' => 'TextSMS',
        'timeStamp' => '2018-09-13T09:46:16.023Z'
    ]
]);

print_r($jsonBody . PHP_EOL);

/*
 * 短信通知处理
 */
onSmsEvent($jsonBody);

/**
 * 短信通知
 * @desc 详细内容以接口文档为准
 * @param jsonBody
 */
function onSmsEvent($jsonBody) {
    //将通知消息解析为关联数组
    $jsonArr = json_decode($jsonBody, true);

    if (!array_key_exists('smsEvent', $jsonArr)) {
        print_r('param error: no smsEvent.');
        return;
    }

    //print_r($jsonArr['appKey'] . PHP_EOL); //商户应用的AppKey

    $smsEvent = $jsonArr['smsEvent']; //短信通知信息

    /**
     * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
     *
```

```
    * 'smsIdentifier': 短信唯一标识
    * 'notificationMode': 通知模式
    * 'calling': 真实发送方号码
    * 'called': 真实接收方号码
    * 'virtualNumber': 隐私号码(X号码)
    * 'event': 短信状态事件
    * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
    * 'subscriptionId': 绑定ID
    * 'smsContent': 用户发送的短信内容
    */
    if (array_key_exists('notificationMode', $smsEvent)) {
        if (strcasecmp($smsEvent['notificationMode'], 'Block') == 0) {
            //收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
            $actions = [
                'operation' => 'vNumberRoute', //操作类型:转发短信/丢弃短信
                'message' => [
                    'called' => '+86138****7022', //真实接收方号码
                    'calling' => '+86138****7021' //真实发送方号码
                ]
            ];
            $resp = json_encode(['actions' => [$actions]]); //Block模式响应消息
            print_r($resp . PHP_EOL);

        } elseif (strcasecmp($smsEvent['notificationMode'], 'Notify') == 0) {
            //收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
            //http_response_code(200);
            print_r('This is AX sms Notify mode.');
        } else {
            print_r('notificationMode param error.');
        }
    }
}
?>
```

3.2.3 AXE 模式

样例	**AXE模式绑定接口** **AXE模式解绑接口** **AXE模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口**
环境要求	基于PHP 7.2.9版本，要求PHP 5.6及以上版本。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXE 模式绑定接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0'; // APP接入地址+接口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret
$virtualNum = '+86180****0001'; // AXE中的X号码
$bindNum = '+86186****5678'; // AXE中的A号码

/*
 * 选填,各参数要求请参考"AXE模式绑定接口"
 */
// $areaCode = '0755'; // 需要绑定的X号码对应的城市码
// $displayNumMode = '0'; // 非A用户呼叫X号码时，A看到的主显号码
// $recordFlag = 'false'; // 是否需要针对该绑定关系产生的所有通话录音
// $recordHintTone = 'recordHintTone.wav'; // 设置录音提示音
// $callbackTone = 'callbackTone.wav'; // A呼叫X不存在回呼记录的提示音
// $callbackNum = '+86180****0021'; // A呼叫X不存在回呼记录的转接号码
// $bindExpiredTime = 24; // 绑定关系的有效时间
// $callbackExpiredTime = 24; // 回呼记录有效时间
// $userData = 'abcdefg'; // 用户自定义数据

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求Body,可按需删除选填参数
$data = json_encode([
    'virtualNum' => $virtualNum,
    'bindNum' => $bindNum
    // 'areaCode' => $areaCode,
    // 'displayNumMode' => $displayNumMode,
    // 'recordFlag' => $recordFlag,
    // 'recordHintTone' => $recordHintTone,
    // 'callbackTone' => $callbackTone,
    // 'callbackNum' => $callbackNum,
    // 'bindExpiredTime' => $bindExpiredTime,
    // 'callbackExpiredTime' => $callbackExpiredTime,
    // 'userData' => $userData
]);

$context_options = [
    'http' => [
        'method' => 'POST', // 请求方法为POST
        'header' => $headers,
        'content' => $data,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    print_r($data . PHP_EOL); // 打印请求数据
    fwrite($file, '绑定请求数据：' . $data . PHP_EOL); //绑定请求参数记录到本地文件,方便定位问题
    $response = file_get_contents($realUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
    fwrite($file, '绑定结果：' . $response . PHP_EOL); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
} catch (Exception $e) {
    echo $e->getMessage();
```

```
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AXE 模式解绑接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0'; // APP接入地址+接
口访问URI
$APP_KEY = 'a1********'; // APP_Key
$APP_SECRET = 'cfc8****************'; // APP_Secret

/*
 * 选填,各参数要求请参考"AXE模式解绑接口"
 * subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
 */
$subscriptionId = '********'; // 指定"AXE模式绑定接口"返回的绑定ID进行解绑
$virtualNum = '+86180****0001'; // AXE中的X号码
$extendNum = '1234'; // AXE中的E号码

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'subscriptionId' => $subscriptionId,
    'virtualNum' => $virtualNum,
    'extendNum' => $extendNum
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'DELETE', // 请求方法为DELETE
        'header' => $headers,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];

try {
```

```
    print_r($data . PHP_EOL); // 打印请求数据
    $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
} catch (Exception $e) {
    echo $e->getMessage();
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

AXE 模式绑定信息查询接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0'; // APP接入地址+接
口访问URI
$APP_KEY = 'a1****'; // APP_Key
$APP_SECRET = 'cfc8**********'; // APP_Secret

/*
 * 选填,各参数要求请参考"AXE模式绑定信息查询接口"
 * subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
 */
$subscriptionId = '********'; // 指定"AXE模式绑定接口"返回的绑定ID进行解绑
$virtualNum = '+86180****0001'; // AXE中的X号码
$extendNum = '1234'; // AXE中的E号码

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'subscriptionId' => $subscriptionId,
    'virtualNum' => $virtualNum,
    'extendNum' => $extendNum
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'GET', // 请求方法为GET
        'header' => $headers,
        'ignore_errors' => true // 获取错误码,方便调测
    ],
    'ssl' => [
        'verify_peer' => false,
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
```

```
];

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); // 发送请求
    print_r($response . PHP_EOL); // 打印响应结果
    fwrite($file, '绑定查询结果： ' . $response . PHP_EOL); //查询结果,记录到本地文件
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //
PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s
\"", $appKey, $base64, $nonce, $Created);
}
?>
```

获取录音文件下载地址接口

```
<?php

// 必填,请参考"开发准备"获取如下数据,替换为实际值
$realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0'; // APP接入地址
+接口访问URI
$APP_KEY = 'a1****'; // APP_Key
$APP_SECRET = 'cfc8**********'; // APP_Secret

// 必填,通过"话单通知接口"获取
$recordDomain = '****.com';
$fileName = '****.wav';

// 请求Headers
$headers = [
    'Accept: application/json',
    'Content-Type: application/json;charset=UTF-8',
    'Authorization: AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
    'X-AKSK: ' . buildAKSKHeader($APP_KEY, $APP_SECRET)
];
// 请求URL参数
$data = http_build_query([
    'recordDomain' => $recordDomain,
    'fileName' => $fileName
]);
// 完整请求地址
$fullUrl = $realUrl . '?' . $data;

$context_options = [
    'http' => [
        'method' => 'GET', // 请求方法为GET
        'header' => $headers,
        'max_redirects' => '0', // 关闭重定向
        'ignore_errors' => true // 获取错误码,方便调测
    ],// 为防止因代码工具重定向导致获取内容乱码,需要关闭重定向.若PHP是7.1.0及以上版本,若不需要可注释掉
    'ssl' => [
        'verify_peer' => false,
```

```
        'verify_peer_name' => false
    ] // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
];
//若PHP版本介于5.1.3和7.1.0之间,请使用如下代码.若PHP是7.1.0及以上版本,请注释掉如下代码
// stream_context_set_default([
//     'ssl' => [
//         'verify_peer' => false,
//         'verify_peer_name' => false
//     ]
// ]);

try {
    $file=fopen('bind_data.txt', 'a'); //打开文件
    //若PHP版本介于5.1.3和7.1.0之间,请使用如下代码获取响应消息头域信息
    //$http_response_header = get_headers($fullUrl, 1); //获取响应消息头域信息
    //若PHP是7.1.0及以上版本,请使用如下代码获取响应消息头域信息
    $http_response_header = get_headers($fullUrl, 1, stream_context_create($context_options)); //获取响应消息头域信息
    if(strpos($http_response_header[0], '301') !== false){
        foreach ($http_response_header as $loop){
            if(strpos($loop, "Location") !== false){
                $fileUrl = trim(substr($loop, 10));
                print_r($fileUrl); //获取录音文件下载地址
                fwrite($file, '获取录音文件下载地址：' . $fileUrl . PHP_EOL); //查询结果,记录到本地文件
            }
        }
    }else{
        print_r($http_response_header[0] . PHP_EOL); //打印响应码400/401/403/500
//        $response = file_get_contents($fullUrl, false, stream_context_create($context_options)); //获取响应消息数据信息
//        print_r($response . PHP_EOL); // 打印响应结果
    }
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    fclose($file); //关闭文件
}

/**
 * buildAKSKHeader
 * @param string $appKey
 * @param string $appSecret
 * @return string
 */
function buildAKSKHeader($appKey, $appSecret) {
    date_default_timezone_set("UTC");
    $Created = date('Y-m-d\TH:i:s\Z'); //Created
    $nonce = uniqid(); //Nonce
    $base64 = base64_encode(hash_hmac('sha256', ($nonce . $Created), $appSecret, TRUE)); //PasswordDigest

    return sprintf("UsernameToken Username=\"%s\",PasswordDigest=\"%s\",Nonce=\"%s\",Created=\"%s\"", $appKey, $base64, $nonce, $Created);
}
?>
```

呼叫事件通知接口

```
<?php
/**
 * 呼叫事件通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
 */

//呼叫事件通知样例
$jsonBody = json_encode([
    'eventType' => 'disconnect',
    'statusInfo' => [
        'sessionId' => '1202_1051_4294967295_20190124070250@callenabler246.huaweicaas.com',
```



```
        'timestamp' => '2019-01-24 07:03:28',
        'caller' => '+86138****0022',
        'called' => '+86138****7021',
        'stateCode' => 0,
        'stateDesc' => 'The user releases the call.',
        'subscriptionId' => '************'
    ]
]);

print_r($jsonBody . PHP_EOL);

//呼叫事件处理
onCallEvent($jsonBody);

/**
 * 呼叫事件通知
 * @desc 详细内容以接口文档为准
 * @param jsonArr
 */
function onCallEvent($jsonBody) {
    $jsonArr = json_decode($jsonBody, true); //将通知消息解析为关联数组
    $eventType = $jsonArr['eventType']; //通知事件类型

    if (strcasecmp($eventType, 'fee') == 0) {
        print_r('EventType error: ' . $eventType);
        return;
    }

    if (!array_key_exists('statusInfo', $jsonArr)) {
        print_r('param error: no statusInfo.');
        return;
    }
    $statusInfo = $jsonArr['statusInfo']; //呼叫状态事件信息

    print_r('eventType: ' . $eventType . PHP_EOL); //打印通知事件类型
    //callin：呼入事件
    if (strcasecmp($eventType, 'callin') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
        ;
    }
    //collectInfo：放音收号结果事件,仅AXE模式下的A被叫场景携带
    if (strcasecmp($eventType, 'collectInfo') == 0) {
        /**
         * Example: 此处以解析digitInfo为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'digitInfo': AXE场景中携带收号结果(即用户输入的数字)
         */
        if (array_key_exists('digitInfo', $statusInfo)) {
            print_r('digitInfo: ' . $statusInfo['digitInfo'] . PHP_EOL);
        }
        return;
    }
    //callout：呼出事件
    if (strcasecmp($eventType, 'callout') == 0) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
```

```
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
    //alerting：振铃事件
    if (strcasecmp($eventType, 'alerting') == 0) {
        /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
    //answer：应答事件
    if (strcasecmp($eventType, 'answer') == 0) {
        /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
    //disconnect：挂机事件
    if (strcasecmp($eventType, 'disconnect') == 0) {
        /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'stateCode': 通话挂机的原因值
        * 'stateDesc': 通话挂机的原因值的描述
        * 'subscriptionId': 绑定关系ID
        */
        if (array_key_exists('sessionId', $statusInfo)) {
            print_r('sessionId: ' . $statusInfo['sessionId'] . PHP_EOL);
        }
        return;
    }
}
?>
```

话单通知接口

```
<?php
/**
```

```
 * 话单通知
 * 客户平台收到隐私保护通话平台的话单通知的接口通知
 */

//话单通知样例
$jsonBody = json_encode([
    'eventType' => 'fee',
    'feeLst' => [
        [
            'direction' => 0,
            'spId' => '********',
            'appKey' => '************',
            'icid' => 'ba171f34e6953fcd751edc77127748f4.3757289590.338833305.5',
            'bindNum' => '+86138****0022',
            'sessionId' => '1201_11275_4294967295_20190124033310@callenabler246.huaweicaas.com',
            'subscriptionId' => '************',
            'callerNum' => '+86138****7021',
            'calleeNum' => '+86138****0022',
            'fwdDisplayNum' => '+86138****0022',
            'fwdDstNum' => '+86138****0021',
            'callInTime' => '2019-01-24 03:33:10',
            'fwdStartTime' => '2019-01-24 03:33:16',
            'fwdAlertingTime' => '2019-01-24 03:33:19',
            'fwdAnswerTime' => '2019-01-24 03:33:28',
            'callEndTime' => '2019-01-24 03:33:57',
            'fwdUnaswRsn' => 0,
            'ulFailReason' => 0,
            'sipStatusCode' => 0,
            'callOutUnaswRsn' => 0,
            'recordFlag' => 1,
            'recordStartTime' => '2019-01-24 03:33:28',
            'recordDomain' => '****.com',
            'recordBucketName' => '********',
            'recordObjectName' => '****.wav',
            'ttsPlayTimes' => 0,
            'ttsTransDuration' => 0,
            'mptyId' => '********',
            'serviceType' => '005',
            'hostName' => 'callenabler246.huaweicaas.com',
            'extendNumber' => '02'
        ]
    ]
]);

print_r($jsonBody . PHP_EOL);

//话单处理
onFeeEvent($jsonBody);

/**
 * 话单通知
 * @desc 详细内容以接口文档为准
 * @param jsonArr
 */
function onFeeEvent($jsonBody) {
    $jsonArr = json_decode($jsonBody, true); //将通知消息解析为关联数组
    $eventType = $jsonArr['eventType']; //通知事件类型

    if (strcasecmp($eventType, 'fee') != 0) {
        print_r('EventType error: ' . $eventType);
        return;
    }

    if (!array_key_exists('feeLst', $jsonArr)) {
        print_r('param error: no feeLst.');
        return;
    }
    $feeLst = $jsonArr['feeLst']; //呼叫话单事件信息
```

```
    print_r('eventType: ' . $eventType . PHP_EOL); //打印通知事件类型
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
     * 'sipStatusCode': 呼入,呼出的失败SIP状态码
     * 'recordFlag': 录音标识
     * 'recordStartTime': 录音开始时间
     * 'recordObjectName': 录音文件名
     * 'recordBucketName': 录音文件所在的目录名
     * 'recordDomain': 存放录音文件的域名
     * 'serviceType': 携带呼叫的业务类型信息
     * 'hostName': 话单生成的服务器设备对应的主机名
     * 'subscriptionId': 绑定关系ID
     * 'extendNum': 分机号E
     */
    //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (sizeof($feeLst) > 1) {
        foreach ($feeLst as $loop){
            if (array_key_exists('sessionId', $loop)) {
                print_r('sessionId: ' . $loop['sessionId'] . PHP_EOL);
            }
        }
    } else if(sizeof($feeLst) == 1) {
        if (array_key_exists('sessionId', $feeLst[0])) {
            print_r('sessionId: ' . $feeLst[0]['sessionId'] . PHP_EOL);
        }
    } else {
        print_r('feeLst error: no element.');
    }
}
?>
```

3.3 Python 代码样例

3.3.1 AXB 模式

样例	**AXB模式绑定接口** **AXB模式解绑接口** **AXB模式绑定信息修改接口** **AXB模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于Python 3.7.0版本，要求Python 3.0及以上版本。
引用库	requests 2.18.1（仅“获取录音文件下载地址接口”引用） 1. 请自行下载安装Python 3.x，并完成环境配置。 2. 打开命令行窗口，执行pip install requests命令。 3. 执行pip list查看安装结果。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXB 模式绑定接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import json
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret
relationNum = '+86180****0001' #X号码(隐私号码)
callerNum = '+86186****5678' #A号码
calleeNum = '+86186****5679' #B号码

'''
选填,各参数要求请参考"AXB模式绑定接口"
```

```
'''
# areaCode = '0755' #需要绑定的X号码对应的城市码
# callDirection = 0 #允许呼叫的方向
# duration = 86400 #绑定关系保持时间,到期后会被系统自动解除绑定关系
# recordFlag = 'false' #是否需要针对该绑定关系产生的所有通话录音
# recordHintTone = 'recordHintTone.wav' #设置录音提示音
# maxDuration = 60 #设置允许单次通话进行的最长时间,通话时间从接通被叫的时刻开始计算
# privateSms = 'true' #设置该绑定关系是否支持短信功能

# callerHintTone = 'callerHintTone.wav' #设置A拨打X号码时的通话前等待音
# calleeHintTone = 'calleeHintTone.wav' #设置B拨打X号码时的通话前等待音
# preVoice = {
#     'callerHintTone': callerHintTone,
#     'calleeHintTone': calleeHintTone
# };

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey,
digestBase64, nonce, now);

def main():
    # 请求Body,可按需删除选填参数
    jsonData = json.dumps({
        'relationNum':relationNum,
#         'areaCode':areaCode,
        'callerNum':callerNum,
        'calleeNum':calleeNum,
#         'callDirection':callDirection,
#         'duration':duration,
#         'recordFlag':recordFlag,
#         'recordHintTone':recordHintTone,
#         'maxDuration':maxDuration,
#         'privateSms':privateSms,
#         'preVoice':preVoice
    }).encode('ascii')

    req = urllib.request.Request(url=realUrl, data=jsonData, method='POST') #请求方法为POST
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        fo.write('绑定请求数据：' + jsonData.decode('utf-8') + '\n') #绑定请求参数记录到本地文件,方便定位问题
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
        fo.write('绑定结果：' + str(r.read().decode('utf-8')) + '\n') #绑定ID很重要,请记录到本地文件,方便后续修
改绑定关系及解绑
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

AXB 模式解绑接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AXB模式解绑接口"
subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
'''
subscriptionId = '****' #指定"AXB模式绑定接口"返回的绑定ID进行解绑
relationNum = '+86180****0001' #指定X号码(隐私号码)进行解绑

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'subscriptionId':subscriptionId,
        'relationNum':relationNum
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    req = urllib.request.Request(url=fullUrl, method='DELETE') #请求方法为DELETE
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        print(formData) #打印请求数据
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

AXB 模式绑定信息修改接口

```
# -*- coding: utf-8 -*-
import time
import uuid
```

```
import hashlib
import base64
import json
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

subscriptionId = '0167ecc9-bfb6-4eec-b671-a7dab2ba78c' #必填,指定"AXB模式绑定接口"返回的绑定ID进行修改

'''
选填,各参数要求请参考"AXB模式绑定信息修改接口"
'''
callerNum = '+86186****5678' #A号码
calleeNum = '+86186****5679' #B号码
# callDirection = 0 #允许呼叫的方向
# duration = 86400 #绑定关系保持时间,到期后会被系统自动解除绑定关系
# maxDuration = 60 #设置允许单次通话进行的最长时间,通话时间从接通被叫的时刻开始计算
# privateSms = 'true' #设置该绑定关系是否支持短信功能

# callerHintTone = 'callerHintTone.wav' #设置A拨打X号码时的通话前等待音
# calleeHintTone = 'calleeHintTone.wav' #设置B拨打X号码时的通话前等待音
# preVoice = {
#     'callerHintTone': callerHintTone,
#     'calleeHintTone': calleeHintTone
# };

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求Body,可按需删除选填参数
    jsonData = json.dumps({
        'subscriptionId':subscriptionId,
        'callerNum':callerNum,
        'calleeNum':calleeNum,
#         'callDirection':callDirection,
#         'duration':duration,
#         'maxDuration':maxDuration,
#         'privateSms':privateSms,
#         'preVoice':preVoice
    }).encode('ascii')

    req = urllib.request.Request(url=realUrl, data=jsonData, method='PUT') #请求方法为PUT
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        print(jsonData.decode('utf-8')) #打印请求数据
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
    except urllib.error.HTTPError as e:
```

```
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

AXB 模式绑定信息查询接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AXB模式绑定信息查询接口"
subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
'''
subscriptionId = '****' #指定"AXB模式绑定接口"返回的绑定ID进行查询
relationNum = '+86180****0001' #指定X号码(隐私号码)进行查询
# pageIndex = 1 #查询的分页索引,从1开始编号
# pageSize = 20 #查询的分页大小,即每次查询返回多少条数据

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数,可按需删除选填参数
    formData = urllib.parse.urlencode({
        'subscriptionId':subscriptionId,
        'relationNum':relationNum,
#         'pageIndex':pageIndex,
#         'pageSize':pageSize
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    req = urllib.request.Request(url=fullUrl, method='GET') #请求方法为GET
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
        fo.write('绑定查询结果：' + str(r.read().decode('utf-8')) + '\n') #查询结果,记录到本地文件
    except urllib.error.HTTPError as e:
```

```
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

获取录音文件下载地址接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import urllib
import requests #需要先使用pip install requests命令安装依赖
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

# 必填,通过"话单通知接口"获取
recordDomain = '****.com' #录音文件存储的服务器域名
fileName = '****.wav' #录音文件名

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'recordDomain':recordDomain,
        'fileName':fileName
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    # 请求Headers参数
    header = {
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret),
        'Content-Type': 'application/json;charset=UTF-8'
    }

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        r = requests.get(fullUrl, headers=header, allow_redirects=False, verify=False) #发送请求
        if(301 == r.status_code):
            print(r.status_code) #打印响应结果
            print(r.headers['Location'])
            fo.write('获取录音文件下载地址：' + r.headers['Location'] + '\n')
        else:
            print(r.status_code) #打印响应结果
            print(r.text)
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
```

```
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

呼叫事件通知接口

