消息推送方式

1. 对单个用户推送消息

1.1 接口说明

接口名称	支持推送类型	说明
pushMessageToSingle	透传（payload）、点击通知启动应用、点击通知打开网页等	对单个用户(clientid）推送

正常推送不需要传requestId，如果发生异常重试时将requestId传入，具体用法详见示例代码

```
pushMessageToSingle(message, target, requestId)
```

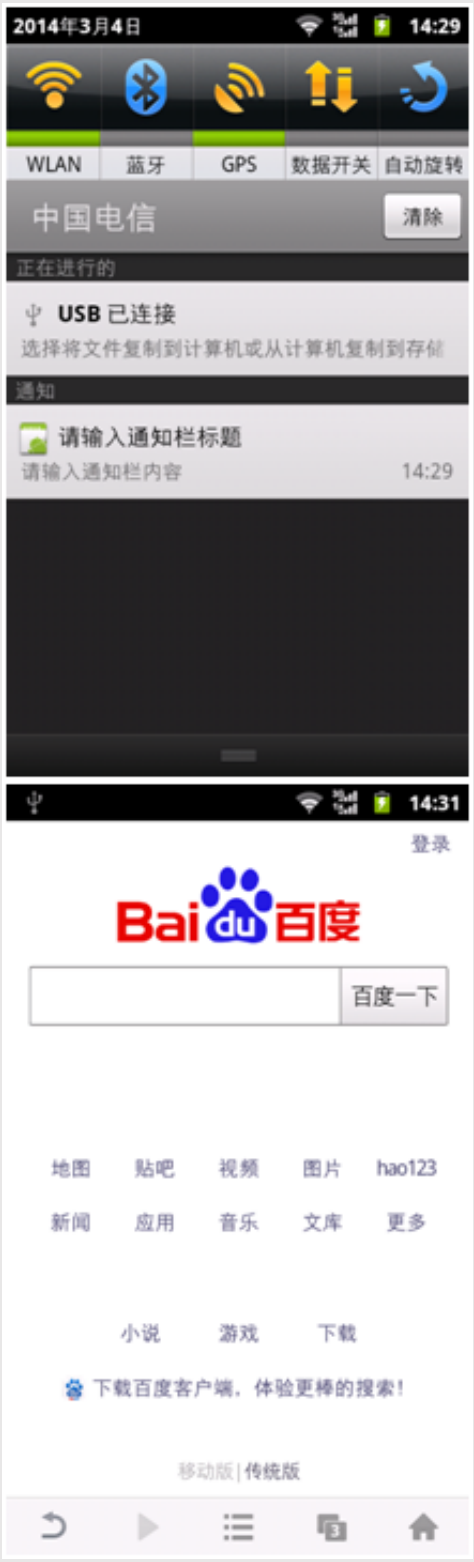
1.2 pushMessageToSingle代码实例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

namespace GetuiServerApiSDKDemo
{
    public class demo
    {

        //采用"C# SDK 快速入门"， "第二步 获取访问凭证 "中获得的应用配置,用户可自行替换
        private static String APPID = "";
        private static String APPKEY = "";
```

```
        private static String MASTERSECRET = "";
        //您获取的clientID
        private static String CLIENTID = "";
        //别名推送方式
        //private static String ALIAS = "";
        //HOST：OpenService接口地址
        private static String HOST =
"http://sdk.open.api.igexin.com/apiex.htm";

        static void Main(string[] args)
        {
            //toList接口每个用户状态返回是否开启，可选
            Console.OutputEncoding = Encoding.GetEncoding(936);
            Environment.SetEnvironmentVariable("needDetails", "true");


            PushMessageToSingle();
        }

        private static void PushMessageToSingle()
       {

            IGtPush push = new IGtPush(HOST,APPKEY, MASTERSECRET);

            //消息模版：TransmissionTemplate:透传模板

            TransmissionTemplate template =  TransmissionTemplateDemo();


            // 单推消息模型
            SingleMessage message = new SingleMessage();
            message.IsOffline = true;                   // 用户当前不在线时，是否离
线存储,可选
            message.OfflineExpireTime = 1000 * 3600 * 12;         // 离线有效时
间，单位为毫秒，可选
            message.Data = template;
            //判断是否客户端是否wifi环境下推送，2为4G/3G/2G，1为在WIFI环
境下，0为不限制环境
            //message.PushNetWorkType = 1;

            com.igetui.api.openservice.igetui.Target target = new
com.igetui.api.openservice.igetui.Target();
            target.appId = APPID;
            target.clientId = CLIENTID;
            //target.alias = ALIAS;
            try
            {
```

```
            String pushResult = push.pushMessageToSingle(message, target);

            System.Console.WriteLine("----------------------------------------
-------");
            System.Console.WriteLine("----------------------------------------
-------");
            System.Console.WriteLine("----------------服务端返回结果：" +
pushResult);
        }
        catch (RequestException e)
        {
            String requestId=e.RequestId;
            //发送失败后的重发
            String pushResult = push.pushMessageToSingle(message, target,
requestId);
            System.Console.WriteLine("----------------------------------------
-------");
            System.Console.WriteLine("----------------------------------------
-------");
            System.Console.WriteLine("----------------服务端返回结果：" +
pushResult);
        }
    }


    //透传模板动作内容
     public  static TransmissionTemplate TransmissionTemplateDemo()
     {
        TransmissionTemplate template = new TransmissionTemplate();
        template.AppId = APPID;
        template.AppKey = APPKEY;
        //应用启动类型，1：强制应用启动 2：等待应用启动
        template.TransmissionType = 1;
        //透传内容
        template.TransmissionContent = "透传内容";
        //设置通知定时展示时间，结束时间与开始时间相差需大于6分钟，消
息推送后，客户端将在指定时间差内展示消息（误差6分钟）
        String begin = "2015-03-06 14:36:10";
        String end = "2015-03-06 14:46:20";
        template.setDuration(begin, end);