```
# -*- coding: utf-8 -*-
'''
呼叫事件通知
客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
'''
import json

#呼叫事件通知样例
jsonBody = json.dumps({
    'eventType': 'disconnect',
    'statusInfo': {
        'sessionId': '1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com',
        'timestamp': '2019-01-23 09:16:41',
        'caller': '+86138****0021',
        'called': '+86138****7021',
        'stateCode': 0,
        'stateDesc': 'The user releases the call.',
        'subscriptionId': '****'
    }
}).encode('ascii')

print(jsonBody)

'''
呼叫事件通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onCallEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj
    eventType = jsonObj['eventType'] #通知事件类型

    if ('fee' == eventType):
        print('EventType error: ' + eventType)
        return

    if ('statusInfo' not in jsonObj):
        print('param error: no statusInfo.')
        return
    statusInfo = jsonObj['statusInfo'] #呼叫状态事件信息

    print('eventType: ' + eventType) #打印通知事件类型
    #callin：呼入事件
    if ('callin' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #callout：呼出事件
    if ('callout' == eventType):
```



```
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #alerting：振铃事件
    if ('alerting' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #answer：应答事件
    if ('answer' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #disconnect：挂机事件
    if ('disconnect' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'stateCode': 通话挂机的原因值
        'stateDesc': 通话挂机的原因值的描述
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return

def main():
    onCallEvent(jsonBody) #呼叫事件处理

if __name__ == '__main__':
    main()
```

话单通知接口

```
# -*- coding: utf-8 -*-
'''
话单通知
客户平台收到隐私保护通话平台的话单通知的接口通知
```

```
'''
import json

#话单通知样例
jsonBody = json.dumps({
    'eventType': 'fee',
    'feeLst': [
        {
            'direction': 1,
            'spId': '****',
            'appKey': '********',
            'icid': 'ba171f34e6953fcd751edc77127748f4.3757223714.337238282.9',
            'bindNum': '+86138****0022',
            'sessionId': '1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com',
            'subscriptionId': '****',
            'callerNum': '+86138****0021',
            'calleeNum': '+86138****0022',
            'fwdDisplayNum': '+86138****0022',
            'fwdDstNum': '+86138****7021',
            'callInTime': '2019-01-23 09:15:14',
            'fwdStartTime': '2019-01-23 09:15:15',
            'fwdAlertingTime': '2019-01-23 09:15:21',
            'fwdAnswerTime': '2019-01-23 09:15:36',
            'callEndTime': '2019-01-23 09:16:41',
            'fwdUnaswRsn': 0,
            'ulFailReason': 0,
            'sipStatusCode': 0,
            'callOutUnaswRsn': 0,
            'recordFlag': 1,
            'recordStartTime': '2019-01-23 09:15:37',
            'recordDomain': '****.com',
            'recordBucketName': 'sp-********',
            'recordObjectName': '********.wav',
            'ttsPlayTimes': 0,
            'ttsTransDuration': 0,
            'mptyId': '****',
            'serviceType': '004',
            'hostName': 'callenabler246.huaweicaas.com'
        }
    ]
}).encode('ascii')

print(jsonBody)

'''
话单通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onFeeEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj
    eventType = jsonObj['eventType'] #通知事件类型

    if ('fee' != eventType):
        print('EventType error: ' + eventType)
        return

    if ('feeLst' not in jsonObj):
        print('param error: no feeLst.');
        return
    feeLst = jsonObj['feeLst'] #呼叫话单事件信息

    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'direction': 通话的呼叫方向
    'spId': 客户的云服务账号
    'appKey': 商户应用的AppKey
```



```
    'icid': 呼叫记录的唯一标识
    'bindNum': 隐私保护号码
    'sessionId': 通话链路的唯一标识
    'callerNum': 主叫号码
    'calleeNum': 被叫号码
    'fwdDisplayNum': 转接呼叫时的显示号码
    'fwdDstNum': 转接呼叫时的转接号码
    'callInTime': 呼入的开始时间
    'fwdStartTime': 转接呼叫操作的开始时间
    'fwdAlertingTime': 转接呼叫操作后的振铃时间
    'fwdAnswerTime': 转接呼叫操作后的应答时间
    'callEndTime': 呼叫结束时间
    'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
    'failTime': 呼入,呼出的失败时间
    'ulFailReason': 通话失败的拆线点
    'sipStatusCode': 呼入,呼出的失败SIP状态码
    'recordFlag': 录音标识
    'recordStartTime': 录音开始时间
    'recordObjectName': 录音文件名
    'recordBucketName': 录音文件所在的目录名
    'recordDomain': 存放录音文件的域名
    'serviceType': 携带呼叫的业务类型信息
    'hostName': 话单生成的服务器设备对应的主机名
    'subscriptionId': 绑定关系ID
    '''
    #短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (len(feeLst) > 1):
        for loop in feeLst:
            if ('sessionId' in loop):
                print('sessionId: ' + loop['sessionId'])
    elif(len(feeLst) == 1):
        if ('sessionId' in feeLst[0]):
            print('sessionId: ' + feeLst[0]['sessionId'])
    else:
        print('feeLst error: no element.');

def main():
    onFeeEvent(jsonBody) #话单处理

if __name__ == '__main__':
    main()
```

短信通知接口

```
# -*- coding: utf-8 -*-
'''
短信通知
客户平台收到隐私保护通话平台的短信通知的接口通知
'''
import json

#短信通知样例
jsonBody = json.dumps({
    'appKey': '****',
    'smsEvent': {
        'smsIdentifier': '****',
        'notificationMode': 'Block',
        'calling': '+86138****0001',
        'virtualNumber': '+86138****0000',
        'event': 'TextSMS',
        'timeStamp': '2018-09-13T09:46:16.023Z'
    }
}).encode('ascii')

print(jsonBody)

'''
短信通知
@see: 详细内容以接口文档为准
```



```
@param param: jsonBody
@return:
'''
def onSmsEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj

    if ('smsEvent' not in jsonObj):
        print('param error: no smsEvent.')
        return

    #print(jsonObj['appKey']) #商户应用的AppKey

    smsEvent = jsonObj['smsEvent'] #短信通知信息

    '''
    Example: 此处已解析notificationMode为例,请按需解析所需参数并自行实现相关处理

    'smsIdentifier': 短信唯一标识
    'notificationMode': 通知模式
    'calling': 真实发送方号码
    'called': 真实接收方号码
    'virtualNumber': 隐私号码(X号码)
    'event': 短信状态事件
    'timeStamp': 短信事件发生的系统时间戳,UTC时间
    'subscriptionId': 绑定ID
    'smsContent': 用户发送的短信内容
    '''
    if ('notificationMode' in smsEvent):
        if ('Block' == smsEvent['notificationMode']):
            #收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
            actions = {
                'operation': 'vNumberRoute', #操作类型:转发短信/丢弃短信
                'message': {
                    'called': '+86138****7022', #真实接收方号码
                    'calling': '+86138****7021' #真实发送方号码
                }
            }
            resp = json.dumps({'actions': [actions]}) #Block模式响应消息
            print(resp);
        elif ('Notify' == smsEvent['notificationMode']):
            #收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
            #statusCode = 200;
            print('This is AXB sms Notify mode.');
        else:
            print('notificationMode param error.');

def main():
    onSmsEvent(jsonBody); #短信通知处理

if __name__ == '__main__':
    main()
```

3.3.2 AX 模式

样例	**AX模式绑定接口** **AX模式解绑接口** **AX模式绑定信息修改接口** **AX模式绑定信息查询接口** **AX模式设置临时被叫接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于Python 3.7.0版本，要求Python 3.0及以上版本。
引用库	requests 2.18.1（仅“获取录音文件下载地址接口”引用） 1. 请自行下载安装Python 3.x，并完成环境配置。 2. 打开命令行窗口，执行pip install requests命令。 3. 执行pip list查看安装结果。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AX 模式绑定接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import json
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret
origNum = '+86186****5678' #A号码
privateNum = '+86180****0001' #X号码(隐私号码)

'''
```

```
选填,各参数要求请参考"AX模式绑定接口"
'''
privateNumType = 'mobile-virtual' #固定为mobile-virtual
# areaCode = '0755' #需要绑定的X号码对应的城市码
# recordFlag = 'true' #是否需要针对该绑定关系产生的所有通话录音
# recordHintTone = 'recordHintTone.wav' #设置录音提示音
calleeNumDisplay = '0' #设置非A用户呼叫X时,A接到呼叫时的主显号码
# privateSms = 'true' #设置该绑定关系是否支持短信功能

# callerHintTone = 'callerHintTone.wav' #设置A拨打X号码时的通话前等待音
# calleeHintTone = 'calleeHintTone.wav' #设置非A用户拨打X号码时的通话前等待音
# preVoice = {
#     'callerHintTone': callerHintTone,
#     'calleeHintTone': calleeHintTone
# }

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey,
digestBase64, nonce, now);

def main():
    # 请求Body,可按需删除选填参数
    jsonData = json.dumps({
        'origNum':origNum,
        'privateNum':privateNum,
        'privateNumType':privateNumType,
#         'areaCode':areaCode,
#         'recordFlag':recordFlag,
#         'recordHintTone':recordHintTone,
        'calleeNumDisplay':calleeNumDisplay,
#         'privateSms':privateSms,
#         'preVoice':preVoice
    }).encode('ascii')

    req = urllib.request.Request(url=realUrl, data=jsonData, method='POST') #请求方法为POST
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        fo.write('绑定请求数据：' + jsonData.decode('utf-8') + '\n') #绑定请求参数记录到本地文件,方便定位问题
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
        fo.write('绑定结果：' + str(r.read().decode('utf-8')) + '\n') #绑定ID很重要,请记录到本地文件,方便后续修
改绑定关系及解绑
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

AX 模式解绑接口

```
# -*- coding: utf-8 -*-
import time
```

```
import uuid
import hashlib
import base64
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AX模式解绑接口"
subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
'''
origNum = '+86186****5678' #A号码
privateNum = '+86180****0001' #X号码
# subscriptionId = '****' #指定"AX模式绑定接口"返回的绑定ID进行解绑

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'origNum':origNum,
        'privateNum':privateNum,
#         'subscriptionId':subscriptionId
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    req = urllib.request.Request(url=fullUrl, method='DELETE') #请求方法为DELETE
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        print(formData) #打印请求数据
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

AX 模式绑定信息修改接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
```

```
import json
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AX模式绑定信息修改接口"
subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
'''
origNum = '+86186****5678' #A号码
privateNum = '+86180****0001' #X号码
# subscriptionId = '****' #指定"AX模式绑定接口"返回的绑定ID进行修改

privateSms = 'true' #必填,修改该绑定关系是否支持短信功能

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求Body,可按需删除选填参数
    jsonData = json.dumps({
        'origNum':origNum,
        'privateNum':privateNum,
#         'subscriptionId':subscriptionId,
        'privateSms':privateSms
    }).encode('ascii')

    req = urllib.request.Request(url=realUrl, data=jsonData, method='PUT') #请求方法为PUT
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        print(jsonData.decode('utf-8')) #打印请求数据
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

AX 模式绑定信息查询接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import ssl
```

```
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AX模式绑定信息查询接口"
subscriptionId和origNum为二选一关系,两者都携带时以subscriptionId为准
'''
origNum = '+86186****5678' #指定A号码进行查询
subscriptionId = '****' #指定"AX模式绑定接口"返回的绑定ID进行查询

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'origNum':origNum,
        'subscriptionId':subscriptionId
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    req = urllib.request.Request(url=fullUrl, method='GET') #请求方法为GET
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
        fo.write('绑定查询结果：' + str(r.read().decode('utf-8')) + '\n') #查询结果,记录到本地文件
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

AX 模式设置临时被叫接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import json
import ssl
import urllib.request
```

```
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/privatenumber/calleenumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AX模式设置临时被叫接口"
subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
'''
origNum = '+86186****5678' #A号码
privateNum = '+86180****0001' #X号码
# subscriptionId = '****' #指定“AX模式绑定接口”返回的绑定ID设置临时被叫

calleeNum = '+86186****5679' #必填,本次呼叫的真实被叫号码

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求Body,可按需删除选填参数
    jsonData = json.dumps({
        'origNum':origNum,
        'privateNum':privateNum,
#         'subscriptionId':subscriptionId,
        'calleeNum':calleeNum
    }).encode('ascii')

    req = urllib.request.Request(url=realUrl, data=jsonData, method='PUT') #请求方法为PUT
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        print(jsonData.decode('utf-8')) #打印请求数据
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

获取录音文件下载地址接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import urllib
import requests #需要先使用pip install requests命令安装依赖
import hmac
```

```
from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

# 必填,通过"话单通知接口"获取
recordDomain = '****.com' #录音文件存储的服务器域名
fileName = '****.wav' #录音文件名

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'recordDomain':recordDomain,
        'fileName':fileName
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    # 请求Headers参数
    header = {
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret),
        'Content-Type': 'application/json;charset=UTF-8'
    }

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        r = requests.get(fullUrl, headers=header, allow_redirects=False, verify=False) #发送请求
        if(301 == r.status_code):
            print(r.status_code) #打印响应结果
            print(r.headers['Location'])
            fo.write('获取录音文件下载地址：' + r.headers['Location'] + '\n')
        else:
            print(r.status_code) #打印响应结果
            print(r.text)
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

呼叫事件通知接口

```
# -*- coding: utf-8 -*-
'''
呼叫事件通知
客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
'''
import json

#呼叫事件通知样例
jsonBody = json.dumps({
    'eventType': 'disconnect',
    'statusInfo': {
```

```
        'sessionId': '1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com',
        'timestamp': '2019-01-24 02:30:22',
        'caller': '+86138****0022',
        'called': '+86138****7021',
        'stateCode': 0,
        'stateDesc': 'The user releases the call.',
        'subscriptionId': '****'
    }
}).encode('ascii')

print(jsonBody)

'''
呼叫事件通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onCallEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj
    eventType = jsonObj['eventType'] #通知事件类型

    if ('fee' == eventType):
        print('EventType error: ' + eventType)
        return

    if ('statusInfo' not in jsonObj):
        print('param error: no statusInfo.')
        return
    statusInfo = jsonObj['statusInfo'] #呼叫状态事件信息

    print('eventType: ' + eventType) #打印通知事件类型
    #callin：呼入事件
    if ('callin' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #callout：呼出事件
    if ('callout' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #alerting：振铃事件
    if ('alerting' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
```

```
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #answer：应答事件
    if ('answer' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return
    #disconnect：挂机事件
    if ('disconnect' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'stateCode': 通话挂机的原因值
        'stateDesc': 通话挂机的原因值的描述
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return

def main():
    onCallEvent(jsonBody) #呼叫事件处理

if __name__ == '__main__':
    main()
```

话单通知接口

```
# -*- coding: utf-8 -*-
'''
话单通知
客户平台收到隐私保护通话平台的话单通知的接口通知
'''
import json

#话单通知样例
jsonBody = json.dumps({
    'eventType': 'fee',
    'feeLst': [
        {
            'direction': 1,
            'spId': '****',
            'appKey': '****',
            'icid': 'ba171f34e6953fcd751edc77127748f4.3757285803.338666679.5',
            'bindNum': '+86138****0022',
            'sessionId': '1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com',
            'subscriptionId': '****',
            'callerNum': '+86138****0021',
            'calleeNum': '+86138****0022',
            'fwdDisplayNum': '+86138****0022',
            'fwdDstNum': '+86138****7021',
            'callInTime': '2019-01-24 02:30:03',
            'fwdStartTime': '2019-01-24 02:30:03',
```

```
            'fwdAlertingTime': '2019-01-24 02:30:04',
            'fwdAnswerTime': '2019-01-24 02:30:06',
            'callEndTime': '2019-01-24 02:30:22',
            'fwdUnaswRsn': 0,
            'ulFailReason': 0,
            'sipStatusCode': 0,
            'callOutUnaswRsn': 0,
            'recordFlag': 1,
            'recordStartTime': '2019-01-24 02:30:06',
            'recordDomain': '****.com',
            'recordBucketName': '****',
            'recordObjectName': '****.wav',
            'ttsPlayTimes': 0,
            'ttsTransDuration': 0,
            'mptyId': '****',
            'serviceType': '003',
            'hostName': 'callenabler246.huaweicaas.com'
        }
    ]
}).encode('ascii')

print(jsonBody)

'''
话单通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onFeeEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj
    eventType = jsonObj['eventType'] #通知事件类型

    if ('fee' != eventType):
        print('EventType error: ' + eventType)
        return

    if ('feeLst' not in jsonObj):
        print('param error: no feeLst.');
        return
    feeLst = jsonObj['feeLst'] #呼叫话单事件信息

    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'direction': 通话的呼叫方向
    'spId': 客户的云服务账号
    'appKey': 商户应用的AppKey
    'icid': 呼叫记录的唯一标识
    'bindNum': 隐私保护号码
    'sessionId': 通话链路的唯一标识
    'callerNum': 主叫号码
    'calleeNum': 被叫号码
    'fwdDisplayNum': 转接呼叫时的显示号码
    'fwdDstNum': 转接呼叫时的转接号码
    'callInTime': 呼入的开始时间
    'fwdStartTime': 转接呼叫操作的开始时间
    'fwdAlertingTime': 转接呼叫操作后的振铃时间
    'fwdAnswerTime': 转接呼叫操作后的应答时间
    'callEndTime': 呼叫结束时间
    'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
    'failTime': 呼入,呼出的失败时间
    'ulFailReason': 通话失败的拆线点
    'sipStatusCode': 呼入,呼出的失败SIP状态码
    'recordFlag': 录音标识
    'recordStartTime': 录音开始时间
    'recordObjectName': 录音文件名
    'recordBucketName': 录音文件所在的目录名
    'recordDomain': 存放录音文件的域名
```

```
    'serviceType': 携带呼叫的业务类型信息
    'hostName': 话单生成的服务器设备对应的主机名
    'subscriptionId': 绑定关系ID
    '''
    #短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (len(feeLst) > 1):
        for loop in feeLst:
            if ('sessionId' in loop):
                print('sessionId: ' + loop['sessionId'])
    elif(len(feeLst) == 1):
        if ('sessionId' in feeLst[0]):
            print('sessionId: ' + feeLst[0]['sessionId'])
    else:
        print('feeLst error: no element.');

def main():
    onFeeEvent(jsonBody) #话单处理

if __name__ == '__main__':
    main()
```

短信通知接口

```
# -*- coding: utf-8 -*-
'''
短信通知
客户平台收到隐私保护通话平台的短信通知的接口通知
'''
import json

#短信通知样例
jsonBody = json.dumps({
    'appKey': '****',
    'smsEvent': {
        'smsIdentifier': '****',
        'notificationMode': 'Block',
        'calling': '+86138****0001',
        'virtualNumber': '+86138****0000',
        'event': 'TextSMS',
        'timeStamp': '2018-09-13T09:46:16.023Z'
    }
}).encode('ascii')

print(jsonBody)

'''
短信通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onSmsEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj

    if ('smsEvent' not in jsonObj):
        print('param error: no smsEvent.')
        return

    #print(jsonObj['appKey']) #商户应用的AppKey

    smsEvent = jsonObj['smsEvent'] #短信通知信息

    '''
    Example: 此处已解析notificationMode为例,请按需解析所需参数并自行实现相关处理

    'smsIdentifier': 短信唯一标识
    'notificationMode': 通知模式
    'calling': 真实发送方号码
    'called': 真实接收方号码
```

```
    'virtualNumber': 隐私号码(X号码)
    'event': 短信状态事件
    'timeStamp': 短信事件发生的系统时间戳,UTC时间
    'subscriptionId': 绑定ID
    'smsContent': 用户发送的短信内容
    '''
    if ('notificationMode' in smsEvent):
        if ('Block' == smsEvent['notificationMode']):
            #收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
            actions = {
                'operation': 'vNumberRoute', #操作类型:转发短信/丢弃短信
                'message': {
                    'called': '+86138****7022', #真实接收方号码
                    'calling': '+86138****7021' #真实发送方号码
                }
            }
            resp = json.dumps({'actions': [actions]}) #Block模式响应消息
            print('resp: ' + resp);
        elif ('Notify' == smsEvent['notificationMode']):
            #收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
            #statusCode = 200;
            print('This is AX sms Notify mode.');
        else:
            print('notificationMode param error.');

def main():
    onSmsEvent(jsonBody); #短信通知处理

if __name__ == '__main__':
    main()
```

3.3.3 AXE 模式

样例	**AXE模式绑定接口** **AXE模式解绑接口** **AXE模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口**
环境要求	基于Python 3.7.0版本，要求Python 3.0及以上版本。
引用库	requests 2.18.1（仅“获取录音文件下载地址接口”引用） 1. 请自行下载安装Python 3.x，并完成环境配置。 2. 打开命令行窗口，执行pip install requests命令。 3. 执行pip list查看安装结果。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXE 模式绑定接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import json
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret
virtualNum = '+86180****0001' #AXE中的X号码
bindNum = '+86186****5678' #AXE中的A号码