        return template;
     }
  }
}
```

1.3 返回值

字段	返回码
result	请查询返回值
客户端展示	

2. 对指定列表用户推送消息

如果仅对单个用户推送务必使用单推接口，否则会严重影响推送性能，如果对少量甚至几个用户推送同样的消息，建议使用单推实现，性能会更高

2.1 接口说明

接口名称	支持推送类型	说明
pushMessageToList	透传（payload）、点击通知启动应用、点击通知打开网页等	（通过ClientID列表）群推，可查看clientid列表中每个用户的在线状态

2.2 pushMessageToList代码实例

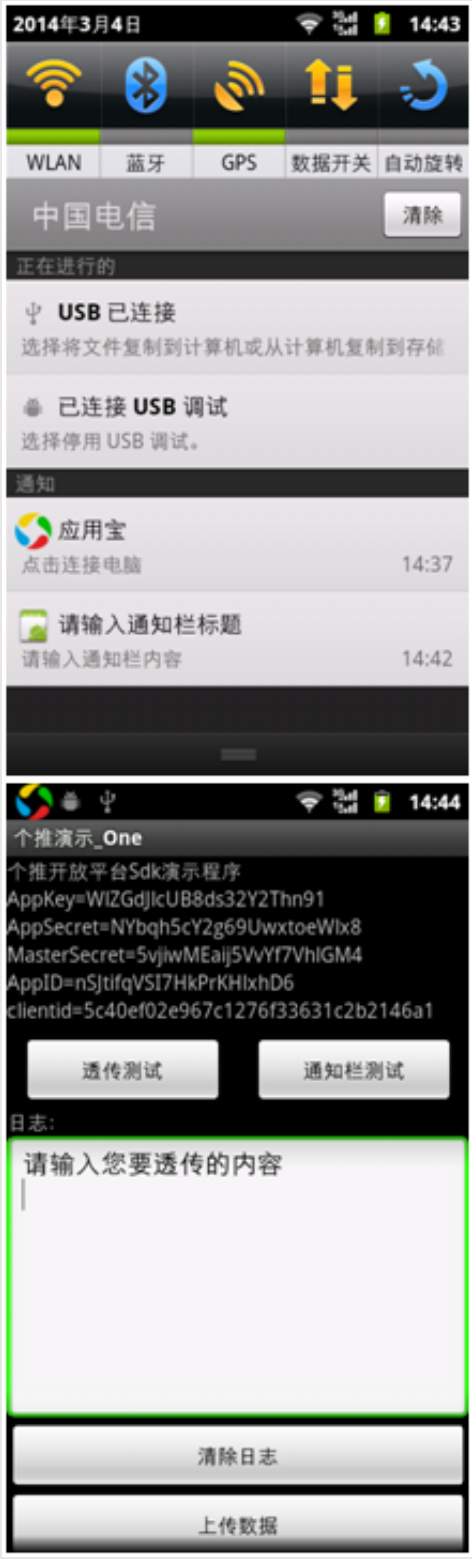
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

namespace GetuiServerApiSDKDemo
{
  public class demo
  {

    //采用"C# SDK 快速入门"， "第二步 获取访问凭证 "中获得的应用配置,用户可自行替换
    private static String APPID = "";
    private static String APPKEY = "";
    private static String MASTERSECRET = "";
    //您获取的clientID
    private static String CLIENTID1 = "";
    private static String CLIENTID2 = "";
    //别名推送方式
    //private static String ALIAS1 = "";
    //private static String ALIAS2 = "";
    //HOST：OpenService接口地址
    private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";

    static void Main(string[] args)
```

```
    {
        //toList接口每个用户状态返回是否开启，可选
        Console.OutputEncoding = Encoding.GetEncoding(936);
        Environment.SetEnvironmentVariable("needDetails", "true");

        //2.PushMessageToList接口
        PushMessageToList();
    }

    //PushMessageToList接口测试代码
    private static void PushMessageToList()
    {
        // 推送主类（方式1，不可与方式2共存）
        IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
        // 推送主类（方式2，不可与方式1共存）此方式可通过获取服务端地址列表判断最快域名后进行消息推送，每10分钟检查一次最快域名
        //IGtPush push = new IGtPush("",APPKEY,MASTERSECRET);
        ListMessage message = new ListMessage();

        NotificationTemplate template =  NotificationTemplateDemo();
        // 用户当前不在线时，是否离线存储,可选
        message.IsOffline = true;
        // 离线有效时间，单位为毫秒，可选
        message.OfflineExpireTime = 1000 * 3600 * 12;
        message.Data = template;
        //message.PushNetWorkType = 0;      //判断是否客户端是否wifi环境下推送，1为在WIFI环境下，0为不限制网络环境。
        //设置接收者
        List<com.igetui.api.openservice.igetui.Target> targetList = new List<com.igetui.api.openservice.igetui.Target>();
        com.igetui.api.openservice.igetui.Target target1 = new com.igetui.api.openservice.igetui.Target();
        target1.appId = APPID;
        target1.clientId = CLIENTID1;
        //target1.alias = ALIAS1;

        // 如需要，可以设置多个接收者
        com.igetui.api.openservice.igetui.Target target2 = new com.igetui.api.openservice.igetui.Target();
        target2.appId = APPID;
        target2.clientId = CLIENTID2;
        //target2.alias = ALIAS2;

        targetList.Add(target1);
        targetList.Add(target2);

        String contentId = push.getContentId(message);
```

```
        String pushResult = push.pushMessageToList(contentId, targetList);
        System.Console.WriteLine("----------------------------------------
-----");
        System.Console.WriteLine("服务端返回结果:" + pushResult);
    }


    //通知透传模板动作内容
    public static NotificationTemplate NotificationTemplateDemo()
    {
        NotificationTemplate template = new NotificationTemplate();
        template.AppId = APPID;
        template.AppKey = APPKEY;
        //通知栏标题
        template.Title = "请填写通知标题";
        //通知栏内容
        template.Text = "请填写通知内容";
        //通知栏显示本地图片
        template.Logo = "";
        //通知栏显示网络图标
        template.LogoURL = "";
        //应用启动类型，1：强制应用启动 2：等待应用启动
        template.TransmissionType = 1;
        //透传内容
        template.TransmissionContent = "请填写透传内容";
        //接收到消息是否响铃，true：响铃 false：不响铃
        template.IsRing = true;
        //接收到消息是否震动，true：震动 false：不震动
        template.IsVibrate = true;
        //接收到消息是否可清除，true：可清除 false：不可清除
        template.IsClearable = true;
        //设置通知定时展示时间，结束时间与开始时间相差需大于6分钟，消
息推送后，客户端将在指定时间差内展示消息（误差6分钟）
        String begin = "2015-03-06 14:36:10";
        String end = "2015-03-06 14:46:20";
        template.setDuration(begin, end);

        return template;
    }
  }
}
```

2.3 返回值

字段	返回码

result	请查询返回值
客户端展示	

注：此接口有频次控制，申请修改请联系邮箱：kegf@getui.com 。

3. 对指定应用群推送消息

3.1 接口说明

接口名称	支持推送类型	说明
pushMessageToApp	透传（payload）、点击通知启动应用、点击通知打开网页等	（通过应用AppID）群推，给所有符合条件的客户端用户推送

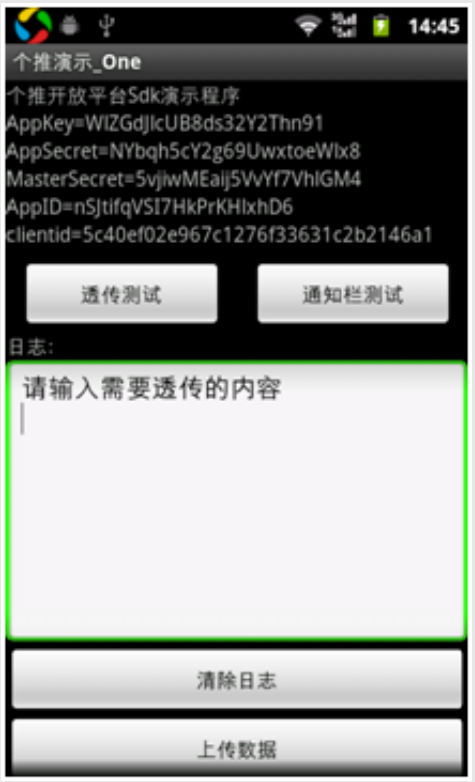
3.2 pushMessageToApp代码实例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

namespace GetuiServerApiSDKDemo
{
   public class demo {

      //采用"C# SDK 快速入门"， "第二步 获取访问凭证 "中获得的应用配置,用户可自行替换
      private static String APPID = "";
      private static String APPKEY = "";
      private static String MASTERSECRET = "";
       //您获取的clientID
      private static String CLIENTID = "";
      //HOST：OpenService接口地址
      private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";

      static void Main(string[] args){
         //3.pushMessageToApp接口
         pushMessageToApp();
      }
      //pushMessageToApp接口测试代码
      private static void pushMessageToApp(){
         // 推送主类（方式1，不可与方式2共存）
         IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
         // 推送主类（方式2，不可与方式1共存）此方式可通过获取服务端地址列表判断最快域名后进行消息推送，每10分钟检查一次最快域名
         //IGtPush push = new IGtPush("",APPKEY,MASTERSECRET);
```

```
        AppMessage message = new AppMessage();

        // 设置群推接口的推送速度，单位为条/秒，仅对
pushMessageToApp（对指定应用群推接口）有效
        message.Speed = 100;

        TransmissionTemplate template =  TransmissionTemplateDemo();

        // 用户当前不在线时，是否离线存储,可选
        message.IsOffline = false;
        // 离线有效时间，单位为毫秒，可选
        message.OfflineExpireTime = 1000 * 3600 * 12;
        message.Data = template;
        //message.PushNetWorkType = 0;      //判断是否客户端是否wifi环境
下推送，1为在WIFI环境下，0为不限制网络环境。
        List<String> appIdList = new List<string>();
        appIdList.Add(APPID);

        //通知接收者的手机操作系统类型
        List<String> phoneTypeList = new List<string>();
            phoneTypeList.Add("ANDROID");
            phoneTypeList.Add("IOS");
        //通知接收者所在省份
        List<String> provinceList = new List<string>();
            provinceList.Add("浙江");
            provinceList.Add("上海");
            provinceList.Add("北京");

        List<String> tagList = new List<string>();
            tagList.Add("开心");

        message.AppIdList = appIdList;
        message.PhoneTypeList = phoneTypeList;
        message.ProvinceList = provinceList;
        message.TagList = tagList;



        String pushResult = push.pushMessageToApp(message);
        System.Console.WriteLine("----------------------------------------
-----");
        System.Console.WriteLine("服务端返回结果：" + pushResult);
    }
    //透传模板动作内容
    public  static TransmissionTemplate TransmissionTemplateDemo()
    {
        TransmissionTemplate template = new TransmissionTemplate();
```