'''
选填,各参数要求请参考"AXE模式绑定接口"
'''
# areaCode = '0755' #需要绑定的X号码对应的城市码
# displayNumMode = '0' #非A用户呼叫X号码时，A看到的主显号码
# recordFlag = 'false' #是否需要针对该绑定关系产生的所有通话录音
# recordHintTone = 'recordHintTone.wav' #设置录音提示音
# callbackTone = 'callbackTone.wav' #A呼叫X不存在回呼记录的提示音
# callbackNum = '+86180****0021' #A呼叫X不存在回呼记录的转接号码
# bindExpiredTime = 24 #绑定关系的有效时间
# callbackExpiredTime = 24 #回呼记录有效时间
# userData = 'abcdefg' #用户自定义数据

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求Body,可按需删除选填参数
    jsonData = json.dumps({
        'virtualNum':virtualNum,
        'bindNum':bindNum,
#         'areaCode':areaCode,
#         'displayNumMode':displayNumMode,
#         'recordFlag':recordFlag,
#         'recordHintTone':recordHintTone,
#         'callbackTone':callbackTone,
#         'callbackNum':callbackNum,
#         'bindExpiredTime':bindExpiredTime,
#         'callbackExpiredTime':callbackExpiredTime,
```

```
#         'userData':userData
    }).encode('ascii')

    req = urllib.request.Request(url=realUrl, data=jsonData, method='POST') #请求方法为POST
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        fo.write('绑定请求数据：' + jsonData.decode('utf-8') + '\n') #绑定请求参数记录到本地文件,方便定位问题
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
        fo.write('绑定结果：' + str(r.read().decode('utf-8')) + '\n') #绑定ID很重要,请记录到本地文件,方便后续修
改绑定关系及解绑
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

AXE 模式解绑接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0' #APP接入地址+接口访
问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AXE模式解绑接口"
subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
'''
subscriptionId = '****' #指定"AXE模式绑定接口"返回的绑定ID进行解绑
virtualNum = '+86180****0001' #AXE中的X号码
extendNum = '1234' #AXE中的E号码

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey,
digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'subscriptionId':subscriptionId,
        'virtualNum':virtualNum,
```

```
        'extendNum':extendNum
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    req = urllib.request.Request(url=fullUrl, method='DELETE') #请求方法为DELETE
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        print(formData) #打印请求数据
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)

if __name__ == '__main__':
    main()
```

AXE 模式绑定信息查询接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import ssl
import urllib.request
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

'''
选填,各参数要求请参考"AXE模式绑定信息查询接口"
subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
'''
subscriptionId = '****' #指定"AXE模式绑定接口"返回的绑定ID进行查询
virtualNum = '+86180****0001' #AXE中的X号码
extendNum = '1234' #AXE中的E号码

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数,可按需删除选填参数
    formData = urllib.parse.urlencode({
        'subscriptionId':subscriptionId,
        'virtualNum':virtualNum,
        'extendNum':extendNum,
    })
    #完整请求地址
```

```
    fullUrl = realUrl + '?' + formData

    req = urllib.request.Request(url=fullUrl, method='GET') #请求方法为GET
    # 请求Headers参数
    req.add_header('Authorization', 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"')
    req.add_header('X-AKSK', buildAKSKHeader(APP_KEY, APP_SECRET))
    req.add_header('Content-Type', 'application/json;charset=UTF-8')

    # 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
    ssl._create_default_https_context = ssl._create_unverified_context

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        r = urllib.request.urlopen(req) #发送请求
        print(r.read().decode('utf-8')) #打印响应结果
        fo.write('绑定查询结果：' + str(r.read().decode('utf-8')) + '\n') #查询结果,记录到本地文件
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

获取录音文件下载地址接口

```
# -*- coding: utf-8 -*-
import time
import uuid
import hashlib
import base64
import urllib
import requests #需要先使用pip install requests命令安装依赖
import hmac

from hashlib import sha256
# 必填,请参考"开发准备"获取如下数据,替换为实际值
realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0' #APP接入地址+接口访问URI
APP_KEY = "a1********" #APP_Key
APP_SECRET = "cfc8********" #APP_Secret

# 必填,通过"话单通知接口"获取
recordDomain = '****.com' #录音文件存储的服务器域名
fileName = '****.wav' #录音文件名

def buildAKSKHeader(appKey, appSecret):
    now = time.strftime('%Y-%m-%dT%H:%M:%SZ') #Created
    nonce = str(uuid.uuid4()).replace('-','') #Nonce
    digist = hmac.new(appSecret.encode(), (nonce + now).encode(), digestmod=sha256).digest()
    digestBase64 = base64.b64encode(digist).decode() #PasswordDigest
    return 'UsernameToken Username="{}",PasswordDigest="{}",Nonce="{}",Created="{}"'.format(appKey, digestBase64, nonce, now);

def main():
    # 请求URL参数
    formData = urllib.parse.urlencode({
        'recordDomain':recordDomain,
        'fileName':fileName
    })
    #完整请求地址
    fullUrl = realUrl + '?' + formData

    # 请求Headers参数
    header = {
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
```

```
        'X-AKSK': buildAKSKHeader(appKey, appSecret),
        'Content-Type': 'application/json;charset=UTF-8'
    }

    try:
        fo = open('bind_data.txt', 'a', encoding='utf-8') #打开本地文件
        r = requests.get(fullUrl, headers=header, allow_redirects=False, verify=False) #发送请求
        if(301 == r.status_code):
            print(r.status_code) #打印响应结果
            print(r.headers['Location'])
            fo.write('获取录音文件下载地址：' + r.headers['Location'] + '\n')
        else:
            print(r.status_code) #打印响应结果
            print(r.text)
    except urllib.error.HTTPError as e:
        print(e.code)
        print(e.read().decode('utf-8')) #打印错误信息
    except urllib.error.URLError as e:
        print(e.reason)
    finally:
        fo.close() #关闭文件

if __name__ == '__main__':
    main()
```

呼叫事件通知接口

```
# -*- coding: utf-8 -*-
'''
呼叫事件通知
客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
'''
import json

#呼叫事件通知样例
jsonBody = json.dumps({
    'eventType': 'disconnect',
    'statusInfo': {
        'sessionId': '1202_1051_4294967295_20190124070250@callenabler246.huaweicaas.com',
        'timestamp': '2019-01-24 07:03:28',
        'caller': '+86138****0022',
        'called': '+86138****7021',
        'stateCode': 0,
        'stateDesc': 'The user releases the call.',
        'subscriptionId': '********'
    }
}).encode('ascii')

print(jsonBody)

'''
呼叫事件通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onCallEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj
    eventType = jsonObj['eventType'] #通知事件类型

    if ('fee' == eventType):
        print('EventType error: ' + eventType)
        return

    if ('statusInfo' not in jsonObj):
        print('param error: no statusInfo.')
        return
    statusInfo = jsonObj['statusInfo'] #呼叫状态事件信息
```

```
print('eventType: ' + eventType) #打印通知事件类型
#callin：呼入事件
if ('callin' == eventType):
    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
    'sessionId': 通话链路的标识ID
    'caller': 主叫号码
    'called': 被叫号码
    '''
    if ('sessionId' in statusInfo):
        print('sessionId: ' + statusInfo['sessionId'])
    return
#collectInfo：放音收号结果事件,仅AXE模式下的A被叫场景携带
if ('collectInfo' == eventType):
    '''
    Example: 此处以解析digitInfo为例,请按需解析所需参数并自行实现相关处理

    'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
    'sessionId': 通话链路的标识ID
    'digitInfo': AXE场景中携带收号结果(即用户输入的数字)
    '''
    if ('digitInfo' in statusInfo):
        print('digitInfo: ' + statusInfo['digitInfo'])
    return
#callout：呼出事件
if ('callout' == eventType):
    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
    'sessionId': 通话链路的标识ID
    'caller': 主叫号码
    'called': 被叫号码
    'subscriptionId': 绑定关系ID
    '''
    if ('sessionId' in statusInfo):
        print('sessionId: ' + statusInfo['sessionId'])
    return
#alerting：振铃事件
if ('alerting' == eventType):
    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
    'sessionId': 通话链路的标识ID
    'caller': 主叫号码
    'called': 被叫号码
    'subscriptionId': 绑定关系ID
    '''
    if ('sessionId' in statusInfo):
        print('sessionId: ' + statusInfo['sessionId'])
    return
#answer：应答事件
if ('answer' == eventType):
    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
    'sessionId': 通话链路的标识ID
    'caller': 主叫号码
    'called': 被叫号码
    'subscriptionId': 绑定关系ID
    '''
    if ('sessionId' in statusInfo):
        print('sessionId: ' + statusInfo['sessionId'])
    return
#disconnect：挂机事件
```

```
    if ('disconnect' == eventType):
        '''
        Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

        'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        'sessionId': 通话链路的标识ID
        'caller': 主叫号码
        'called': 被叫号码
        'stateCode': 通话挂机的原因值
        'stateDesc': 通话挂机的原因值的描述
        'subscriptionId': 绑定关系ID
        '''
        if ('sessionId' in statusInfo):
            print('sessionId: ' + statusInfo['sessionId'])
        return

def main():
    onCallEvent(jsonBody) #呼叫事件处理

if __name__ == '__main__':
    main()
```

话单通知接口

```
# -*- coding: utf-8 -*-
'''
话单通知
客户平台收到隐私保护通话平台的话单通知的接口通知
'''
import json

#话单通知样例
jsonBody = json.dumps({
    'eventType': 'fee',
    'feeLst': [
        {
            'direction': 0,
            'spId': '****',
            'appKey': '********',
            'icid': 'ba171f34e6953fcd751edc77127748f4.3757289590.338833305.5',
            'bindNum': '+86138****0022',
            'sessionId': '1201_11275_4294967295_20190124033310@callenabler246.huaweicaas.com',
            'subscriptionId': '********',
            'callerNum': '+86138****7021',
            'calleeNum': '+86138****0022',
            'fwdDisplayNum': '+86138****0022',
            'fwdDstNum': '+86138****0021',
            'callInTime': '2019-01-24 03:33:10',
            'fwdStartTime': '2019-01-24 03:33:16',
            'fwdAlertingTime': '2019-01-24 03:33:19',
            'fwdAnswerTime': '2019-01-24 03:33:28',
            'callEndTime': '2019-01-24 03:33:57',
            'fwdUnaswRsn': 0,
            'ulFailReason': 0,
            'sipStatusCode': 0,
            'callOutUnaswRsn': 0,
            'recordFlag': 1,
            'recordStartTime': '2019-01-24 03:33:28',
            'recordDomain': '****.com',
            'recordBucketName': '****',
            'recordObjectName': '********.wav',
            'ttsPlayTimes': 0,
            'ttsTransDuration': 0,
            'mptyId': '********',
            'serviceType': '005',
            'hostName': 'callenabler246.huaweicaas.com',
            'extendNumber': '02'
        }
    ]
```

```
}).encode('ascii')

print(jsonBody)

'''
话单通知
@see: 详细内容以接口文档为准
@param param: jsonBody
@return:
'''
def onFeeEvent(jsonBody):
    jsonObj = json.loads(jsonBody) #将通知消息解析为jsonObj
    eventType = jsonObj['eventType'] #通知事件类型

    if ('fee' != eventType):
        print('EventType error: ' + eventType)
        return

    if ('feeLst' not in jsonObj):
        print('param error: no feeLst.');
        return
    feeLst = jsonObj['feeLst'] #呼叫话单事件信息

    '''
    Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理

    'direction': 通话的呼叫方向
    'spId': 客户的云服务账号
    'appKey': 商户应用的AppKey
    'icid': 呼叫记录的唯一标识
    'bindNum': 隐私保护号码
    'sessionId': 通话链路的唯一标识
    'callerNum': 主叫号码
    'calleeNum': 被叫号码
    'fwdDisplayNum': 转接呼叫时的显示号码
    'fwdDstNum': 转接呼叫时的转接号码
    'callInTime': 呼入的开始时间
    'fwdStartTime': 转接呼叫操作的开始时间
    'fwdAlertingTime': 转接呼叫操作后的振铃时间
    'fwdAnswerTime': 转接呼叫操作后的应答时间
    'callEndTime': 呼叫结束时间
    'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
    'failTime': 呼入,呼出的失败时间
    'ulFailReason': 通话失败的拆线点
    'sipStatusCode': 呼入,呼出的失败SIP状态码
    'recordFlag': 录音标识
    'recordStartTime': 录音开始时间
    'recordObjectName': 录音文件名
    'recordBucketName': 录音文件所在的目录名
    'recordDomain': 存放录音文件的域名
    'serviceType': 携带呼叫的业务类型信息
    'hostName': 话单生成的服务器设备对应的主机名
    'subscriptionId': 绑定关系ID
    'extendNum': 分机号E
    '''
    #短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (len(feeLst) > 1):
        for loop in feeLst:
            if ('sessionId' in loop):
                print('sessionId: ' + loop['sessionId'])
    elif(len(feeLst) == 1):
        if ('sessionId' in feeLst[0]):
            print('sessionId: ' + feeLst[0]['sessionId'])
    else:
        print('feeLst error: no element.');

def main():
    onFeeEvent(jsonBody) #话单处理
```

```
if __name__ == '__main__':
    main()
```

3.4 C#代码样例

3.4.1 AXB 模式

样例	**AXB模式绑定接口** **AXB模式解绑接口** **AXB模式绑定信息修改接口** **AXB模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于.NET Core SDK 2.1.401版本，要求.NET Core 2.0及以上版本或.NET Framework 4.7.1及以上版本。
引用库	Newtonsoft.Json 11.0.2，请参考**https://www.newtonsoft.com/json**获取。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXB 模式绑定接口

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axb_csharp_demo
{
    class axbBind
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
```

```
        string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/
partners/v1.0"; //APP接入地址+接口访问URI
        string appKey = "a1********"; //APP_Key
        string appSecret = "cfc8********"; //APP_Secret
        string relationNum = "+86180****0001"; //X号码(隐私号码)
        string callerNum = "+86186****5678"; //A号码
        string calleeNum = "+86186****5679"; //B号码

        /*
         * 选填,各参数要求请参考"AXB模式绑定接口"
         */
        //string areaCode = "0755"; //需要绑定的X号码对应的城市码
        //int callDirection = 0; //允许呼叫的方向
        //int duration = 86400; //绑定关系保持时间,到期后会被系统自动解除绑定关系
        //string recordFlag = "false"; //是否需要针对该绑定关系产生的所有通话录音
        //string recordHintTone = "recordHintTone.wav"; //设置录音提示音
        //int maxDuration = 60; //设置允许单次通话进行的最长时间,通话时间从接通被叫的时刻开始计算
        //string privateSms = "true"; //设置该绑定关系是否支持短信功能
        //string callerHintTone = "callerHintTone.wav"; //设置A拨打X号码时的通话前等待音
        //string calleeHintTone = "calleeHintTone.wav"; //设置B拨打X号码时的通话前等待音
        //var preVoice = new Dictionary<string, object>(){
        //    { "callerHintTone", callerHintTone},
        //    { "calleeHintTone", calleeHintTone}
        //};

        FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求Body
            var body = new Dictionary<string, object>() {
                {"relationNum", relationNum},
                //{"areaCode", areaCode},
                {"callerNum", callerNum},
                {"calleeNum", calleeNum},
                //{"callDirection", callDirection},
                //{"duration", duration},
                //{"recordFlag", recordFlag},
                //{"recordHintTone", recordHintTone},
                //{"maxDuration", maxDuration},
                //{"privateSms", privateSms},
                //{"preVoice", preVoice}
            };

            HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
            //请求Headers中的Content-Type参数
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            byte[] bbody = Encoding.Default.GetBytes("绑定请求数据：" + JsonConvert.SerializeObject(body)
+ "\r\n");
            fs.Write(bbody, 0, bbody.Length); //绑定请求参数记录到本地文件,方便定位问题

            var response = client.PostAsync(apiAddress, content).Result; //POST请求
            Console.WriteLine(response.StatusCode);
```

```
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应数据
            byte[] bres = Encoding.Default.GetBytes("绑定结果：" + res + "\r\n");
            fs.Write(bres, 0, bres.Length); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
        finally
        {
            fs.Close(); //关闭文件
        }
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;


        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

AXB 模式解绑接口

```
using System;
using System.Collections.Specialized;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axb_csharp_demo
{
  class axbUnbind
  {
    static void Main(string[] args)
    {
        //必填,请参考"开发准备"获取如下数据,替换为实际值
        string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/
partners/v1.0"; //APP接入地址+接口访问URI
        string appKey = "a1********"; //APP_Key
        string appSecret = "cfc8********"; //APP_Secret

        /*
         * 选填,各参数要求请参考"AXB模式解绑接口"
         * subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
         */
```

```
        string subscriptionId = "****"; //指定"AXB模式绑定接口"返回的绑定ID进行解绑
        string relationNum = "+86180****0001"; //指定X号码(隐私号码)进行解绑

        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求url参数
            var keyValues = new NameValueCollection();
            keyValues.Add("subscriptionId", subscriptionId);
            keyValues.Add("relationNum", relationNum);
            //完整请求地址
            var url = apiAddress + "?" + buildQueryString(keyValues);

            Console.WriteLine(buildQueryString(keyValues)); //打印请求数据
            var response = client.DeleteAsync(url).Result; //DELETE请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string buildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
        return temp.Remove(temp.Length-1,1).ToString();
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
```

```
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

AXB 模式绑定信息修改接口

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axb_csharp_demo
{
  class axbModify
  {
    static void Main(string[] args)
    {
      //必填,请参考"开发准备"获取如下数据,替换为实际值
      string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/
partners/v1.0"; //APP接入地址+接口访问URI
      string appKey = "a1********"; //APP_Key
      string appSecret = "cfc8********"; //APP_Secret

      string subscriptionId = "0167ecc9-bfb6-4eec-b671-a7dab2ba78c"; //必填,指定"AXB模式绑定接口"返回
的绑定ID进行修改

      /*
       * 选填,各参数要求请参考"AXB模式绑定信息修改接口"
       */
      string callerNum = "+86186****5678"; //A号码
      string calleeNum = "+86186****5679"; //B号码
      //int callDirection = 0; //允许呼叫的方向
      //int duration = 86400; //绑定关系保持时间,到期后会被系统自动解除绑定关系
      //int maxDuration = 60; //设置允许单次通话进行的最长时间,通话时间从接通被叫的时刻开始计算
      //string privateSms = "true"; //设置该绑定关系是否支持短信功能

      //string callerHintTone = "callerHintTone.wav"; //设置A拨打X号码时的通话前等待音
      //string calleeHintTone = "calleeHintTone.wav"; //设置B拨打X号码时的通话前等待音
      //var preVoice = new Dictionary<string, object>(){
      //    { "callerHintTone", callerHintTone},
      //    { "calleeHintTone", calleeHintTone}
      //};

      try
      {
        //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
        //.NET Framework 4.7.1及以上版本,请采用如下代码
        var sslHandler = new HttpClientHandler()
        {
          ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
        };
        HttpClient client = new HttpClient(sslHandler, true);
        //低于.NET Framework 4.7.1版本,请采用如下代码
        //HttpClient client = new HttpClient();
        //ServicePointManager.ServerCertificateValidationCallback = new
```

```
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求Body
            var body = new Dictionary<string, object>() {
                {"subscriptionId", subscriptionId},
                {"callerNum", callerNum},
                {"calleeNum", calleeNum},
                //{"callDirection", callDirection},
                //{"duration", duration},
                //{"maxDuration", maxDuration},
                //{"privateSms", privateSms},
                //{"preVoice", preVoice}
            };

            HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
            //请求Headers中的Content-Type参数
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            Console.WriteLine(JsonConvert.SerializeObject(body)); //打印请求数据
            var response = client.PutAsync(apiAddress, content).Result; //PUT请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

AXB 模式绑定信息查询接口

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
```

```
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axb_csharp_demo
{
    class axbQueryBind
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/
partners/v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            /*
             * 选填,各参数要求请参考"AXB模式绑定信息查询接口"
             * subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
             */
            string subscriptionId = "****"; //指定"AXB模式绑定接口"返回的绑定ID进行查询
            string relationNum = "+86180****0001"; //指定X号码(隐私号码)进行查询
            //int pageIndex = 1; //查询的分页索引,从1开始编号
            //int pageSize = 20; //查询的分页大小,即每次查询返回多少条数据

            FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
            try
            {
                //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
                //.NET Framework 4.7.1及以上版本,请采用如下代码
                var sslHandler = new HttpClientHandler()
                {
                    ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
                };
                HttpClient client = new HttpClient(sslHandler, true);
                //低于.NET Framework 4.7.1版本,请采用如下代码
                //HttpClient client = new HttpClient();
                //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

                //请求Headers
                client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
                client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
                //请求url参数
                var keyValues = new NameValueCollection();
                keyValues.Add("subscriptionId", subscriptionId);
                keyValues.Add("relationNum", relationNum);
                //keyValues.Add("pageIndex", Convert.ToString(pageIndex));
                //keyValues.Add("pageSize", Convert.ToString(pageSize));
                //完整请求地址
                var url = apiAddress + "?" + buildQueryString(keyValues);

                var response = client.GetAsync(url).Result; //GET请求
                Console.WriteLine(response.StatusCode);
                var res = response.Content.ReadAsStringAsync().Result;
                Console.WriteLine(res); //打印响应结果
                byte[] bres = Encoding.Default.GetBytes("绑定查询结果：" + res + "\r\n");
                fs.Write(bres, 0, bres.Length); //查询结果,记录到本地文件
            }
            catch (Exception e)
            {
                Console.WriteLine(e.StackTrace);
                Console.WriteLine(e.Message); //打印错误信息
            }
            finally
            {
                fs.Close(); //关闭文件
            }
        }
```

```
        /// <summary>
        /// 构造请求url参数
        /// </summary>
        /// <param name="keyValues"></param>
        /// <returns></returns>
        static string buildQueryString(NameValueCollection keyValues)
        {
            StringBuilder temp = new StringBuilder();
            foreach (string item in keyValues.Keys)
            {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
            }
            return temp.Remove(temp.Length - 1, 1).ToString();
        }

        static string buildAKSKHeader(string appKey, string appSecret)
        {
            string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
            string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
            String str = nonce + now;

            byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
            byte[] messageBytes = Encoding.UTF8.GetBytes(str);
            using (var hmacsha256 = new HMACSHA256(keyByte))
            {
                byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                string base64 = Convert.ToBase64String(hashmessage);
                return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

获取录音文件下载地址接口

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axb_csharp_demo
{
    class axbGetRecordLink
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/
v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            //必填,通过"话单通知接口"获取
            string fileName = "****.wav"; //录音文件名
            string recordDomain = "****.com"; //录音文件存储的服务器域名
```

```
        FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            sslHandler.AllowAutoRedirect = false; //关闭重定向
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //var sslHandler = new HttpClientHandler();
            //sslHandler.AllowAutoRedirect = false;
            //HttpClient client = new HttpClient(sslHandler);
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求url参数
            var keyValues = new NameValueCollection();
            keyValues.Add("fileName", fileName);
            keyValues.Add("recordDomain", recordDomain);
            //完整请求地址
            var url = apiAddress + "?" + buildQueryString(keyValues);

            var response = client.GetAsync(apiAddress).Result; //GET请求
            if (response.StatusCode.Equals(301))
            {
                var slink = Convert.ToString(response.Headers.GetValues("Location"));
                Console.WriteLine(slink);
                byte[] blink = Encoding.Default.GetBytes("获取录音文件下载地址：" + slink + "\r\n");
                fs.Write(blink, 0, blink.Length); //查询结果,记录到本地文件
            }
            else
            {
                var res = response.Content.ReadAsStringAsync().Result;
                Console.WriteLine(response.StatusCode);
                Console.WriteLine(res); //打印响应结果
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
        finally
        {
            fs.Close(); //关闭文件
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string buildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
```

```
            return temp.Remove(temp.Length - 1, 1).ToString();
        }

        static string buildAKSKHeader(string appKey, string appSecret)
        {
            string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
            string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
            String str = nonce + now;

            byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
            byte[] messageBytes = Encoding.UTF8.GetBytes(str);
            using (var hmacsha256 = new HMACSHA256(keyByte))
            {
                byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                string base64 = Convert.ToBase64String(hashmessage);
                return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

呼叫事件通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;

namespace privatenumber_axb_csharp_demo
{
    class callEventImpl
    {
        static void Main(string[] args)
        {
            //呼叫事件通知样例
            string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
                {"eventType", "disconnect"},
                {"statusInfo", new Dictionary<string, object>(){
                        {"sessionId", "1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com"},
                        {"timestamp", "2019-01-23 09:16:41"},
                        {"caller", "+86138****0021"},
                        {"called", "+86138****7021"},
                        {"stateCode", 0},
                        {"stateDesc", "The user releases the call."},
                        {"subscriptionId", "****"}
                    }
                }
            });

            Console.WriteLine("jsonBody:" + jsonBody);

            //呼叫事件处理
            OnCallEvent(jsonBody);
        }

        /// <summary>
        /// 呼叫事件通知,详细内容以接口文档为准
        /// </summary>
        /// <param name="jsonBody"></param>
        static void OnCallEvent(string jsonBody)
```



```
{
    JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj
    string eventType = jsonObj["eventType"].ToString(); //通知事件类型

    if ("fee".Equals(eventType))
    {
        Console.WriteLine("EventType error:" + eventType);
        return;
    }

    if (!jsonObj.ContainsKey("statusInfo"))
    {
        Console.WriteLine("param error: no statusInfo.");
        return;
    }
    JObject statusInfo = (JObject)jsonObj["statusInfo"]; //呼叫状态事件信息

    Console.WriteLine("eventType:" + eventType); //打印通知事件类型
    //callin：呼入事件
    if ("callin".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.ContainsKey("sessionId"))
        {
            Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
        }
        return;
    }
    //callout：呼出事件
    if ("callout".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.ContainsKey("sessionId"))
        {
            Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
        }
        return;
    }
    //alerting：振铃事件
    if ("alerting".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.ContainsKey("sessionId"))
        {
            Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
```

```
            }
            return;
        }
        //answer：应答事件
        if ("answer".Equals(eventType))
        {
            /**
             * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'subscriptionId': 绑定关系ID
             */
            if (statusInfo.ContainsKey("sessionId"))
            {
                Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
            }
            return;
        }
        //disconnect：挂机事件
        if ("disconnect".Equals(eventType))
        {
            /**
             * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
             *
             * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
             * 'sessionId': 通话链路的标识ID
             * 'caller': 主叫号码
             * 'called': 被叫号码
             * 'stateCode': 通话挂机的原因值
             * 'stateDesc': 通话挂机的原因值的描述
             * 'subscriptionId': 绑定关系ID
             */
            if (statusInfo.ContainsKey("sessionId"))
            {
                Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
            }
            return;
        }
    }
  }
}
```

话单通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;

namespace privatenumber_axb_csharp_demo
{
  class feeImpl
  {
    static void Main(string[] args)
    {
        //话单通知样例
        string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
            {"eventType", "fee"},
            {"feeLst",  new object[] {
                new Dictionary<string, object>(){
                        {"direction", 1},
                        {"spId", "****"},
                        {"appKey", "********"},
                        {"icid", "ba171f34e6953fcd751edc77127748f4.3757223714.337238282.9"},
                        {"bindNum", "+86138****0022"},
                        {"sessionId",
```

```
"1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com"},
                        {"subscriptionId", "****"},
                        {"callerNum", "+86138****0021"},
                        {"calleeNum", "+86138****0022"},
                        {"fwdDisplayNum", "+86138****0022"},
                        {"fwdDstNum", "+86138****7021"},
                        {"callInTime", "2019-01-23 09:15:14"},
                        {"fwdStartTime", "2019-01-23 09:15:15"},
                        {"fwdAlertingTime", "2019-01-23 09:15:21"},
                        {"fwdAnswerTime", "2019-01-23 09:15:36"},
                        {"callEndTime", "2019-01-23 09:16:41"},
                        {"fwdUnaswRsn", 0},
                        {"ulFailReason", 0},
                        {"sipStatusCode", 0},
                        {"callOutUnaswRsn", 0},
                        {"recordFlag", 1},
                        {"recordStartTime", "2019-01-23 09:15:37"},
                        {"recordDomain", "****.com"},
                        {"recordBucketName", "sp-********"},
                        {"recordObjectName", "********.wav"},
                        {"ttsPlayTimes", 0},
                        {"ttsTransDuration", 0},
                        {"mptyId", "****"},
                        {"serviceType", "004"},
                        {"hostName", "callenabler246.huaweicaas.com"}
                    }
                }
            }
        });

        Console.WriteLine("jsonBody:" + jsonBody);

        //话单处理
        OnFeeEvent(jsonBody);
    }

    /// <summary>
    /// 话单通知,详细内容以接口文档为准
    /// </summary>
    /// <param name="jsonBody"></param>
    static void OnFeeEvent(string jsonBody)
    {
        JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj
        string eventType = jsonObj["eventType"].ToString(); //通知事件类型

        if (!"fee".Equals(eventType))
        {
            Console.WriteLine("EventType error:" + eventType);
            return;
        }

        if (!jsonObj.ContainsKey("feeLst"))
        {
            Console.WriteLine("param error: no feeLst.");
            return;
        }
        JObject[] feeLst = jsonObj["feeLst"].ToObject<JObject[]>(); //呼叫话单事件信息
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'direction': 通话的呼叫方向
         * 'spId': 客户的云服务账号
         * 'appKey': 商户应用的AppKey
         * 'icid': 呼叫记录的唯一标识
         * 'bindNum': 隐私保护号码
         * 'sessionId': 通话链路的唯一标识
         * 'callerNum': 主叫号码
         * 'calleeNum': 被叫号码
         * 'fwdDisplayNum': 转接呼叫时的显示号码
```

```
            * 'fwdDstNum': 转接呼叫时的转接号码
            * 'callInTime': 呼入的开始时间
            * 'fwdStartTime': 转接呼叫操作的开始时间
            * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
            * 'fwdAnswerTime': 转接呼叫操作后的应答时间
            * 'callEndTime': 呼叫结束时间
            * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
            * 'failTime': 呼入,呼出的失败时间
            * 'ulFailReason': 通话失败的拆线点
            * 'sipStatusCode': 呼入,呼出的失败SIP状态码
            * 'recordFlag': 录音标识
            * 'recordStartTime': 录音开始时间
            * 'recordObjectName': 录音文件名
            * 'recordBucketName': 录音文件所在的目录名
            * 'recordDomain': 存放录音文件的域名
            * 'serviceType': 携带呼叫的业务类型信息
            * 'hostName': 话单生成的服务器设备对应的主机名
            * 'subscriptionId': 绑定关系ID
            */
            //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
            if (feeLst.Length > 1)
            {
                foreach (JObject loop in feeLst)
                {
                    if (loop.ContainsKey("sessionId"))
                    {
                        Console.WriteLine("sessionId:" + loop["sessionId"]);
                    }
                }
            }
            else if (feeLst.Length == 1)
            {
                if (feeLst[0].ContainsKey("sessionId"))
                {
                    Console.WriteLine("sessionId:" + feeLst[0]["sessionId"]);
                }
            }
            else
            {
                Console.WriteLine("feeLst error: no element.");
            }
        }
    }
}
```

短信通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections;
using System.Collections.Generic;

namespace privatenumber_axb_csharp_demo
{
    class smsNotifyImpl
    {
        static void Main(string[] args)
        {
            //短信通知样例
            string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
                {"appKey", "****"},
                {"smsEvent", new Dictionary<string, object>(){
                        {"smsIdentifier", "****"},
                        {"notificationMode", "Block"},
                        {"calling", "+86138****0001"},
                        {"virtualNumber", "+86138****0000"},
                        {"event", "TextSMS"},
                        {"timeStamp", "2018-09-13T09:46:16.023Z"}
```

```
                }
            }
        });

        Console.WriteLine("jsonBody:" + jsonBody);

        //短信通知处理
        OnSmsEvent(jsonBody);
    }

    /// <summary>
    /// 短信通知,详细内容以接口文档为准
    /// </summary>
    /// <param name="jsonBody"></param>
    static void OnSmsEvent(string jsonBody)
    {
        JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj

        if (!jsonObj.ContainsKey("smsEvent"))
        {
            Console.WriteLine("param error: no smsEvent.");
            return;
        }

        //Console.WriteLine("appKey:" + jsonObj["appKey"]); //商户应用的AppKey

        JObject smsEvent = (JObject)jsonObj["smsEvent"]; //短信通知信息

        /**
         * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
         *
         * 'smsIdentifier': 短信唯一标识
         * 'notificationMode': 通知模式
         * 'calling': 真实发送方号码
         * 'called': 真实接收方号码
         * 'virtualNumber': 隐私号码(X号码)
         * 'event': 短信状态事件
         * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
         * 'subscriptionId': 绑定ID
         * 'smsContent': 用户发送的短信内容
         */
        if (smsEvent.ContainsKey("notificationMode"))
        {
            if ("Block".Equals(smsEvent["notificationMode"].ToString()))
            {
                //收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
                IDictionary actions = new Dictionary<string, object>(){
                    {"operation", "vNumberRoute"}, //操作类型,转发短信/丢弃短信
                    {"message", new Dictionary<string, object>(){
                            {"called", "+86138****7022"}, //真实接收方号码
                            {"calling", "+86138****7021"} //真实发送方号码
                        }
                    }
                };
                string resp = JsonConvert.SerializeObject(new Dictionary<string, object>() {{"actions", new
object[] { actions }}}); //Block模式响应消息
                Console.WriteLine("resp:" + resp);
            }
            else if ("Notify".Equals(smsEvent["notificationMode"].ToString()))
            {
                //收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
                //string statusCode = 200;
                Console.WriteLine("This is AXB sms Notify mode.");
            }
            else
            {
                Console.WriteLine("notificationMode param error.");
            }
        }
```

```
    }
  }
}
```

3.4.2 AX 模式

样例	**AX模式绑定接口** **AX模式解绑接口** **AX模式绑定信息修改接口** **AX模式绑定信息查询接口** **AX模式设置临时被叫接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于.NET Core SDK 2.1.401版本，要求.NET Core 2.0及以上版本或.NET Framework 4.7.1及以上版本。
引用库	Newtonsoft.Json 11.0.2，请参考**https://www.newtonsoft.com/json**获取。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AX 模式绑定接口

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_ax_csharp_demo
{
    class axBind
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/
privatenumber/v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
```

```
        string appSecret = "cfc8********"; //APP_Secret
        string origNum = "+86186****5678"; //A号码
        string privateNum = "+86180****0001"; //X号码(隐私号码)

        /*
         * 选填,各参数要求请参考"AX模式绑定接口"
         */
        string privateNumType = "mobile-virtual"; //固定为mobile-virtual
        //string areaCode = "0755"; //需要绑定的X号码对应的城市码
        //string recordFlag = "true"; //是否需要针对该绑定关系产生的所有通话录音
        //string recordHintTone = "recordHintTone.wav"; //设置录音提示音
        string calleeNumDisplay = "0"; //设置非A用户呼叫X时,A接到呼叫时的主显号码
        //string privateSms = "true"; //设置该绑定关系是否支持短信功能

        //string callerHintTone = "callerHintTone.wav"; //设置A拨打X号码时的通话前等待音
        //string calleeHintTone = "calleeHintTone.wav"; //设置非A用户拨打X号码时的通话前等待音
        //var preVoice = new Dictionary<string, object>(){
        //    { "callerHintTone", callerHintTone},
        //    { "calleeHintTone", calleeHintTone}
        //};

        FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求Body
            var body = new Dictionary<string, object>() {
                {"origNum", origNum},
                {"privateNum", privateNum},
                {"privateNumType", privateNumType},
                //{"areaCode", areaCode},
                //{"recordFlag", recordFlag},
                //{"recordHintTone", recordHintTone},
                {"calleeNumDisplay", calleeNumDisplay},
                //{"privateSms", privateSms},
                //{"preVoice", preVoice}
            };

            HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
            //请求Headers中的Content-Type参数
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            byte[] bbody = Encoding.Default.GetBytes("绑定请求数据：" + JsonConvert.SerializeObject(body)
+ "\r\n");
            fs.Write(bbody, 0, bbody.Length); //绑定请求参数记录到本地文件,方便定位问题

            var response = client.PostAsync(apiAddress, content).Result; //POST请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应数据
            byte[] bres = Encoding.Default.GetBytes("绑定结果：" + res + "\r\n");
            fs.Write(bres, 0, bres.Length); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
        }
        catch (Exception e)
```

```
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
        finally
        {
            fs.Close(); //关闭文件
        }
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

AX 模式解绑接口

```
using System;
using System.Collections.Specialized;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_ax_csharp_demo
{
  class axUnbind
  {
    static void Main(string[] args)
    {
      //必填,请参考"开发准备"获取如下数据,替换为实际值
      string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/
privatenumber/v1.0"; //APP接入地址+接口访问URI
      string appKey = "a1d1f50cad21415fbdd13d8f53d36d60"; //APP_Key
      string appSecret = "cfc8********"; //APP_Secret

      /*
       * 选填,各参数要求请参考"AX模式解绑接口"
       * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
       */
      //string origNum = "+86186****5678"; //A号码
      //string privateNum = "+86180****0001"; //X号码(隐私号码)
      string subscriptionId = "****"; //指定"AX模式绑定接口"返回的绑定ID进行解绑

      try
      {
        //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
```

```
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求url参数
            var keyValues = new NameValueCollection();
            keyValues.Add("subscriptionId", subscriptionId);
            //keyValues.Add("origNum", origNum);
            //keyValues.Add("privateNum", privateNum);
            //完整请求地址
            var url = apiAddress + "?" + buildQueryString(keyValues);

            Console.WriteLine(buildQueryString(keyValues)); //打印请求数据
            var response = client.DeleteAsync(url).Result; //DELETE请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string buildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
        return temp.Remove(temp.Length - 1, 1).ToString();
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }
```

```
        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

AX 模式绑定信息修改接口

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_ax_csharp_demo
{
    class axModify
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/
privatenumber/v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            /*
             * 选填,各参数要求请参考"AX模式绑定信息修改接口"
             * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
             */
            string origNum = "+86186****5678"; //A号码
            string privateNum = "+86180****0001"; //X号码(隐私号码)
            //string subscriptionId = "****"; //指定"AX模式绑定接口"返回的绑定ID进行解绑

            string privateSms = "true"; //必填,修改该绑定关系是否支持短信功能

            try
            {
                //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
                //.NET Framework 4.7.1及以上版本,请采用如下代码
                var sslHandler = new HttpClientHandler()
                {
                    ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
                };
                HttpClient client = new HttpClient(sslHandler, true);
                //低于.NET Framework 4.7.1版本,请采用如下代码
                //HttpClient client = new HttpClient();
                //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

                //请求Headers
                client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
                client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
                //请求Body
                var body = new Dictionary<string, object>() {
                    {"origNum", origNum},
                    {"privateNum", privateNum},
                    //{"subscriptionId", subscriptionId},
                    {"privateSms", privateSms}
                };

                HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
```

```
            //请求Headers中的Content-Type参数
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            Console.WriteLine(JsonConvert.SerializeObject(body)); //打印请求数据
            var response = client.PutAsync(apiAddress, content).Result; //PUT请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

AX 模式绑定信息查询接口

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_ax_csharp_demo
{
  class axQueryBind
  {
    static void Main(string[] args)
    {
      //必填,请参考"开发准备"获取如下数据,替换为实际值
      string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/
privatenumber/v1.0"; //APP接入地址+接口访问URI
      string appKey = "a1********"; //APP_Key
      string appSecret = "cfc8********"; //APP_Secret

      /*
       * 选填,各参数要求请参考"AX模式绑定信息查询接口"
       * subscriptionId和origNum为二选一关系,两者都携带时以subscriptionId为准
       */
```

```
        string origNum = "+86186****5678"; //指定A号码进行查询
        string subscriptionId = "****"; //指定"AX模式绑定接口"返回的绑定ID进行查询

        FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求url参数
            var keyValues = new NameValueCollection();
            keyValues.Add("subscriptionId", subscriptionId);
            keyValues.Add("origNum", origNum);
            //完整请求地址
            var url = apiAddress + "?" + buildQueryString(keyValues);

            var response = client.GetAsync(url).Result; //GET请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
            byte[] bres = Encoding.Default.GetBytes("绑定查询结果： " + res + "\r\n");
            fs.Write(bres, 0, bres.Length); //查询结果,记录到本地文件
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
        finally
        {
            fs.Close(); //关闭文件
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string buildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
        return temp.Remove(temp.Length - 1, 1).ToString();
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;
```

```
            byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
            byte[] messageBytes = Encoding.UTF8.GetBytes(str);
            using (var hmacsha256 = new HMACSHA256(keyByte))
            {
                byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                string base64 = Convert.ToBase64String(hashmessage);
                return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

AX 模式设置临时被叫接口

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_ax_csharp_demo
{
    class axSetCalleeNum
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/privatenumber/
calleenumber/v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            /*
             * 选填,各参数要求请参考"AX模式设置临时被叫接口"
             * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
             */
            string origNum = "+86186****5678"; //A号码
            string privateNum = "+86180****0001"; //X号码(隐私号码)
            //string subscriptionId = "****"; //指定“AX模式绑定接口”返回的绑定ID设置临时被叫

            string calleeNum = "+86186****5679"; //必填,本次呼叫的真实被叫号码

            try
            {
                //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
                //.NET Framework 4.7.1及以上版本,请采用如下代码
                var sslHandler = new HttpClientHandler()
                {
                    ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
                };
                HttpClient client = new HttpClient(sslHandler, true);
                //低于.NET Framework 4.7.1版本,请采用如下代码
                //HttpClient client = new HttpClient();
                //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

                //请求Headers
                client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
```

```
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求Body
            var body = new Dictionary<string, object>() {
                {"origNum", origNum},
                {"privateNum", privateNum},
                //{"subscriptionId", subscriptionId},
                {"calleeNum", calleeNum}
            };

            HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
            //请求Headers中的Content-Type参数
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            Console.WriteLine(JsonConvert.SerializeObject(body)); //打印请求数据
            var response = client.PutAsync(apiAddress, content).Result; //PUT请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

获取录音文件下载地址接口

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_ax_csharp_demo
{
  class axGetRecordLink
  {
    static void Main(string[] args)
```

```
    {
        //必填,请参考"开发准备"获取如下数据,替换为实际值
        string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/
v1.0"; //APP接入地址+接口访问URI
        string appKey = "a1********"; //APP_Key
        string appSecret = "cfc8********"; //APP_Secret

        //必填,通过"话单通知接口"获取
        string fileName = "****.wav"; //录音文件名
        string recordDomain = "****.com"; //录音文件存储的服务器域名

        FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            sslHandler.AllowAutoRedirect = false; //关闭重定向
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //var sslHandler = new HttpClientHandler();
            //sslHandler.AllowAutoRedirect = false;
            //HttpClient client = new HttpClient(sslHandler);
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求url参数
            var keyValues = new NameValueCollection();
            keyValues.Add("fileName", fileName);
            keyValues.Add("recordDomain", recordDomain);
            //完整请求地址
            var url = apiAddress + "?" + buildQueryString(keyValues);

            var response = client.GetAsync(url).Result; //GET请求
            if (response.StatusCode.Equals(301))
            {
                var slink = Convert.ToString(response.Headers.GetValues("Location"));
                Console.WriteLine(slink);
                byte[] blink = Encoding.Default.GetBytes("获取录音文件下载地址：" + slink + "\r\n");
                fs.Write(blink, 0, blink.Length); //查询结果,记录到本地文件
            }
            else
            {
                var res = response.Content.ReadAsStringAsync().Result;
                Console.WriteLine(response.StatusCode);
                Console.WriteLine(res); //打印响应结果
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
        finally
        {
            fs.Close(); //关闭文件
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
```

```
        /// <param name="keyValues"></param>
        /// <returns></returns>
        static string buildQueryString(NameValueCollection keyValues)
        {
            StringBuilder temp = new StringBuilder();
            foreach (string item in keyValues.Keys)
            {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
            }
            return temp.Remove(temp.Length - 1, 1).ToString();
        }

        static string buildAKSKHeader(string appKey, string appSecret)
        {
            string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
            string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
            String str = nonce + now;

            byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
            byte[] messageBytes = Encoding.UTF8.GetBytes(str);
            using (var hmacsha256 = new HMACSHA256(keyByte))
            {
                byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                string base64 = Convert.ToBase64String(hashmessage);
                return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

呼叫事件通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;

namespace privatenumber_ax_csharp_demo
{
    class callEventImpl
    {
        static void Main(string[] args)
        {
            //呼叫事件通知样例
            string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
                {"eventType", "disconnect"},
                {"statusInfo", new Dictionary<string, object>(){
                        {"sessionId", "1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com"},
                        {"timestamp", "2019-01-24 02:30:22"},
                        {"caller", "+86138****0022"},
                        {"called", "+86138****7021"},
                        {"stateCode", 0},
                        {"stateDesc", "The user releases the call."},
                        {"subscriptionId", "****"}
                    }
                }
            });

            Console.WriteLine("jsonBody:" + jsonBody);
```



```
    //呼叫事件处理
    OnCallEvent(jsonBody);
}

/// <summary>
/// 呼叫事件通知,详细内容以接口文档为准
/// </summary>
/// <param name="jsonBody"></param>
static void OnCallEvent(string jsonBody)
{
    JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj
    string eventType = jsonObj["eventType"].ToString(); //通知事件类型

    if ("fee".Equals(eventType))
    {
        Console.WriteLine("EventType error:" + eventType);
        return;
    }

    if (!jsonObj.ContainsKey("statusInfo"))
    {
        Console.WriteLine("param error: no statusInfo.");
        return;
    }
    JObject statusInfo = (JObject)jsonObj["statusInfo"]; //呼叫状态事件信息

    Console.WriteLine("eventType:" + eventType); //打印通知事件类型
    //callin：呼入事件
    if ("callin".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.ContainsKey("sessionId"))
        {
            Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
        }
        return;
    }
    //callout：呼出事件
    if ("callout".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.ContainsKey("sessionId"))
        {
            Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
        }
        return;
    }
    //alerting：振铃事件
    if ("alerting".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
```

```
                *
                * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
                * 'sessionId': 通话链路的标识ID
                * 'caller': 主叫号码
                * 'called': 被叫号码
                * 'subscriptionId': 绑定关系ID
                */
                if (statusInfo.ContainsKey("sessionId"))
                {
                    Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
                }
                return;
            }
            //answer：应答事件
            if ("answer".Equals(eventType))
            {
                /**
                * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
                *
                * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
                * 'sessionId': 通话链路的标识ID
                * 'caller': 主叫号码
                * 'called': 被叫号码
                * 'subscriptionId': 绑定关系ID
                */
                if (statusInfo.ContainsKey("sessionId"))
                {
                    Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
                }
                return;
            }
            //disconnect：挂机事件
            if ("disconnect".Equals(eventType))
            {
                /**
                * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
                *
                * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
                * 'sessionId': 通话链路的标识ID
                * 'caller': 主叫号码
                * 'called': 被叫号码
                * 'stateCode': 通话挂机的原因值
                * 'stateDesc': 通话挂机的原因值的描述
                * 'subscriptionId': 绑定关系ID
                */
                if (statusInfo.ContainsKey("sessionId"))
                {
                    Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
                }
                return;
            }
        }
    }
}
```

话单通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;

namespace privatenumber_ax_csharp_demo
{
    class feeImpl
    {
        static void Main(string[] args)
        {
            //话单通知样例
```

```
        string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
            {"eventType", "fee"},
            {"feeLst",  new object[] {
                new Dictionary<string, object>(){
                        {"direction", 1},
                        {"spId", "****"},
                        {"appKey", "****"},
                        {"icid", "ba171f34e6953fcd751edc77127748f4.3757285803.338666679.5"},
                        {"bindNum", "+86138****0022"},
                        {"sessionId",
"1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com"},
                        {"subscriptionId", "****"},
                        {"callerNum", "+86138****0021"},
                        {"calleeNum", "+86138****0022"},
                        {"fwdDisplayNum", "+86138****0022"},
                        {"fwdDstNum", "+86138****7021"},
                        {"callInTime", "2019-01-24 02:30:03"},
                        {"fwdStartTime", "2019-01-24 02:30:03"},
                        {"fwdAlertingTime", "2019-01-24 02:30:04"},
                        {"fwdAnswerTime", "2019-01-24 02:30:06"},
                        {"callEndTime", "2019-01-24 02:30:22"},
                        {"fwdUnaswRsn", 0},
                        {"ulFailReason", 0},
                        {"sipStatusCode", 0},
                        {"callOutUnaswRsn", 0},
                        {"recordFlag", 1},
                        {"recordStartTime", "2019-01-24 02:30:06"},
                        {"recordDomain", "****.com"},
                        {"recordBucketName", "****"},
                        {"recordObjectName", "****.wav"},
                        {"ttsPlayTimes", 0},
                        {"ttsTransDuration", 0},
                        {"mptyId", "****"},
                        {"serviceType", "003"},
                        {"hostName", "callenabler246.huaweicaas.com"}
                    }
                }
            }
        });

        Console.WriteLine("jsonBody:" + jsonBody);

        //话单处理
        OnFeeEvent(jsonBody);
    }

    /// <summary>
    /// 话单通知,详细内容以接口文档为准
    /// </summary>
    /// <param name="jsonBody"></param>
    static void OnFeeEvent(string jsonBody)
    {
        JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj
        string eventType = jsonObj["eventType"].ToString(); //通知事件类型

        if (!"fee".Equals(eventType))
        {
            Console.WriteLine("EventType error:" + eventType);
            return;
        }

        if (!jsonObj.ContainsKey("feeLst"))
        {
            Console.WriteLine("param error: no feeLst.");
            return;
        }
        JObject[] feeLst = jsonObj["feeLst"].ToObject<JObject[]>(); //呼叫话单事件信息
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
```

```
            *
            * 'direction': 通话的呼叫方向
            * 'spId': 客户的云服务账号
            * 'appKey': 商户应用的AppKey
            * 'icid': 呼叫记录的唯一标识
            * 'bindNum': 隐私保护号码
            * 'sessionId': 通话链路的唯一标识
            * 'callerNum': 主叫号码
            * 'calleeNum': 被叫号码
            * 'fwdDisplayNum': 转接呼叫时的显示号码
            * 'fwdDstNum': 转接呼叫时的转接号码
            * 'callInTime': 呼入的开始时间
            * 'fwdStartTime': 转接呼叫操作的开始时间
            * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
            * 'fwdAnswerTime': 转接呼叫操作后的应答时间
            * 'callEndTime': 呼叫结束时间
            * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
            * 'failTime': 呼入,呼出的失败时间
            * 'ulFailReason': 通话失败的拆线点
            * 'sipStatusCode': 呼入,呼出的失败SIP状态码
            * 'recordFlag': 录音标识
            * 'recordStartTime': 录音开始时间
            * 'recordObjectName': 录音文件名
            * 'recordBucketName': 录音文件所在的目录名
            * 'recordDomain': 存放录音文件的域名
            * 'serviceType': 携带呼叫的业务类型信息
            * 'hostName': 话单生成的服务器设备对应的主机名
            * 'subscriptionId': 绑定关系ID
            */
            //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
            if (feeLst.Length > 1)
            {
                foreach (JObject loop in feeLst)
                {
                    if (loop.ContainsKey("sessionId"))
                    {
                        Console.WriteLine("sessionId:" + loop["sessionId"]);
                    }
                }
            }
            else if (feeLst.Length == 1)
            {
                if (feeLst[0].ContainsKey("sessionId"))
                {
                    Console.WriteLine("sessionId:" + feeLst[0]["sessionId"]);
                }
            }
            else
            {
                Console.WriteLine("feeLst error: no element.");
            }
        }
    }
}
```

短信通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections;
using System.Collections.Generic;

namespace privatenumber_ax_csharp_demo
{
    class smsNotifyImpl
    {
        static void Main(string[] args)
        {
```

```
        //短信通知样例
        string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
            {"appKey", "****"},
            {"smsEvent", new Dictionary<string, object>(){
                    {"smsIdentifier", "****"},
                    {"notificationMode", "Block"},
                    {"calling", "+86138****0001"},
                    {"virtualNumber", "+86138****0000"},
                    {"event", "TextSMS"},
                    {"timeStamp", "2018-09-13T09:46:16.023Z"}
                }
            }
        });

        Console.WriteLine("jsonBody:" + jsonBody);

        //短信通知处理
        OnSmsEvent(jsonBody);
    }

    /// <summary>
    /// 短信通知,详细内容以接口文档为准
    /// </summary>
    /// <param name="jsonBody"></param>
    static void OnSmsEvent(string jsonBody)
    {
        JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj

        if (!jsonObj.ContainsKey("smsEvent"))
        {
            Console.WriteLine("param error: no smsEvent.");
            return;
        }

        //Console.WriteLine("appKey:" + jsonObj["appKey"]); //商户应用的AppKey

        JObject smsEvent = (JObject)jsonObj["smsEvent"]; //短信通知信息

        /**
         * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
         *
         * 'smsIdentifier': 短信唯一标识
         * 'notificationMode': 通知模式
         * 'calling': 真实发送方号码
         * 'called': 真实接收方号码
         * 'virtualNumber': 隐私号码(X号码)
         * 'event': 短信状态事件
         * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
         * 'subscriptionId': 绑定ID
         * 'smsContent': 用户发送的短信内容
         */
        if (smsEvent.ContainsKey("notificationMode"))
        {
            if ("Block".Equals(smsEvent["notificationMode"].ToString()))
            {
                //收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
                IDictionary actions = new Dictionary<string, object>(){
                    {"operation", "vNumberRoute"}, //操作类型,转发短信/丢弃短信
                    {"message", new Dictionary<string, object>(){
                            {"called", "+86138****7022"}, //真实接收方号码
                            {"calling", "+86138****7021"} //真实发送方号码
                        }
                    }
                };
                string resp = JsonConvert.SerializeObject(new Dictionary<string, object>(){{ "actions", new
object[]{ actions }}}); //Block模式响应消息
                Console.WriteLine("resp:" + resp);
                }
                else if ("Notify".Equals(smsEvent["notificationMode"].ToString()))
```

```
            {
                //收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
                //string statusCode = 200;
                Console.WriteLine("This is AX sms Notify mode.");
            }
            else
            {
                Console.WriteLine("notificationMode param error.");
            }
        }
    }
  }
}
```

3.4.3 AXE 模式

样例	**AXE模式绑定接口** **AXE模式解绑接口** **AXE模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口**
环境要求	基于.NET Core SDK 2.1.401版本，要求.NET Core 2.0及以上版本或.NET Framework 4.7.1及以上版本。
引用库	Newtonsoft.Json 11.0.2，请参考**https://www.newtonsoft.com/json**获取。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXE 模式绑定接口

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axe_csharp_demo
{
    class axeBind
    {
        static void Main(string[] args)
```

```
    {
        //必填,请参考"开发准备"获取如下数据,替换为实际值
        string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/
v1.0"; //APP接入地址+接口访问URI
        string appKey = "a1********"; //APP_Key
        string appSecret = "cfc8********"; //APP_Secret
        string virtualNum = "+86180****0001"; //AXE中的X号码
        string bindNum = "+86186****5678"; //AXE中的A号码

        /*
         * 选填,各参数要求请参考"AXE模式绑定接口"
         */
        //string areaCode = "0755"; // 需要绑定的X号码对应的城市码
        //string displayNumMode = "0"; // 非A用户呼叫X号码时，A看到的主显号码
        //string recordFlag = "false"; // 是否需要针对该绑定关系产生的所有通话录音
        //string recordHintTone = "recordHintTone.wav"; // 设置录音提示音
        //string callbackTone = "callbackTone.wav"; // A呼叫X不存在回呼记录的提示音
        //string callbackNum = "+86180****0021"; // A呼叫X不存在回呼记录的转接号码
        //int bindExpiredTime = 24; // 绑定关系的有效时间
        //int callbackExpiredTime = 24; // 回呼记录有效时间
        //string userData = "abcdefg"; // 用户自定义数据

        FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求Body
            var body = new Dictionary<string, object>() {
                {"virtualNum", virtualNum},
                {"bindNum", bindNum},
                //{"areaCode", areaCode},
                //{"displayNumMode", displayNumMode},
                //{"recordFlag", recordFlag},
                //{"recordHintTone", recordHintTone},
                //{"callbackTone", callbackTone},
                //{"callbackNum", callbackNum},
                //{"bindExpiredTime", bindExpiredTime},
                //{"callbackExpiredTime", callbackExpiredTime},
                //{"userData", userData}
            };

            HttpContent content = new StringContent(JsonConvert.SerializeObject(body));
            //请求Headers中的Content-Type参数
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

            byte[] bbody = Encoding.Default.GetBytes("绑定请求数据：" + JsonConvert.SerializeObject(body)
+ "\r\n");
            fs.Write(bbody, 0, bbody.Length); //绑定请求参数记录到本地文件,方便定位问题

            var response = client.PostAsync(apiAddress, content).Result; //POST请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应数据
            byte[] bres = Encoding.Default.GetBytes("绑定结果：" + res + "\r\n");
```

```
                fs.Write(bres, 0, bres.Length); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
            }
            catch (Exception e)
            {
                Console.WriteLine(e.StackTrace);
                Console.WriteLine(e.Message); //打印错误信息
            }
            finally
            {
                fs.Close(); //关闭文件
            }
        }

        static string buildAKSKHeader(string appKey, string appSecret)
        {
            string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
            string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
            String str = nonce + now;

            byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
            byte[] messageBytes = Encoding.UTF8.GetBytes(str);
            using (var hmacsha256 = new HMACSHA256(keyByte))
            {
                byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                string base64 = Convert.ToBase64String(hashmessage);
                return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

AXE 模式解绑接口

```
using System;
using System.Collections.Specialized;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axe_csharp_demo
{
    class axeUnbind
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/
v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            /*
             * 选填,各参数要求请参考"AXE模式解绑接口"
             * subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
             */
            string subscriptionId = "****"; //指定"AXE模式绑定接口"返回的绑定ID进行解绑
            string virtualNum = "+86180****0001"; // AXE中的X号码
            string extendNum = "1234"; // AXE中的E号码
```

```
        try
        {
            //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
            //.NET Framework 4.7.1及以上版本,请采用如下代码
            var sslHandler = new HttpClientHandler()
            {
                ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
            };
            HttpClient client = new HttpClient(sslHandler, true);
            //低于.NET Framework 4.7.1版本,请采用如下代码
            //HttpClient client = new HttpClient();
            //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

            //请求Headers
            client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
            client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
            //请求url参数
            var keyValues = new NameValueCollection();
            keyValues.Add("subscriptionId", subscriptionId);
            keyValues.Add("virtualNum", virtualNum);
            keyValues.Add("extendNum", extendNum);
            //完整请求地址
            var url = apiAddress + "?" + buildQueryString(keyValues);

            Console.WriteLine(buildQueryString(keyValues)); //打印请求数据
            var response = client.DeleteAsync(url).Result; //DELETE请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string buildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
        return temp.Remove(temp.Length - 1, 1).ToString();
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
```

```
                appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

AXE 模式绑定信息查询接口

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axe_csharp_demo
{
    class axeQueryBind
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/
v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            /*
             * 选填,各参数要求请参考"AXE模式绑定信息查询接口"
             * subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
             */
            string subscriptionId = "****"; //指定"AXE模式绑定接口"返回的绑定ID进行查询
            string virtualNum = "+86180****0001"; // AXE中的X号码
            string extendNum = "1234"; // AXE中的E号码

            FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
            try
            {
                //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
                //.NET Framework 4.7.1及以上版本,请采用如下代码
                var sslHandler = new HttpClientHandler()
                {
                    ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
                };
                HttpClient client = new HttpClient(sslHandler, true);
                //低于.NET Framework 4.7.1版本,请采用如下代码
                //HttpClient client = new HttpClient();
                //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

                //请求Headers
                client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
                client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
                //请求url参数
                var keyValues = new NameValueCollection();
                keyValues.Add("subscriptionId", subscriptionId);
                keyValues.Add("virtualNum", virtualNum);
                keyValues.Add("extendNum", extendNum);
                //完整请求地址
                var url = apiAddress + "?" + buildQueryString(keyValues);
```

```
            var response = client.GetAsync(url).Result; //GET请求
            Console.WriteLine(response.StatusCode);
            var res = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(res); //打印响应结果
            byte[] bres = Encoding.Default.GetBytes("绑定查询结果：" + res + "\r\n");
            fs.Write(bres, 0, bres.Length); //查询结果,记录到本地文件
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            Console.WriteLine(e.Message); //打印错误信息
        }
        finally
        {
            fs.Close(); //关闭文件
        }
    }

    /// <summary>
    /// 构造请求url参数
    /// </summary>
    /// <param name="keyValues"></param>
    /// <returns></returns>
    static string buildQueryString(NameValueCollection keyValues)
    {
        StringBuilder temp = new StringBuilder();
        foreach (string item in keyValues.Keys)
        {
temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
        }
        return temp.Remove(temp.Length - 1, 1).ToString();
    }

    static string buildAKSKHeader(string appKey, string appSecret)
    {
        string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
        string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
        String str = nonce + now;

        byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
        byte[] messageBytes = Encoding.UTF8.GetBytes(str);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
            byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
            string base64 = Convert.ToBase64String(hashmessage);
            return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                appKey, base64, nonce, now);
        }
    }

    //低于.NET Framework 4.7.1版本,启用如下方法
    //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
    //{
    //    return true;
    //}
  }
}
```

获取录音文件下载地址接口

```
using System;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Http;
```



```
using System.Security.Cryptography;
using System.Text;

namespace privatenumber_axe_csharp_demo
{
    class axeGetRecordLink
    {
        static void Main(string[] args)
        {
            //必填,请参考"开发准备"获取如下数据,替换为实际值
            string apiAddress = "https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/
v1.0"; //APP接入地址+接口访问URI
            string appKey = "a1********"; //APP_Key
            string appSecret = "cfc8********"; //APP_Secret

            //必填,通过"话单通知接口"获取
            string fileName = "****.wav"; //录音文件名
            string recordDomain = "****.com"; //录音文件存储的服务器域名

            FileStream fs = File.Open("bind_data.txt", FileMode.Append); //打开文件
            try
            {
                //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
                //.NET Framework 4.7.1及以上版本,请采用如下代码
                var sslHandler = new HttpClientHandler()
                {
                    ServerCertificateCustomValidationCallback = (message, cert, chain, err) => { return true; }
                };
                sslHandler.AllowAutoRedirect = false; //关闭重定向
                HttpClient client = new HttpClient(sslHandler, true);
                //低于.NET Framework 4.7.1版本,请采用如下代码
                //var sslHandler = new HttpClientHandler();
                //sslHandler.AllowAutoRedirect = false;
                //HttpClient client = new HttpClient(sslHandler);
                //ServicePointManager.ServerCertificateValidationCallback = new
RemoteCertificateValidationCallback(CheckValidationResult);

                //请求Headers
                client.DefaultRequestHeaders.Add("Authorization", "AKSK realm=\"SDP\",profile=
\"UsernameToken\",type=\"Appkey\"");
                client.DefaultRequestHeaders.Add("X-AKSK", buildAKSKHeader(appKey, appSecret));
                //请求url参数
                var keyValues = new NameValueCollection();
                keyValues.Add("fileName", fileName);
                keyValues.Add("recordDomain", recordDomain);
                //完整请求地址
                var url = apiAddress + "?" + buildQueryString(keyValues);

                var response = client.GetAsync(apiAddress).Result; //GET请求
                if (response.StatusCode.Equals(301))
                {
                    var slink = Convert.ToString(response.Headers.GetValues("Location"));
                    Console.WriteLine(slink);
                    byte[] blink = Encoding.Default.GetBytes("获取录音文件下载地址：" + slink + "\r\n");
                    fs.Write(blink, 0, blink.Length); //查询结果,记录到本地文件
                }
                else
                {
                    var res = response.Content.ReadAsStringAsync().Result;
                    Console.WriteLine(response.StatusCode);
                    Console.WriteLine(res); //打印响应结果
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.StackTrace);
                Console.WriteLine(e.Message); //打印错误信息
            }
            finally
```