```
        template.AppId = APPID;
        template.AppKey = APPKEY;
        //应用启动类型，1：强制应用启动 2：等待应用启动
        template.TransmissionType = 1;
        //透传内容
        template.TransmissionContent = "透传内容";
        //设置通知定时展示时间，结束时间与开始时间相差需大于6分钟，消息推送后，客户端将在指定时间差内展示消息（误差6分钟）
        String begin = "2015-03-06 14:36:10";
        String end = "2015-03-06 14:46:20";
        template.setDuration(begin, end);

        return template;
    }
  }
}
```

3.3 返回值

字段	返回码
result	请查询返回值
客户端展示	

注：此接口有频次控制，申请修改请联系邮箱：kegf@getui.com 。

4. 任务组名推送

4.1 描述

一个应用同时下发了n个推送任务，为了更好地跟踪这n个任务的推送效果，可以把他们组成一个任务组，在查询数据时，只需要输入该任务组名即可同时查到n个任务的数据结果。

4.2 应用场景

- 场景：做AB test，分别下发A组、B组推送任务，将A、B任务建成“任务组1”，查数据时，仅需要查找任务组1，即可以一起看到A、B两组测试的结果，可以更直观地对比数据。

4.3 对应接口

命名同一个应用的不同taskid为同一个任务组名，任务组名由第三方填写。tolist（对指定用户列表推送消息）、toapp（对指定应用群推消息）接口支持该功能。

4.3.1 Tolist接口代码实例

```
public void toListGroupName(string host, string appkey, string
mastersecret, string taskGroupName, ListMessage msg)
{
   IGtPush push = new IGtPush(host, appkey, mastersecret);
   string contentId = push.getContentId(msg, taskGroupName);
}
```

4.3.2 Toapp接口代码实例

```
public void toAppGroupName(string host, string appkey, string
mastersecret, string taskGroupName, AppMessage msg)
{
   IGtPush push = new IGtPush(host, appkey, mastersecret);
   push.pushMessageToApp(msg, taskGroupName);
}
```

5. 批量单推功能

5.1 描述

用于一次创建提交多个单推任务。

5.2应用场景

当单推任务较多时，推荐使用该接口，可以减少与服务端的交互次数。

5.3 对应接口

函数说明：

接口定义	说明
add(***message***,target)	追加单推消息
submit()	提交消息
retry()	重新提交

参数说明：

推送参数message和target与对单个用户推送消息使用的的参数相同

推送返回值：

批量单推每个单推消息的返回结果在submit返回值的info字段中，具体为加入时的序号和对应的返回结果。返回值详情请点击Result返回值

代码实例：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

public static void singleBatchDemo()
{
   IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
   IBatch batch = new BatchImpl(APPKEY, push);
```

```
        //消息模版：TransmissionTemplate:透传模板
    TransmissionTemplate templateTrans = TransmissionTemplateDemo();

    // 单推消息模型
    SingleMessage messageTrans = new SingleMessage();
    messageTrans.IsOffline = true;                    // 用户当前不在线时，是否离线存储,可选
    messageTrans.OfflineExpireTime = 1000 * 3600 * 12;          // 离线有效时间，单位为毫秒，可选
    messageTrans.Data = templateTrans;
    //判断是否客户端是否wifi环境下推送，2为4G/3G/2G，1为在WIFI环境下，0为不限制环境
    //messageTrans.PushNetWorkType = 1;

    com.igetui.api.openservice.igetui.Target targetTrans = new com.igetui.api.openservice.igetui.Target();
    targetTrans.appId = APPID;
    targetTrans.clientId = CLIENTID1;
    batch.add(messageTrans, targetTrans);

     NotificationTemplate templateNoti = NotificationTemplateDemo();

    // 单推消息模型
    SingleMessage messageNoti = new SingleMessage();
    messageNoti.IsOffline = true;                    // 用户当前不在线时，是否离线存储,可选
    messageNoti.OfflineExpireTime = 1000 * 3600 * 12;          // 离线有效时间，单位为毫秒，可选
    messageNoti.Data = templateNoti;
    //判断是否客户端是否wifi环境下推送，2为4G/3G/2G，1为在WIFI环境下，0为不限制环境
    //messageNoti.PushNetWorkType = 1;

    com.igetui.api.openservice.igetui.Target targetNoti = new com.igetui.api.openservice.igetui.Target();
    targetNoti.appId = APPID;
    targetNoti.clientId = CLIENTID2;
    batch.add(messageNoti, targetNoti);
    try
    {
        batch.submit();
    } catch(Exception e)
    {
        batch.retry();
    }
}
//透传模板动作内容
```

```
public static TransmissionTemplate TransmissionTemplateDemo()
{
   TransmissionTemplate template = new TransmissionTemplate();
   template.AppId = APPID;
   template.AppKey = APPKEY;
   //应用启动类型，1：强制应用启动 2：等待应用启动
   template.TransmissionType = 1;
   //透传内容
   template.TransmissionContent = "透传内容";
   //设置通知定时展示时间，结束时间与开始时间相差需大于6分钟，消息推送后，客户端将在指定时间差内展示消息（误差6分钟）
   String begin = "2016-11-10 18:00:00";
   String end = "2016-11-10 22:46:20";
   template.setDuration(begin, end);

   return template;
}

//通知透传模板动作内容
public static NotificationTemplate NotificationTemplateDemo()
{
   NotificationTemplate template = new NotificationTemplate();
   template.AppId = APPID;
   template.AppKey = APPKEY;
   //通知栏标题
   template.Title = "请填写通知标题";
   //通知栏内容
   template.Text = "请填写通知内容";
   //通知栏显示本地图片
   template.Logo = "";
   //通知栏显示网络图标
   template.LogoURL = "";
   //应用启动类型，1：强制应用启动 2：等待应用启动
   template.TransmissionType = 1;
   //透传内容
   template.TransmissionContent = "请填写透传内容";
   //接收到消息是否响铃，true：响铃 false：不响铃
   template.IsRing = true;
   //接收到消息是否震动，true：震动 false：不震动
   template.IsVibrate = true;
   //接收到消息是否可清除，true：可清除 false：不可清除
   template.IsClearable = true;
   //设置通知定时展示时间，结束时间与开始时间相差需大于6分钟，消息推送后，客户端将在指定时间差内展示消息（误差6分钟）
   String begin = "2015-03-06 14:36:10";
   String end = "2015-03-06 14:46:20";
   template.setDuration(begin, end);
```

```
    return template;
}
```

6. PushResult返回值查询

具体返回值请查询下表

正确返回	返回码	结果说明
	successed_online	用户在线，消息在线下发
	successed_offline	用户离线，消息存入离线系统
	Ok	发送成功
	details	返回用户状态的详细信息
	contentId	任务ID（当result值为ok时，有此字段）

错误返回	返回码	结果说明
	Error	请求信息填写有误
	action_error	未找到对应的action动作
	appkey_error	Appkey填写错误
	domain_error	填写的域名错误或者无法解析
	sign_error	Appkey与ClientId不匹配，鉴权失败
	AppidNoMatchAppKey	appid和鉴权的appkey不匹配
	PushMsgToListOrAppTimesOverLimit	群推次数超过最大值
	PushTotalNumOverLimit	推送个数总数超过最大值
	AppIdNoUsers	该AppId下的用户总数为0
	SendError	消息推送发送错误
	SynSendError	报文发送错误
	flow_exceeded	接口消息推送流量已超限
	TargetListIsNullOrSizeIs0	推送target列表为空
	PushTotalNumOverLimit	推送消息个数总数超限
	TokenMD5NoUsers	target列表没有有效的clientID
	NullMsgCommon	未找到contentId对应的任务

	TaskIdHasBeanCanceled	任务已经被取消
	AppidError	clientid绑定的appid与推送的appid不符
	successed_ignore	无效用户，消息丢弃
	TokenMD5Error	clientID填写有误
	SendError	消息发送错误
	AppidNoAppSecret	appid未找到对应的appSecret
	OtherError	未知错误，无法判定错误类型

7. 定时任务推送

7.1 定时对指定应用群推消息

描述

对单个指定应用的所有用户群发推送消息。该消息可以在用户设定的时间点进行推送。

注：此接口需要申请开通，申请邮箱：kegf@getui.com 。

推送接口

代码示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
```

```
private static String MASTERSECRET = "";
private static void pushMessageToApp()
    {
        IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);