```
            {
                fs.Close(); //关闭文件
            }
        }

        /// <summary>
        /// 构造请求url参数
        /// </summary>
        /// <param name="keyValues"></param>
        /// <returns></returns>
        static string buildQueryString(NameValueCollection keyValues)
        {
            StringBuilder temp = new StringBuilder();
            foreach (string item in keyValues.Keys)
            {

temp.Append(item).Append("=").Append(WebUtility.UrlEncode(keyValues.Get(item))).Append("&");
            }
            return temp.Remove(temp.Length - 1, 1).ToString();
        }

        static string buildAKSKHeader(string appKey, string appSecret)
        {
            string now = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ"); //Created
            string nonce = Guid.NewGuid().ToString().Replace("-", ""); //Nonce
            String str = nonce + now;

            byte[] keyByte = Encoding.UTF8.GetBytes(appSecret);
            byte[] messageBytes = Encoding.UTF8.GetBytes(str);
            using (var hmacsha256 = new HMACSHA256(keyByte))
            {
                byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                string base64 = Convert.ToBase64String(hashmessage);
                return String.Format("UsernameToken Username=\"{0}\",PasswordDigest=\"{1}\",Nonce=
\"{2}\",Created=\"{3}\"",
                    appKey, base64, nonce, now);
            }
        }

        //低于.NET Framework 4.7.1版本,启用如下方法
        //static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain,
SslPolicyErrors errors)
        //{
        //    return true;
        //}
    }
}
```

呼叫事件通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;

namespace privatenumber_axe_csharp_demo
{
    class callEventImpl
    {
        static void Main(string[] args)
        {
            //呼叫事件通知样例
            string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
                {"eventType", "disconnect"},
                {"statusInfo", new Dictionary<string, object>(){
                        {"sessionId", "1202_1051_4294967295_20190124070250@callenabler246.huaweicaas.com"},
                        {"timestamp", "2019-01-24 07:03:28"},
                        {"caller", "+86138****0022"},
                        {"called", "+86138****7021"},
```

```
                {"stateCode", 0},
                {"stateDesc", "The user releases the call."},
                {"subscriptionId", "********"}
            }
        }
    });

    Console.WriteLine("jsonBody:" + jsonBody);

    //呼叫事件处理
    OnCallEvent(jsonBody);
}

/// <summary>
/// 呼叫事件通知,详细内容以接口文档为准
/// </summary>
/// <param name="jsonBody"></param>
static void OnCallEvent(string jsonBody)
{
    JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj
    string eventType = jsonObj["eventType"].ToString(); //通知事件类型

    if ("fee".Equals(eventType))
    {
        Console.WriteLine("EventType error:" + eventType);
        return;
    }

    if (!jsonObj.ContainsKey("statusInfo"))
    {
        Console.WriteLine("param error: no statusInfo.");
        return;
    }
    JObject statusInfo = (JObject)jsonObj["statusInfo"]; //呼叫状态事件信息

    Console.WriteLine("eventType:" + eventType); //打印通知事件类型
    //callin：呼入事件
    if ("callin".Equals(eventType))
    {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         */
        if (statusInfo.ContainsKey("sessionId"))
        {
            Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
        }
        return;
    }
    //collectInfo：放音收号结果事件,仅AXE模式下的A被叫场景携带
    if ("collectInfo".Equals(eventType))
    {
        /**
         * Example: 此处以解析digitInfo为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'digitInfo': AXE场景中携带收号结果(即用户输入的数字)
         */
        if (statusInfo.ContainsKey("digitInfo"))
        {
            Console.WriteLine("digitInfo:" + statusInfo["digitInfo"]);
        }
        return;
    }
```

```
//callout：呼出事件
if ("callout".Equals(eventType))
{
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (statusInfo.ContainsKey("sessionId"))
    {
        Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
    }
    return;
}
//alerting：振铃事件
if ("alerting".Equals(eventType))
{
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (statusInfo.ContainsKey("sessionId"))
    {
        Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
    }
    return;
}
//answer：应答事件
if ("answer".Equals(eventType))
{
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'subscriptionId': 绑定关系ID
     */
    if (statusInfo.ContainsKey("sessionId"))
    {
        Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
    }
    return;
}
//disconnect：挂机事件
if ("disconnect".Equals(eventType))
{
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
     * 'sessionId': 通话链路的标识ID
     * 'caller': 主叫号码
     * 'called': 被叫号码
     * 'stateCode': 通话挂机的原因值
     * 'stateDesc': 通话挂机的原因值的描述
     * 'subscriptionId': 绑定关系ID
     */
    if (statusInfo.ContainsKey("sessionId"))
```

```
            {
                Console.WriteLine("sessionId:" + statusInfo["sessionId"]);
            }
            return;
        }
    }
  }
}
```

话单通知接口

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;

namespace privatenumber_axe_csharp_demo
{
   class feeImpl
   {
      static void Main(string[] args)
      {
         //话单通知样例
         string jsonBody = JsonConvert.SerializeObject(new Dictionary<string, object>(){
            {"eventType", "fee"},
            {"feeLst",  new object[] {
               new Dictionary<string, object>(){
                     {"direction", 0},
                     {"spId", "****"},
                     {"appKey", "********"},
                     {"icid", "ba171f34e6953fcd751edc77127748f4.3757289590.338833305.5"},
                     {"bindNum", "+86138****0022"},
                     {"sessionId",
"1201_11275_4294967295_20190124033310@callenabler246.huaweicaas.com"},
                     {"subscriptionId", "********"},
                     {"callerNum", "+86138****7021"},
                     {"calleeNum", "+86138****0022"},
                     {"fwdDisplayNum", "+86138****0022"},
                     {"fwdDstNum", "+86138****0021"},
                     {"callInTime", "2019-01-24 03:33:10"},
                     {"fwdStartTime", "2019-01-24 03:33:16"},
                     {"fwdAlertingTime", "2019-01-24 03:33:19"},
                     {"fwdAnswerTime", "2019-01-24 03:33:28"},
                     {"callEndTime", "2019-01-24 03:33:57"},
                     {"fwdUnaswRsn", 0},
                     {"ulFailReason", 0},
                     {"sipStatusCode", 0},
                     {"callOutUnaswRsn", 0},
                     {"recordFlag", 1},
                     {"recordStartTime", "2019-01-24 03:33:28"},
                     {"recordDomain", "****.com"},
                     {"recordBucketName", "****"},
                     {"recordObjectName", "********.wav"},
                     {"ttsPlayTimes", 0},
                     {"ttsTransDuration", 0},
                     {"mptyId", "********"},
                     {"serviceType", "005"},
                     {"hostName", "callenabler246.huaweicaas.com"},
                     {"extendNumber", "02"}
                  }
               }
            }
         });

         Console.WriteLine("jsonBody:" + jsonBody);

         //话单处理
         OnFeeEvent(jsonBody);
      }
```

```
/// <summary>
/// 话单通知,详细内容以接口文档为准
/// </summary>
/// <param name="jsonBody"></param>
static void OnFeeEvent(string jsonBody)
{
    JObject jsonObj = (JObject)JsonConvert.DeserializeObject(jsonBody); //将通知消息解析为jsonObj
    string eventType = jsonObj["eventType"].ToString(); //通知事件类型

    if (!"fee".Equals(eventType))
    {
        Console.WriteLine("EventType error:" + eventType);
        return;
    }

    if (!jsonObj.ContainsKey("feeLst"))
    {
        Console.WriteLine("param error: no feeLst.");
        return;
    }
    JObject[] feeLst = jsonObj["feeLst"].ToObject<JObject[]>(); //呼叫话单事件信息
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
     * 'sipStatusCode': 呼入,呼出的失败SIP状态码
     * 'recordFlag': 录音标识
     * 'recordStartTime': 录音开始时间
     * 'recordObjectName': 录音文件名
     * 'recordBucketName': 录音文件所在的目录名
     * 'recordDomain': 存放录音文件的域名
     * 'serviceType': 携带呼叫的业务类型信息
     * 'hostName': 话单生成的服务器设备对应的主机名
     * 'subscriptionId': 绑定关系ID
     * 'extendNum': 分机号E
     */
    //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
    if (feeLst.Length > 1)
    {
        foreach (JObject loop in feeLst)
        {
            if (loop.ContainsKey("sessionId"))
            {
                Console.WriteLine("sessionId:" + loop["sessionId"]);
            }
        }
    }
    else if (feeLst.Length == 1)
    {
        if (feeLst[0].ContainsKey("sessionId"))
        {
```

```
                Console.WriteLine("sessionId:" + feeLst[0]["sessionId"]);
            }
        }
        else
        {
            Console.WriteLine("feeLst error: no element.");
        }
    }
  }
}
```

3.5 Node.js 代码样例

3.5.1 AXB 模式

样例	**AXB模式绑定接口** **AXB模式解绑接口** **AXB模式绑定信息修改接口** **AXB模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于Node.js 8.12.0版本，要求Node.js 7.0.0及以上版本。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXB 模式绑定接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var fs = require('fs'); //引入fs模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret
var relationNum = '+86180****0001'; //X号码(隐私号码)
var callerNum = '+86186****5678'; //A号码
var calleeNum = '+86186****5679'; //B号码
```

```
/*
 * 选填,各参数要求请参考"AXB模式绑定接口"
 */
//var areaCode = '0755'; //需要绑定的X号码对应的城市码
//var callDirection = 0; //允许呼叫的方向
//var duration = 86400; //绑定关系保持时间,到期后会被系统自动解除绑定关系
//var recordFlag = 'false'; //是否需要针对该绑定关系产生的所有通话录音
//var recordHintTone = 'recordHintTone.wav'; //设置录音提示音
//var maxDuration = 60; //设置允许单次通话进行的最长时间,通话时间从接通被叫的时刻开始计算
//var privateSms = 'true'; //设置该绑定关系是否支持短信功能
//
//var callerHintTone = 'callerHintTone.wav'; //设置A拨打X号码时的通话前等待音
//var calleeHintTone = 'calleeHintTone.wav'; //设置B拨打X号码时的通话前等待音
//var preVoice = {callerHintTone, calleeHintTone};

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname, //URI
    method: 'POST', //请求方法为POST
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var body = JSON.stringify({
    'relationNum':relationNum,
//    'areaCode':areaCode,
    'callerNum':callerNum,
    'calleeNum':calleeNum,
//    'callDirection':callDirection,
//    'duration':duration,
//    'recordFlag':recordFlag,
//    'recordHintTone':recordHintTone,
//    'maxDuration':maxDuration,
//    'privateSms':privateSms,
//    'preVoice':preVoice
});

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
```

```
        fs.appendFileSync('bind_data.txt', '绑定结果：' + d + '\n'); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.write(body); //发送请求Body数据
fs.appendFileSync('bind_data.txt', '绑定请求数据：' + body + '\n'); //绑定请求参数记录到本地文件,方便定位问题
req.end(); //结束请求
fs.closeSync(0); //结束文件操作
```

AXB 模式解绑接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var querystring = require('querystring'); //引入querystring模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

/*
 * 选填,各参数要求请参考"AXB模式解绑接口"
 * subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
 */
var subscriptionId = '****'; //指定"AXB模式绑定接口"返回的绑定ID进行解绑
var relationNum = '+86180****0001'; //指定X号码(隐私号码)进行解绑

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce + time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"', appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'subscriptionId': subscriptionId,
    'relationNum': relationNum
});

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname + "?" + ops, //URI+请求url参数
    method: 'DELETE', //请求方法为DELETE
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
```

```
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.end(); //结束请求
```

AXB 模式绑定信息修改接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

var subscriptionId = '0167ecc9-bfb6-4eec-b671-a7dab2ba78c'; //必填,指定"AXB模式绑定接口"返回的绑定ID进行修改

/*
 * 选填,各参数要求请参考"AXB模式绑定信息修改接口"
 */
var callerNum = '+86186****5678'; //A号码
var calleeNum = '+86186****5679'; //B号码
//var callDirection = 0; //允许呼叫的方向
//var duration = 86400; //绑定关系保持时间,到期后会被系统自动解除绑定关系
//var maxDuration = 60; //设置允许单次通话进行的最长时间,通话时间从接通被叫的时刻开始计算
//var privateSms = 'true'; //设置该绑定关系是否支持短信功能
//
//var callerHintTone = 'callerHintTone.wav'; //设置A拨打X号码时的通话前等待音
//var calleeHintTone = 'calleeHintTone.wav'; //设置B拨打X号码时的通话前等待音
//var preVoice = {callerHintTone, calleeHintTone};

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname, //URI
    method: 'PUT', //请求方法为PUT
```

```
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var body = JSON.stringify({
    'subscriptionId':subscriptionId,
    'callerNum':callerNum,
    'calleeNum':calleeNum,
//    'callDirection':callDirection,
//    'duration':duration,
//    'maxDuration':maxDuration,
//    'privateSms':privateSms,
//    'preVoice':preVoice
});

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.write(body); //发送请求Body数据
req.end(); //结束请求
```

AXB 模式绑定信息查询接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var fs = require('fs'); //引入fs模块
var querystring = require('querystring'); //引入querystring模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/relationnumber/partners/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

/*
 * 选填,各参数要求请参考"AXB模式绑定信息查询接口"
 * subscriptionId和relationNum为二选一关系,两者都携带时以subscriptionId为准
 */
var subscriptionId = '****'; //指定"AXB模式绑定接口"返回的绑定ID进行查询
var relationNum = '+86180****0001'; //指定X号码(隐私号码)进行查询
//var pageIndex = 1; //查询的分页索引,从1开始编号
//var pageSize = 20; //查询的分页大小,即每次查询返回多少条数据

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
```

```
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'subscriptionId': subscriptionId,
    'relationNum': relationNum,
//    'pageIndex': pageIndex,
//    'pageSize': pageSize
});

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname + "?" + ops, //URI+请求url参数
    method: 'GET', //请求方法为GET
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
        fs.appendFileSync('bind_data.txt', '绑定查询结果：' + d + '\n'); //查询结果,记录到本地文件
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.end(); //结束请求
fs.closeSync(0); //结束文件操作
```

获取录音文件下载地址接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var fs = require('fs'); //引入fs模块
var querystring = require('querystring'); //引入querystring模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0'; //APP接入地址
+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

//必填,通过"话单通知接口"获取
var recordDomain = '****.com'; //录音文件存储的服务器域名
var fileName = '****.wav'; //录音文件名

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
```

```
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'recordDomain': recordDomain,
    'fileName': fileName
});

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname + "?" + ops, //URI+请求url参数
    method: 'GET', //请求方法为GET
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    if(301 === res.statusCode){
        var link = Object.getOwnPropertyDescriptor(res.headers, 'location').value;
        fs.appendFileSync('bind_data.txt', '获取录音文件下载地址：' + link + '\n'); //查询结果,记录到本地文件
    }

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.end(); //结束请求
fs.closeSync(0); //结束文件操作
```

呼叫事件通知接口

```
/**
 * 呼叫事件通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
 */

//呼叫事件通知样例
var jsonBody = JSON.stringify({
    'eventType': 'disconnect',
    'statusInfo': {
        'sessionId': '1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com',
        'timestamp': '2019-01-23 09:16:41',
        'caller': '+86138****0021',
        'called': '+86138****7021',
        'stateCode': 0,
        'stateDesc': 'The user releases the call.',
        'subscriptionId': '****'
```



```
    }
});

console.log('jsonBody:', jsonBody);

/**
 * 呼叫事件通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 */
function onCallEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj
    var eventType = jsonObj.eventType; //通知事件类型

    if ('fee' === eventType) {
        console.log('EventType error:', eventType);
        return;
    }

    if (!jsonObj.hasOwnProperty('statusInfo')) {
        console.log('param error: no statusInfo.');
        return;
    }
    var statusInfo = jsonObj.statusInfo; //呼叫状态事件信息

    console.log('eventType:', eventType); //打印通知事件类型
    //callin：呼入事件
    if ('callin' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //callout：呼出事件
    if ('callout' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //alerting：振铃事件
    if ('alerting' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
```

```
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //answer：应答事件
    if ('answer' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //disconnect：挂机事件
    if ('disconnect' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'stateCode': 通话挂机的原因值
         * 'stateDesc': 通话挂机的原因值的描述
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
}

//呼叫事件处理
onCallEvent(jsonBody);
```

话单通知接口

```
/**
 * 话单通知
 * 客户平台收到隐私保护通话平台的话单通知的接口通知
 */

//话单通知样例
var jsonBody = JSON.stringify({
    'eventType': 'fee',
    'feeLst': [
        {
            'direction': 1,
            'spId': '****',
            'appKey': '********',
            'icid': 'ba171f34e6953fcd751edc77127748f4.3757223714.337238282.9',
            'bindNum': '+86138****0022',
            'sessionId': '1200_1029_4294967295_20190123091514@callenabler246.huaweicaas.com',
            'subscriptionId': '****',
            'callerNum': '+86138****0021',
            'calleeNum': '+86138****0022',
            'fwdDisplayNum': '+86138****0022',
            'fwdDstNum': '+86138****7021',
            'callInTime': '2019-01-23 09:15:14',
            'fwdStartTime': '2019-01-23 09:15:15',
```

```
            'fwdAlertingTime': '2019-01-23 09:15:21',
            'fwdAnswerTime': '2019-01-23 09:15:36',
            'callEndTime': '2019-01-23 09:16:41',
            'fwdUnaswRsn': 0,
            'ulFailReason': 0,
            'sipStatusCode': 0,
            'callOutUnaswRsn': 0,
            'recordFlag': 1,
            'recordStartTime': '2019-01-23 09:15:37',
            'recordDomain': '****.com',
            'recordBucketName': 'sp-********',
            'recordObjectName': '********.wav',
            'ttsPlayTimes': 0,
            'ttsTransDuration': 0,
            'mptyId': '****',
            'serviceType': '004',
            'hostName': 'callenabler246.huaweicaas.com'
        }
    ]
});

console.log('jsonBody:', jsonBody);

/**
 * 话单通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 */
function onFeeEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj
    var eventType = jsonObj.eventType; //通知事件类型

    if ('fee' !== eventType) {
        console.log('EventType error:', eventType);
        return;
    }

    if (!jsonObj.hasOwnProperty('feeLst')) {
        console.log('param error: no feeLst.');
        return;
    }
    var feeLst = jsonObj.feeLst; //呼叫话单事件信息
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
     * 'sipStatusCode': 呼入,呼出的失败SIP状态码
     * 'recordFlag': 录音标识
     * 'recordStartTime': 录音开始时间
     * 'recordObjectName': 录音文件名
     * 'recordBucketName': 录音文件所在的目录名
     * 'recordDomain': 存放录音文件的域名
```

```
    * 'serviceType': 携带呼叫的业务类型信息
    * 'hostName': 话单生成的服务器设备对应的主机名
    * 'subscriptionId': 绑定关系ID
    */
   //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
   if (feeLst.length > 1) {
       for ( var i=0; i<feeLst.length; i++) {
           if (feeLst[i].hasOwnProperty('sessionId')) {
               console.log('sessionId:', feeLst[i].sessionId);
           }
       }
   } else if(feeLst.length === 1) {
       if (feeLst[0].hasOwnProperty('sessionId')) {
           console.log('sessionId:', feeLst[0].sessionId);
       }
   } else {
       console.log('feeLst error: no element.');
   }
}

//话单处理
onFeeEvent(jsonBody);
```

短信通知接口

```
/**
 * 短信通知
 * 客户平台收到隐私保护通话平台的短信通知的接口通知
 */

//短信通知样例
var jsonBody = JSON.stringify({
    'appKey': '****',
    'smsEvent': {
        'smsIdentifier': '****',
        'notificationMode': 'Block',
        'calling': '+86138****0001',
        'virtualNumber': '+86138****0000',
        'event': 'TextSMS',
        'timeStamp': '2018-09-13T09:46:16.023Z'
    }
});

console.log('jsonBody:', jsonBody);

/**
 * 短信通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 */
function onSmsEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj

    if (!jsonObj.hasOwnProperty('smsEvent')) {
        console.log('param error: no smsEvent.');
        return;
    }

    //console.log('appKey:', jsonObj.appKey); //商户应用的AppKey

    var smsEvent = jsonObj.smsEvent; //短信通知信息

    /**
     * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
     *
     * 'smsIdentifier': 短信唯一标识
     * 'notificationMode': 通知模式
     * 'calling': 真实发送方号码
     * 'called': 真实接收方号码
```

```
    * 'virtualNumber': 隐私号码(X号码)
    * 'event': 短信状态事件
    * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
    * 'subscriptionId': 绑定ID
    * 'smsContent': 用户发送的短信内容
    */
   if (smsEvent.hasOwnProperty('notificationMode')) {
      if ('Block' === smsEvent.notificationMode) {
         //收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
         var actions = {
            'operation': 'vNumberRoute', //操作类型:转发短信/丢弃短信
            'message': {
               'called': '+86138****7022', //真实接收方号码
               'calling': '+86138****7021' //真实发送方号码
            }
         };
         var resp = JSON.stringify({'actions': [actions]}); //Block模式响应消息
         console.log('resp:', resp);
      } else if ('Notify' === smsEvent.notificationMode) {
         //收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
         //var statusCode = 200;
         console.log('This is AXB sms Notify mode.');
      } else {
         console.log('notificationMode param error.');
      }
   }
}

//短信通知处理
onSmsEvent(jsonBody);
```

3.5.2 AX 模式

样例	**AX模式绑定接口** **AX模式解绑接口** **AX模式绑定信息修改接口** **AX模式绑定信息查询接口** **AX模式设置临时被叫接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口** **短信通知接口**
环境要求	基于Node.js 8.12.0版本，要求Node.js 7.0.0及以上版本。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AX 模式绑定接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var fs = require('fs'); //引入fs模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret
var origNum = '+86186****5678'; //A号码
var privateNum = '+86180****0001'; //X号码(隐私号码)

/*
 * 选填,各参数要求请参考"AX模式绑定接口"
 */
var privateNumType = 'mobile-virtual'; //固定为mobile-virtual
//var areaCode = '0755'; //需要绑定的X号码对应的城市码
//var recordFlag = 'true'; //是否需要针对该绑定关系产生的所有通话录音
//var recordHintTone = 'recordHintTone.wav'; //设置录音提示音
var calleeNumDisplay = '0'; //设置非A用户呼叫X时,A接到呼叫时的主显号码
//var privateSms = 'true'; //设置该绑定关系是否支持短信功能

//var callerHintTone = 'callerHintTone.wav'; //设置A拨打X号码时的通话前等待音
//var calleeHintTone = 'calleeHintTone.wav'; //设置非A用户拨打X号码时的通话前等待音
//var preVoice = {callerHintTone, calleeHintTone};

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname, //URI
    method: 'POST', //请求方法为POST
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var body = JSON.stringify({
    'origNum':origNum,
    'privateNum':privateNum,
    'privateNumType':privateNumType,
//    'areaCode':areaCode,
//    'recordFlag':recordFlag,
//    'recordHintTone':recordHintTone,
    'calleeNumDisplay':calleeNumDisplay,
//    'privateSms':privateSms,
```

```
//    'preVoice':preVoice
});

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
        fs.appendFileSync('bind_data.txt', '绑定结果：' + d + '\n'); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.write(body); //发送请求Body数据
fs.appendFileSync('bind_data.txt', '绑定请求数据：' + body + '\n'); //绑定请求参数记录到本地文件,方便定位问题
req.end(); //结束请求
fs.closeSync(0); //结束文件操作
```

AX 模式解绑接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var querystring = require('querystring'); //引入querystring模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

/*
 * 选填,各参数要求请参考"AX模式解绑接口"
 * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
 */
var origNum = '+86186****5678'; //A号码
var privateNum = '+86180****0001'; //X号码(隐私号码)
var subscriptionId = '****'; //指定"AX模式绑定接口"返回的绑定ID进行解绑

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce + time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"', appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'origNum': origNum,
    'privateNum': privateNum,
    'subscriptionId': subscriptionId
});

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象
```

```
var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname + "?" + ops, //URI+请求url参数
    method: 'DELETE', //请求方法为DELETE
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.end(); //结束请求
```

AX 模式绑定信息修改接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

/*
 * 选填,各参数要求请参考"AX模式绑定信息修改接口"
 * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
 */
var origNum = '+86186****5678'; //A号码
var privateNum = '+86180****0001'; //X号码(隐私号码)
//var subscriptionId = '****'; //指定"AX模式绑定接口"返回的绑定ID进行修改

var privateSms = 'true'; //必填,修改该绑定关系是否支持短信功能

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象
```

```
var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname, //URI
    method: 'PUT', //请求方法为PUT
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var body = JSON.stringify({
    'origNum':origNum,
    'privateNum':privateNum,
//    'subscriptionId':subscriptionId,
    'privateSms':privateSms,
});

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.write(body); //发送请求Body数据
req.end(); //结束请求
```

AX 模式绑定信息查询接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var fs = require('fs'); //引入fs模块
var querystring = require('querystring'); //引入querystring模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

/*
 * 选填,各参数要求请参考"AX模式绑定信息查询接口"
 * subscriptionId和origNum为二选一关系,两者都携带时以subscriptionId为准
 */
var origNum = '+86186****5678'; //指定A号码进行查询
var subscriptionId = '****'; //指定"AX模式绑定接口"返回的绑定ID进行查询

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
```

```
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'origNum': origNum,
    'subscriptionId': subscriptionId
});

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname + "?" + ops, //URI+请求url参数
    method: 'GET', //请求方法为GET
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
        fs.appendFileSync('bind_data.txt', '绑定查询结果：' + d + '\n'); //查询结果,记录到本地文件
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.end(); //结束请求
fs.closeSync(0); //结束文件操作
```

AX 模式设置临时被叫接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/privatenumber/calleenumber/
v1.0'; //APP接入地址+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

/*
 * 选填,各参数要求请参考"AX模式设置临时被叫接口"
 * subscriptionId和(origNum+privateNum)为二选一关系,都携带时以subscriptionId为准
 */
var origNum = '+86186****5678'; //A号码
var privateNum = '+86180****0001'; //X号码(隐私号码)
//var subscriptionId = '****'; //指定"AX模式绑定接口"返回的绑定ID进行修改

var calleeNum = '+86186****5679'; //必填,本次呼叫的真实被叫号码

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
```

```
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname, //URI
    method: 'PUT', //请求方法为PUT
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var body = JSON.stringify({
    'origNum':origNum,
    'privateNum':privateNum,
//    'subscriptionId':subscriptionId,
    'calleeNum':calleeNum,
});

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.write(body); //发送请求Body数据
req.end(); //结束请求
```

获取录音文件下载地址接口

```
/*jshint esversion: 6 */
var https = require('https'); //引入https模块
var url = require('url'); //引入url模块
var fs = require('fs'); //引入fs模块
var querystring = require('querystring'); //引入querystring模块

//必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0'; //APP接入地址
+接口访问URI
var appKey = 'a1********'; //APP_Key
var appSecret = 'cfc8********'; //APP_Secret

//必填,通过"话单通知接口"获取
var recordDomain = '****.com'; //录音文件存储的服务器域名
var fileName = '****.wav'; //录音文件名

/**
 * Build the X-AKSK value.
 * @param appKey
```

```
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'recordDomain': recordDomain,
    'fileName': fileName
});

var urlobj = url.parse(realUrl); //解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, //主机名
    port: urlobj.port, //端口
    path: urlobj.pathname + "?" + ops, //URI+请求url参数
    method: 'GET', //请求方法为GET
    headers: { //请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false //为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); //打印响应码
    console.log('headers:', res.headers); //打印响应Headers

    if(301 === res.statusCode){
        var link = Object.getOwnPropertyDescriptor(res.headers, 'location').value;
        fs.appendFileSync('bind_data.txt', '获取录音文件下载地址：' + link + '\n'); //查询结果,记录到本地文件
    }

    res.setEncoding('utf8'); //设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); //打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); //请求错误时,打印错误信息
});
req.end(); //结束请求
fs.closeSync(0); //结束文件操作
```

呼叫事件通知接口

```
/**
 * 呼叫事件通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
 */

//呼叫事件通知样例
var jsonBody = JSON.stringify({
    'eventType': 'disconnect',
    'statusInfo': {
        'sessionId': '1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com',
        'timestamp': '2019-01-24 02:30:22',
        'caller': '+86138****0022',
```

```
        'called': '+86138****7021',
        'stateCode': 0,
        'stateDesc': 'The user releases the call.',
        'subscriptionId': '****'
    }
});

console.log('jsonBody:', jsonBody);

/**
 * 呼叫事件通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 * @returns
 */
function onCallEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj
    var eventType = jsonObj.eventType; //通知事件类型

    if ('fee' === eventType) {
        console.log('EventType error:', eventType);
        return;
    }

    if (!jsonObj.hasOwnProperty('statusInfo')) {
        console.log('param error: no statusInfo.');
        return;
    }
    var statusInfo = jsonObj.statusInfo; //呼叫状态事件信息

    console.log('eventType:', eventType); //打印通知事件类型
    //callin：呼入事件
    if ('callin' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //callout：呼出事件
    if ('callout' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         * 'subscriptionId': 绑定关系ID
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //alerting：振铃事件
    if ('alerting' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
```

```
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
   //answer：应答事件
   if ('answer' === eventType) {
       /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
   //disconnect：挂机事件
   if ('disconnect' === eventType) {
       /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'stateCode': 通话挂机的原因值
        * 'stateDesc': 通话挂机的原因值的描述
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
}

//呼叫事件处理
onCallEvent(jsonBody);
```

话单通知接口

```
/**
 * 话单通知
 * 客户平台收到隐私保护通话平台的话单通知的接口通知
 */

//话单通知样例
var jsonBody = JSON.stringify({
    'eventType': 'fee',
    'feeLst': [
        {
            'direction': 1,
            'spId': '****',
            'appKey': '****',
            'icid': 'ba171f34e6953fcd751edc77127748f4.3757285803.338666679.5',
            'bindNum': '+86138****0022',
            'sessionId': '1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com',
            'subscriptionId': '****',
            'callerNum': '+86138****0021',
```

```
            'calleeNum': '+86138****0022',
            'fwdDisplayNum': '+86138****0022',
            'fwdDstNum': '+86138****7021',
            'callInTime': '2019-01-24 02:30:03',
            'fwdStartTime': '2019-01-24 02:30:03',
            'fwdAlertingTime': '2019-01-24 02:30:04',
            'fwdAnswerTime': '2019-01-24 02:30:06',
            'callEndTime': '2019-01-24 02:30:22',
            'fwdUnaswRsn': 0,
            'ulFailReason': 0,
            'sipStatusCode': 0,
            'callOutUnaswRsn': 0,
            'recordFlag': 1,
            'recordStartTime': '2019-01-24 02:30:06',
            'recordDomain': '****.com',
            'recordBucketName': '****',
            'recordObjectName': '****.wav',
            'ttsPlayTimes': 0,
            'ttsTransDuration': 0,
            'mptyId': '****',
            'serviceType': '003',
            'hostName': 'callenabler246.huaweicaas.com'
        }
    ]
});

console.log('jsonBody:', jsonBody);

/**
 * 话单通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 * @returns
 */
function onFeeEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj
    var eventType = jsonObj.eventType; //通知事件类型

    if ('fee' !== eventType) {
        console.log('EventType error:', eventType);
        return;
    }

    if (!jsonObj.hasOwnProperty('feeLst')) {
        console.log('param error: no feeLst.');
        return;
    }
    var feeLst = jsonObj.feeLst; //呼叫话单事件信息
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
     *
     * 'direction': 通话的呼叫方向
     * 'spId': 客户的云服务账号
     * 'appKey': 商户应用的AppKey
     * 'icid': 呼叫记录的唯一标识
     * 'bindNum': 隐私保护号码
     * 'sessionId': 通话链路的唯一标识
     * 'callerNum': 主叫号码
     * 'calleeNum': 被叫号码
     * 'fwdDisplayNum': 转接呼叫时的显示号码
     * 'fwdDstNum': 转接呼叫时的转接号码
     * 'callInTime': 呼入的开始时间
     * 'fwdStartTime': 转接呼叫操作的开始时间
     * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
     * 'fwdAnswerTime': 转接呼叫操作后的应答时间
     * 'callEndTime': 呼叫结束时间
     * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
     * 'failTime': 呼入,呼出的失败时间
     * 'ulFailReason': 通话失败的拆线点
```

```
    * 'sipStatusCode': 呼入,呼出的失败SIP状态码
    * 'recordFlag': 录音标识
    * 'recordStartTime': 录音开始时间
    * 'recordObjectName': 录音文件名
    * 'recordBucketName': 录音文件所在的目录名
    * 'recordDomain': 存放录音文件的域名
    * 'serviceType': 携带呼叫的业务类型信息
    * 'hostName': 话单生成的服务器设备对应的主机名
    * 'subscriptionId': 绑定关系ID
    */
   //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
   if (feeLst.length > 1) {
       for ( var i=0; i<feeLst.length; i++) {
           if (feeLst[i].hasOwnProperty('sessionId')) {
               console.log('sessionId:', feeLst[i].sessionId);
           }
       }
   } else if(feeLst.length === 1) {
       if (feeLst[0].hasOwnProperty('sessionId')) {
           console.log('sessionId:', feeLst[0].sessionId);
       }
   } else {
       console.log('feeLst error: no element.');
   }
}

//话单处理
onFeeEvent(jsonBody);
```

短信通知接口

```
/**
 * 短信通知
 * 客户平台收到隐私保护通话平台的短信通知的接口通知
 */

//短信通知样例
var jsonBody = JSON.stringify({
    'appKey': '****',
    'smsEvent': {
        'smsIdentifier': '****',
        'notificationMode': 'Block',
        'calling': '+86138****0001',
        'virtualNumber': '+86138****0000',
        'event': 'TextSMS',
        'timeStamp': '2018-09-13T09:46:16.023Z'
    }
});

console.log('jsonBody:', jsonBody);

/**
 * 短信通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 * @returns
 */
function onSmsEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj

    if (!jsonObj.hasOwnProperty('smsEvent')) {
        console.log('param error: no smsEvent.');
        return;
    }

    //console.log('appKey:', jsonObj.appKey); //商户应用的AppKey

    var smsEvent = jsonObj.smsEvent; //短信通知信息
```

```
    /**
     * Example: 此处以解析notificationMode为例,请按需解析所需参数并自行实现相关处理
     *
     * 'smsIdentifier': 短信唯一标识
     * 'notificationMode': 通知模式
     * 'calling': 真实发送方号码
     * 'called': 真实接收方号码
     * 'virtualNumber': 隐私号码(X号码)
     * 'event': 短信状态事件
     * 'timeStamp': 短信事件发生的系统时间戳,UTC时间
     * 'subscriptionId': 绑定ID
     * 'smsContent': 用户发送的短信内容
     */
    if (smsEvent.hasOwnProperty('notificationMode')) {
        if ('Block' === smsEvent.notificationMode) {
            //收到隐私保护通话平台的短信通知,若为Block模式,请参考接口文档回消息指示下一步操作
            var actions = {
                'operation': 'vNumberRoute', //操作类型:转发短信/丢弃短信
                'message': {
                    'called': '+86138****7022', //真实接收方号码
                    'calling': '+86138****7021' //真实发送方号码
                }
            };
            var resp = JSON.stringify({'actions': [actions]}); //Block模式响应消息
            console.log('resp:', resp);
        } else if ('Notify' === smsEvent.notificationMode) {
            //收到隐私保护通话平台的短信通知,若为Notify模式,请回HTTP状态码为200的空消息
            //var statusCode = 200;
            console.log('This is AX sms Notify mode.');
        } else {
            console.log('notificationMode param error.');
        }
    }
}

//短信通知处理
onSmsEvent(jsonBody);
```

3.5.3 AXE 模式

样例	**AXE模式绑定接口** **AXE模式解绑接口** **AXE模式绑定信息查询接口** **获取录音文件下载地址接口** **呼叫事件通知接口** **话单通知接口**
环境要求	基于Node.js 8.12.0版本，要求Node.js 7.0.0及以上版本。

须知

- 本文档所述Demo在提供服务的过程中，可能会涉及个人数据的使用，建议您遵从国家的相关法律采取足够的措施，以确保用户的个人数据受到充分的保护。
- 本文档所述Demo仅用于功能演示，不允许客户直接进行商业使用。
- 本文档信息仅供参考，不构成任何要约或承诺。
- 本文档接口携带参数只是用作参考，不可以直接复制使用，填写参数需要替换为实际值，请参考“**开发准备**”获取所需数据。

AXE 模式绑定接口

```
/*jshint esversion: 6 */
var https = require('https'); // 引入https模块
var url = require('url'); // 引入url模块
var fs = require('fs'); // 引入fs模块

// 必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0'; // APP接入地址
+接口访问URI
var appKey = 'a1********'; // APP_Key
var appSecret = 'cfc8********'; // APP_Secret
var virtualNum = '+86180****0001'; // AXE中的X号码
var bindNum = '+86186****5678'; // AXE中的A号码

/*
 * 选填,各参数要求请参考"AXE模式绑定接口"
 */
//var areaCode = '0755'; // 需要绑定的X号码对应的城市码
//var displayNumMode = '0'; // 非A用户呼叫X号码时，A看到的主显号码
//var recordFlag = 'false'; // 是否需要针对该绑定关系产生的所有通话录音
//var recordHintTone = 'recordHintTone.wav'; // 设置录音提示音
//var callbackTone = 'callbackTone.wav'; // A呼叫X不存在回呼记录的提示音
//var callbackNum = '+86180****0021'; // A呼叫X不存在回呼记录的转接号码
//var bindExpiredTime = 24; // 绑定关系的有效时间
//var callbackExpiredTime = 24; // 回呼记录有效时间
//var userData = 'abcdefg'; // 用户自定义数据

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

var urlobj = url.parse(realUrl); // 解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, // 主机名
    port: urlobj.port, // 端口
    path: urlobj.pathname, // URI
    method: 'POST', // 请求方法为POST
    headers: { // 请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
```

```
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var body = JSON.stringify({
    'virtualNum':virtualNum,
    'bindNum':bindNum,
// 'areaCode':areaCode,
// 'displayNumMode':displayNumMode,
// 'recordFlag':recordFlag,
// 'recordHintTone':recordHintTone,
// 'callbackTone':callbackTone,
// 'callbackNum':callbackNum,
// 'bindExpiredTime':bindExpiredTime,
// 'callbackExpiredTime':callbackExpiredTime,
// 'userData':userData
});

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); // 打印响应码
    console.log('headers:', res.headers); // 打印响应Headers

    res.setEncoding('utf8'); // 设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); // 打印响应数据
        fs.appendFileSync('bind_data.txt', '绑定结果：' + d + '\n'); //绑定ID很重要,请记录到本地文件,方便后续修改绑定关系及解绑
    });
});
req.on('error', (e) => {
    console.error(e.message); // 请求错误时,打印错误信息
});
req.write(body); // 发送请求Body数据
fs.appendFileSync('bind_data.txt', '绑定请求数据：' + body + '\n'); //绑定请求参数记录到本地文件,方便定位问题
req.end(); // 结束请求
fs.closeSync(0); //结束文件操作
```

AXE 模式解绑接口

```
/*jshint esversion: 6 */
var https = require('https'); // 引入https模块
var url = require('url'); // 引入url模块
var querystring = require('querystring'); //引入querystring模块

// 必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0'; // APP接入地址+接口访问URI
var appKey = 'a1********'; // APP_Key
var appSecret = 'cfc8********'; // APP_Secret

/*
 * 选填,各参数要求请参考"AXE模式解绑接口"
 * subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
 */
var subscriptionId = '****'; // 指定"AXE模式绑定接口"返回的绑定ID进行解绑
var virtualNum = '+86180****0001'; // AXE中的X号码
var extendNum = '1234'; // AXE中的E号码

/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
```

```
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'subscriptionId': subscriptionId,
    'virtualNum': virtualNum,
    'extendNum': extendNum
});

var urlobj = url.parse(realUrl); // 解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, // 主机名
    port: urlobj.port, // 端口
    path: urlobj.pathname + "?" + ops, // URI+请求url参数
    method: 'DELETE', // 请求方法为DELETE
    headers: { // 请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); // 打印响应码
    console.log('headers:', res.headers); // 打印响应Headers

    res.setEncoding('utf8'); // 设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); // 打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); // 请求错误时,打印错误信息
});
req.end(); // 结束请求
```

AXE 模式绑定信息查询接口

```
/*jshint esversion: 6 */
var https = require('https'); // 引入https模块
var url = require('url'); // 引入url模块
var fs = require('fs'); // 引入fs模块
var querystring = require('querystring'); //引入querystring模块

// 必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/caas/extendnumber/v1.0'; // APP接入地址
+接口访问URI
var appKey = 'a1********'; // APP_Key
var appSecret = 'cfc8********'; // APP_Secret

/*
 * 选填,各参数要求请参考"AXE模式绑定信息查询接口"
 * subscriptionId和(virtualNum+extendNum)二选一即可,当都传入时,优先选用subscriptionId
 */
var subscriptionId = '****'; // 指定"AXE模式绑定接口"返回的绑定ID进行查询
var virtualNum = '+86180****0001'; // AXE中的X号码
var extendNum = '1234'; // AXE中的E号码

/**
 * Build the X-AKSK value.
 * @param appKey
```

```
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'subscriptionId': subscriptionId,
    'virtualNum': virtualNum,
    'extendNum': extendNum
});

var urlobj = url.parse(realUrl); // 解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, // 主机名
    port: urlobj.port, // 端口
    path: urlobj.pathname + "?" + ops, // URI+请求url参数
    method: 'GET', // 请求方法为GET
    headers: { // 请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)
    },
    rejectUnauthorized: false // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); // 打印响应码
    console.log('headers:', res.headers); // 打印响应Headers

    res.setEncoding('utf8'); // 设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); // 打印响应数据
        fs.appendFileSync('bind_data.txt', '绑定查询结果：' + d + '\n'); //查询结果,记录到本地文件
    });
});
req.on('error', (e) => {
    console.error(e.message); // 请求错误时,打印错误信息
});
req.end(); // 结束请求
fs.closeSync(0); //结束文件操作
```

获取录音文件下载地址接口

```
/*jshint esversion: 6 */
var https = require('https'); // 引入https模块
var url = require('url'); // 引入url模块
var fs = require('fs'); // 引入fs模块
var querystring = require('querystring'); //引入querystring模块

// 必填,请参考"开发准备"获取如下数据,替换为实际值
var realUrl = 'https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/voice/record/v1.0'; // APP接入地址
+接口访问URI
var appKey = 'a1********'; // APP_Key
var appSecret = 'cfc8********'; // APP_Secret

// 必填,通过"话单通知接口"获取
var recordDomain = '****.com'; // 录音文件存储的服务器域名
var fileName = '****.wav'; // 录音文件名
```

```
/**
 * Build the X-AKSK value.
 * @param appKey
 * @param appSecret
 * @returns
 */
function buildAKSKHeader(appKey, appSecret){
    var crypto = require('crypto');
    var util = require('util');
    var time = new Date(Date.now()).toISOString().replace(/.[0-9]+\Z/, 'Z'); //Created
    var nonce = crypto.randomBytes(16).toString('hex'); //Nonce
    var passwordDigestBase64Str = crypto.createHmac('sha256', appSecret).update(nonce +
time).digest('base64'); //PasswordDigest
    return util.format('UsernameToken Username="%s",PasswordDigest="%s",Nonce="%s",Created="%s"',
appKey, passwordDigestBase64Str, nonce, time);
}

//请求url参数
var ops = querystring.stringify({
    'recordDomain': recordDomain,
    'fileName': fileName
});

var urlobj = url.parse(realUrl); // 解析realUrl字符串并返回一个URL对象

var options = {
    host: urlobj.hostname, // 主机名
    port: urlobj.port, // 端口
    path: urlobj.pathname + "?" + ops, // URI+请求url参数
    method: 'GET', // 请求方法为GET
    headers: { // 请求Headers
        'Content-Type': 'application/json;charset=UTF-8',
        'Authorization': 'AKSK realm="SDP",profile="UsernameToken",type="Appkey"',
        'X-AKSK': buildAKSKHeader(appKey, appSecret)

    },
    rejectUnauthorized: false // 为防止因HTTPS证书认证失败造成API调用失败,需要先忽略证书信任问题
};

var req = https.request(options, (res) => {
    console.log('statusCode:', res.statusCode); // 打印响应码
    console.log('headers:', res.headers); // 打印响应Headers

    if(301 === res.statusCode){
        var link = Object.getOwnPropertyDescriptor(res.headers, 'location').value;
        fs.appendFileSync('bind_data.txt', '获取录音文件下载地址：' + link + '\n'); //查询结果,记录到本地文件
    }

    res.setEncoding('utf8'); // 设置响应数据编码格式
    res.on('data', (d) => {
        console.log('resp:', d); // 打印响应数据
    });
});
req.on('error', (e) => {
    console.error(e.message); // 请求错误时,打印错误信息
});
req.end(); // 结束请求
fs.closeSync(0); //结束文件操作
```

呼叫事件通知接口

```
/**
 * 呼叫事件通知
 * 客户平台收到隐私保护通话平台的呼叫事件通知的接口通知
 */

//呼叫事件通知样例
var jsonBody = JSON.stringify({
```

```
    'eventType': 'disconnect',
    'statusInfo': {
        'sessionId': '1202_1051_4294967295_20190124070250@callenabler246.huaweicaas.com',
        'timestamp': '2019-01-24 07:03:28',
        'caller': '+86138****0022',
        'called': '+86138****7021',
        'stateCode': 0,
        'stateDesc': 'The user releases the call.',
        'subscriptionId': '********'
    }
});

console.log('jsonBody:', jsonBody);

/**
 * 呼叫事件通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 */
function onCallEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj
    var eventType = jsonObj.eventType; //通知事件类型

    if ('fee' === eventType) {
        console.log('EventType error:', eventType);
        return;
    }

    if (!jsonObj.hasOwnProperty('statusInfo')) {
        console.log('param error: no statusInfo.');
        return;
    }
    var statusInfo = jsonObj.statusInfo; //呼叫状态事件信息

    console.log('eventType:', eventType); //打印通知事件类型
    //callin：呼入事件
    if ('callin' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'caller': 主叫号码
         * 'called': 被叫号码
         */
        if (statusInfo.hasOwnProperty('sessionId')) {
            console.log('sessionId:', statusInfo.sessionId);
        }
        return;
    }
    //collectInfo：放音收号结果事件,仅AXE模式下的A被叫场景携带
    if ('collectInfo' === eventType) {
        /**
         * Example: 此处以解析digitInfo为例,请按需解析所需参数并自行实现相关处理
         *
         * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
         * 'sessionId': 通话链路的标识ID
         * 'digitInfo': AXE场景中携带收号结果(即用户输入的数字)
         */
        if (statusInfo.hasOwnProperty('digitInfo')) {
            console.log('digitInfo:', statusInfo.digitInfo);
        }
        return;
    }
    //callout：呼出事件
    if ('callout' === eventType) {
        /**
         * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
         *
```

```
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
   //alerting：振铃事件
   if ('alerting' === eventType) {
       /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
   //answer：应答事件
   if ('answer' === eventType) {
       /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
   //disconnect：挂机事件
   if ('disconnect' === eventType) {
       /**
        * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
        *
        * 'timestamp': 呼叫事件发生时隐私保护通话平台的UNIX时间戳
        * 'sessionId': 通话链路的标识ID
        * 'caller': 主叫号码
        * 'called': 被叫号码
        * 'stateCode': 通话挂机的原因值
        * 'stateDesc': 通话挂机的原因值的描述
        * 'subscriptionId': 绑定关系ID
        */
       if (statusInfo.hasOwnProperty('sessionId')) {
           console.log('sessionId:', statusInfo.sessionId);
       }
       return;
   }
}

//呼叫事件处理
onCallEvent(jsonBody);
```

话单通知接口

```
/**
 * 话单通知
 * 客户平台收到隐私保护通话平台的话单通知的接口通知
 */

//话单通知样例
var jsonBody = JSON.stringify({
    'eventType': 'fee',
    'feeLst': [
        {
            'direction': 0,
            'spId': '****',
            'appKey': '********',
            'icid': 'ba171f34e6953fcd751edc77127748f4.3757289590.338833305.5',
            'bindNum': '+86138****0022',
            'sessionId': '1201_11275_4294967295_20190124033310@callenabler246.huaweicaas.com',
            'subscriptionId': '********',
            'callerNum': '+86138****7021',
            'calleeNum': '+86138****0022',
            'fwdDisplayNum': '+86138****0022',
            'fwdDstNum': '+86138****0021',
            'callInTime': '2019-01-24 03:33:10',
            'fwdStartTime': '2019-01-24 03:33:16',
            'fwdAlertingTime': '2019-01-24 03:33:19',
            'fwdAnswerTime': '2019-01-24 03:33:28',
            'callEndTime': '2019-01-24 03:33:57',
            'fwdUnaswRsn': 0,
            'ulFailReason': 0,
            'sipStatusCode': 0,
            'callOutUnaswRsn': 0,
            'recordFlag': 1,
            'recordStartTime': '2019-01-24 03:33:28',
            'recordDomain': '****.com',
            'recordBucketName': '****',
            'recordObjectName': '********.wav',
            'ttsPlayTimes': 0,
            'ttsTransDuration': 0,
            'mptyId': '********',
            'serviceType': '005',
            'hostName': 'callenabler246.huaweicaas.com',
            'extendNumber': '02'
        }
    ]
});

console.log('jsonBody:', jsonBody);

/**
 * 话单通知
 * @brief 详细内容以接口文档为准
 * @param jsonBody
 */
function onFeeEvent(jsonBody) {
    var jsonObj = JSON.parse(jsonBody); //将通知消息解析为jsonObj
    var eventType = jsonObj.eventType; //通知事件类型

    if ('fee' !== eventType) {
        console.log('EventType error:', eventType);
        return;
    }

    if (!jsonObj.hasOwnProperty('feeLst')) {
        console.log('param error: no feeLst.');
        return;
    }
    var feeLst = jsonObj.feeLst; //呼叫话单事件信息
    /**
     * Example: 此处以解析sessionId为例,请按需解析所需参数并自行实现相关处理
```



```
    *
    * 'direction': 通话的呼叫方向
    * 'spId': 客户的云服务账号
    * 'appKey': 商户应用的AppKey
    * 'icid': 呼叫记录的唯一标识
    * 'bindNum': 隐私保护号码
    * 'sessionId': 通话链路的唯一标识
    * 'callerNum': 主叫号码
    * 'calleeNum': 被叫号码
    * 'fwdDisplayNum': 转接呼叫时的显示号码
    * 'fwdDstNum': 转接呼叫时的转接号码
    * 'callInTime': 呼入的开始时间
    * 'fwdStartTime': 转接呼叫操作的开始时间
    * 'fwdAlertingTime': 转接呼叫操作后的振铃时间
    * 'fwdAnswerTime': 转接呼叫操作后的应答时间
    * 'callEndTime': 呼叫结束时间
    * 'fwdUnaswRsn': 转接呼叫操作失败的Q850原因值
    * 'failTime': 呼入,呼出的失败时间
    * 'ulFailReason': 通话失败的拆线点
    * 'sipStatusCode': 呼入,呼出的失败SIP状态码
    * 'recordFlag': 录音标识
    * 'recordStartTime': 录音开始时间
    * 'recordObjectName': 录音文件名
    * 'recordBucketName': 录音文件所在的目录名
    * 'recordDomain': 存放录音文件的域名
    * 'serviceType': 携带呼叫的业务类型信息
    * 'hostName': 话单生成的服务器设备对应的主机名
    * 'subscriptionId': 绑定关系ID
    * 'extendNum': 分机号E
    */
   //短时间内有多个通话结束时隐私保护通话平台会将话单合并推送,每条消息最多携带50个话单
   if (feeLst.length > 1) {
       for ( var i=0; i<feeLst.length; i++) {
           if (feeLst[i].hasOwnProperty('sessionId')) {
               console.log('sessionId:', feeLst[i].sessionId);
           }
       }
   } else if(feeLst.length === 1) {
       if (feeLst[0].hasOwnProperty('sessionId')) {
           console.log('sessionId:', feeLst[0].sessionId);
       }
   } else {
       console.log('feeLst error: no element.');
   }
}

//话单处理
onFeeEvent(jsonBody);
```

4 API 错误码处理

> **说明**
>
> 本章节响应码数量较多，建议您使用快捷键**Ctrl+F**在界面进行搜索，能更快地找到您需要的错误码处理建议。

接口调用 404 问题处理

若调用接口时返回了404响应，请检查**APP接入地址**和访问URI（详见**API参考文档**）是否都填写正确，且拼接成了完整的请求URL，如“https://rtcpns.cn-north-1.myhuaweicloud.com/rest/provision/caas/privatenumber/v1.0”。

接口调用错误码处理

调用**API接口**会产生接口调用响应结果码，响应示例如下：

注：请根据响应码和结果码查看处理方法。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
Content-Length: xx

{
"resultcode":"0",
"resultdesc":"Success",
"origNum":"+86138****8888",
"privateNum":"+86138****6666",
"subscriptionId":"******"
}
```

表 4-1 响应结果码

响应码	结果码	英文描述	中文描述	处理方法
200	0	Success.	成功。	-
400	1023006	Authorization not contained in the HTTP header.	HTTP消息头未找到Authorization字段。	请检查HTTP消息头中是否携带了Authorization字段。

响应码	结果码	英文描述	中文描述	处理方法
	1023007	realm not contained in Authorization.	Authorization字段中未找到realm属性。	请检查Authorization字段中的是否携带了realm属性。
	1023008	profile not contained in Authorization.	Authorization字段中未找到profile属性。	请检查Authorization字段中的是否携带了profile属性。
	1023009	The value of realm in Authorization must be SDP.	Authorization中realm属性值应该为“SDP”。	请检查Authorization字段中的realm属性值是否为“SDP”。
	1023010	The value of profile in Authorization must be UsernameToken.	Authorization中profile属性值应该为“UsernameToken”。	请检查Authorization字段中的profile属性值是否为“UsernameToken”。
	1023011	The value of type in Authorization must be app_key.	Authorization中type属性值应该为“Appkey”。	请检查Authorization字段中的type属性值是否为“Appkey”。
	1023012	type not contained in Authorization.	Authorization字段中未找到type属性。	请检查Authorization字段中是否携带了type属性。
	1023033	HTTP header not found X-AKSK field	HTTP头未找到X-AKSK字段	请检查HTTP消息头中是否携带了X-AKSK字段。
	1023034	UserName not contained in X-AKSK.	X-AKSK字段中未找到UserName属性。	请检查X-AKSK字段中的是否携带了Username属性。
	1023035	Nonce not contained in X-AKSK.	X-AKSK字段中未找到Nonce属性。	请检查X-AKSK字段中的是否携带了Nonce属性。
	1023036	Created not contained in X-AKSK.	X-AKSK字段中未找到Created属性。	请检查X-AKSK字段中的是否携带了Created属性。
	1023037	PasswordDigest not contained in X-AKSK.	X-AKSK字段中未找到PasswordDigest属性。	请检查X-AKSK字段中的是否携带了PasswordDigest属性。

响应码	结果码	英文描述	中文描述	处理方法
	1023038	UsernameToken not contained in X-AKSK.	X-AKSK中没有携带UsernameToken。	请检查X-AKSK字段中的是否携带了UsernameToken属性。
401	1010010	Invalid digest.	PasswordDigest校验失败。	请检查PasswordDigest字段填写是否正确。
	1010013	Time out limit.	时间超出限制。	请确认X-AKSK鉴权时，生成随机数的时间与发送请求时的本地时间不能相差太大（具体差值请与管理员确认）。
403	1010002	Invalid request.	无效请求。	参考各接口参数说明，检查请求携带的参数格式是否正确，如以下参数格式问题： • 绑定接口填写的号码参数需为全局号码格式，如+86138****0001或+8675528****01； • 放音文件需上传通过审核才可通过接口调用，**点击查看如何上传审核**。 • 参考接口参数说明，检查是否携带了不能同时携带的参数，如AXE模式绑定接口不能同时携带callbackTone和callbackNum，若是，请保留一个； • 参数长度或格式是否错误，如AXE模式分机号长度或timeUnit的格式是否正确。
	1010003	Invalid app_key.	无效的app_key。	请检查请求携带的app_key填写是否正确。
	1010008	The status of the app_key is unavailable.	app_key状态异常。	请检查请求携带的app_key所属应用状态是否正常。应用状态可登录控制台后在“应用管理”界面查看。
	1010010	The flow control upper limit is reached on the platform.	平台达到系统流控上限。	请稍等一分钟后再试。

响应码	结果码	英文描述	中文描述	处理方法
	1010029	The subscriber status is frozen.	用户账号已冻结。	查看账户是否欠费。 • 如欠费需充值后才能继续使用。 • 若未欠费，请联系华为云客服处理。
	1010040	The app_key is not allowed to invoke the API.	app_key没有调用本API的权限。	出现该错误码表示调用的接口和app_key所属的应用模式不一致。如添加应用时选择的AXB模式的应用，调用接口时只能调用AXB模式的接口，不能调用其他模式的接口。
	1016002	The record already exists.	记录已经存在。	出现该错误码表示调用**AX模式绑定接口**时指定的A号码（origNum）和X号码（privateNum）之间已经存在绑定关系，请更换origNum或privateNum参数的值。
	1011001	Account does not exist.	账号不存在。	出现该错误码可能有以下两个原因： • 调用**AX模式相关接口**时指定的X号码（privateNum）可能不是该应用已申请的隐私号码，请确认privateNum参数的填写是否正确； • 调用**AX模式相关接口**时填写的X号码（privateNum）格式不正确，请根据接口文档修改号码格式后再次尝试。
	1011002	Insufficient number resources.	号码资源不足。	出现该错误码表示调用**AX模式绑定接口**时没有可分配的X号码，请申请新的号码资源或修改areaCode的值。**点击查看处理方法**
	1011003	Exceeded the upper limit of resources that can be applied for.	超过允许申请的资源上限。	出现该错误码表示调用**AX模式绑定接口**时指定的A号码已绑定了五个X号码，请更换origNum参数的值。AX模式中一个A号码只能绑定五个X号码。
	1011004	The number is not applied for binding application.	携带的X号码和app_key没有绑定关系。	出现该错误码表示调用**AX模式解绑接口**或**AX模式绑定信息修改接口**时携带的app_key和X号码没有绑定关系，请检查携带的X号码是否属于该应用。

响应码	结果码	英文描述	中文描述	处理方法
	1012001	Resource of number is not to be applied.	资源未申请。	出现该错误码表示调用解绑或查询绑定关系接口时携带的app_key和X号码没有绑定关系，请检查携带的X号码是否属于该应用。
	1012007	The record does not exist.	记录不存在。	• 若调用**AXB模式绑定接口**、**AXB模式解绑接口**、**AXB模式绑定信息修改接口**或**AXB模式绑定信息查询接口**时出现该错误码，表示隐私保护通话平台未查询到绑定关系，请检查携带的relationNum或subscriptionId参数是否属于该应用。 • 若调用**获取录音文件下载地址接口**时出现该错误码，表示隐私保护通话平台未查询到录音信息，请确认携带的fileName参数是否填写正确。
	1012008	Insufficient number of resources.	号码资源不足。	没有可分配的X号码，请申请新的号码资源或修改areaCode的值。**点击查看处理方法**
	1012009	Maximum number of resources has been exceeded.	指定的X号码的绑定数量已达到上限。	出现该错误码表示调用**AXB模式绑定接口**指定的X号码已经绑定了5000对关系。请修改relationNum的值，或者解除指定的X号码上的部分绑定关系。AXB模式中一个X号码只能同时绑定5000对关系。
	1012010	The relation number has been bound.	绑定关系已存在。	出现该错误码表示调用**AXB模式绑定接口**时携带的X号码（relationNum）和A号码（callerNum）或B号码（calleeNum）已存在绑定关系，可确认后更换其他X号码进行绑定。
	1012012	Application does not open recording function.	应用未开启录音功能。	出现该错误码表示添加应用时未开启录音功能，**点击查看如何开启录音功能**。

响应码	结果码	英文描述	中文描述	处理方法
	101212102	The number status is abnormal.	号码状态异常。	出现该错误码表示调用接口时指定的X号码因投诉或号码状态异常被隐私保护通话平台加入了黑名单。 请查看订购号码时填写的邮箱是否有业务下线通知邮件，如果没有，请拨打400电话联系华为云客服处理。
	1011005	Resources have been allocated.	资源已经分配。	出现该错误码表示调用**AX模式绑定接口**时指定的X号码（privateNum）已和其他A号码绑定，可更换其他X号码进行绑定。 如果该X号码的绑定关系可以解除，您还可以调用**AX模式解绑接口**解除该绑定关系后，再使用该X号码进行绑定。
	1016001	The record does not exist.	记录不存在。	• 如果调用**AX模式解绑接口**时出现该错误码，表示调用接口时指定的origNum或者subscriptionId参数不正确，未查询到绑定关系，请确认参数是否正确。 • 如果调用**AXE模式解绑接口**时出现该错误码，表示调用接口时指定的virtualNum，extendNum或subscriptionId参数不正确，未查询到绑定关系，请确认参数是否正确。
	1023005	Virtual number over license limit.	隐私号码超出license限制。	请联系客服处理。
	1020166	The app client ip is not in ip white list.	对端app IP不在白名单列表中。	联系客服检查IP白名单是否配置正确。
	1020167	No idle extend Number.	没有空闲的分机号。	出现该错误码表示调用**AXE模式绑定接口**时指定的分机号（extendNum）已被占用，请重新指定分机号。

响应码	结果码	英文描述	中文描述	处理方法
	1013102	The extend number has been bound.	绑定关系已存在。	出现该错误码表示调用**AXE模式绑定接口**时指定的X号码（virtualNum）和A号码（bindNum）已有绑定关系，无需再次绑定。
500	1010001	Internal system error.	系统错误。	请联系客服处理。

5 挂机原因值、Q850 原因值、呼叫拆线点

调用接口会产生接口调用错误码，详见**API错误码处理**。

调用接口成功后，如果用户在**添加应用**时指定了呼叫状态接收地址和呼叫话单接收地址，则隐私保护通话平台在接收到南向网元返回的呼叫状态通知和话单通知时，会主动将呼叫状态通知和话单通知推送给客户。

话单消息示例如下：

```
POST /fee HTTP/1.1

Content-Length: xx

{"eventType":"fee","feeLst":
[{"direction":1,"spId":"****","appKey":"****","icid":"ba171f34e6953fcd751edc77127748f4.3757285803.338666679.5","bindNum":"+86138****0022","sessionId":"1200_1827_4294967295_20190124023003@callenabler246.huaweicaas.com","subscriptionId":"****","callerNum":"+86138****0021","calleeNum":"+86138****0022","fwdDisplayNum":"+86138****0022","fwdDstNum":"+86138****7021","callInTime":"2019-01-24 02:30:03","fwdStartTime":"2019-01-24 02:30:03","fwdAlertingTime":"2019-01-24 02:30:04","fwdAnswerTime":"2019-01-24 02:30:06","callEndTime":"2019-01-24 02:30:22","fwdUnaswRsn":0,"ulFailReason":0,"sipStatusCode":0,"callOutUnaswRsn":0,"recordFlag":1,"recordStartTime":"2019-01-24 02:30:06","recordDomain":"****.com","recordBucketName":"****","recordObjectName":"****.wav","ttsPlayTimes":0,"ttsTransDuration":0,"mptyId":"****","serviceType":"003","hostName":"callenabler246.huaweicaas.com"}]}
```

如果某一通呼叫失败，客户可根据平台推送的挂机原因值（stateCode）、转接呼叫操作失败的Q850原因值（fwdUnaswRsn）或通话失败的拆线点（ulFailReason）查询失败原因。

1. 在隐私保护通话平台推送的呼叫状态通知和话单通知中查询该通呼叫的“stateCode”、“fwdUnaswRsn”和“ulFailReason”。
2. 查看**附录**的挂机原因值、Q850原因值、呼叫拆线点详细说明，判断呼叫失败的原因。

注：“stateCode”请在接收到的呼叫状态通知（disconnect事件）中查看，“fwdUnaswRsn”和“ulFailReason”请在接收到的话单通知中查看。