        AppMessage message = new AppMessage();
        //TransmissionTemplate template = TransmissionTemplateDemo();
        NotificationTemplate template = NotificationTemplateDemo();
        message.IsOffline = false;
        message.OfflineExpireTime = 1000 * 3600 * 12;
        message.Data = template;
        List<String> appIdList = new List<string>();
        appIdList.Add(APPID);
        List<String> phoneTypeList = new List<string>();
        phoneTypeList.Add("ANDROID");
        phoneTypeList.Add("IOS");
        List<String> provinceList = new List<string>();
        provinceList.Add("浙江");
        provinceList.Add("上海");
        List<String> tagList = new List<string>();
        tagList.Add("lala");
        tagList.Add("la");
        message.AppIdList = appIdList;
        AppConditions cdt = new AppConditions();
        cdt.addCondition(AppConditions.PHONE_TYPE,
phoneTypeList,AppConditions.OptType.or);
        cdt.addCondition(AppConditions.REGION,
provinceList,AppConditions.OptType.or);
        cdt.addCondition(AppConditions.TAG,tagList,
AppConditions.OptType.not);
        message.Conditions =cdt;

        message.PushTime = "201812041010";

        String pushResult = push.pushMessageToApp(message);
        System.Console.WriteLine("----------------------------------------
-----");
        System.Console.WriteLine(pushResult);
    }
public static NotificationTemplate NotificationTemplateDemo()
    {
        NotificationTemplate template = new NotificationTemplate();
        template.AppId = APPID;
        template.AppKey = APPKEY;
        template.Title = "";
        template.Text = "";
        template.Logo = "";
```

```
        template.LogoURL = "";
        template.TransmissionType = 1;
        template.TransmissionContent = "请填写透传内容";
        template.IsRing = true;
        template.IsVibrate = true;
        template.IsClearable = true;

        Style1 style1 = new Style1();
        style1.Title = "style1";
        style1.Text = "style1";
        style1.Logo = "push.png";
        style1.LogoUrl = "";

        template.setStyle(style1);

        Console.WriteLine(template.getTransparent());
        return template;
    }
```

接口部分参数详细说明

- 设定推送的时间格式为yyyyMMddHHmm 例如:201710251900,任务将会在2017年10月25日17点00分推送。
- 对时间的设定有一定的要求:
    - 时间格式不正确 提交任务时 将直接返回失败。
    - 下发时间小于当前时间 提交任务时将直接返回失败。
    - 下发时间超过系统所允许的最大时间点 提交任务 将直接返回失败

7.2 定时任务查询接口

描述

该接口主要用来返回 已提交的定时任务的相关信息。

对应接口

```
public String getScheduleTask(String taskId, String appId)
```

参数说明

参数名	类型	必需	默认值	参数描述
taskId	String	是	无	任务ID
appId	String	是	无	应用ID

代码示例

```
private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String TASKID = "";
private static void getScheduleTaskDemo()
    {
      IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
      string res = push.getScheduleTask(TASKID, APPID);
      Console.WriteLine("----------------------------------------------");
      Console.WriteLine(res);
    }
```

返回的主要参数如下:

参数名称	参数含义
pushContent	推送类容（transmission的内容）
pushTime	推送时间
creatTime	任务创建时间
sendResult	任务状态

7.3 定时任务删除接口

描述

用来删除还未下发的任务

对应接口

```
public String delScheduleTask(String taskId, String appId)
```

参数说明

参数名	类型	必需	默认值	参数描述
taskId	String	是	无	任务ID
appId	String	是	无	应用ID

代码示例

```
private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String TASKID = "";
private static void delScheduleTaskDemo()
    {
      IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
      string res = push.delScheduleTask(TASKID, APPID);
      Console.WriteLine("------------------------------------------------");
      Console.WriteLine(res);
    }
```

需要补充说明的是:

- 距离下发还有一分钟的任务 将无法删除 也即 停止任务下发。

8. 应用群推条件交并补功能

8.1 描述

应用群推对于复杂的查询条件新增加的交并补功能，以对应查询语义中的与或非的关系

8.2 应用场景

- 场景：需要发送给城市在A,B,C里面，没有设置tagtest标签，手机型号为android的用户，用条件交并补功能可以实现，city(A|B|C) && !tag(tagtest) && phonetype(andriod)

8.3 对应接口

```
// AppConditions类的实例方法
public AppConditions addCondition(String key, List<String> values，int optType)
```

参数说明：

参数名	类型	必需	默认值	参数描述
key	str	是	无	查询条件键(phoneType 手机类型,region 省市,tag 用户标签)
values	list	是	无	查询条件值列表
optType	int	否	0	条件类型(OptType.or 或, OptType.and 与, OptType.not 非)

8.4 代码实例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static void pushMessageToApp()
    {
      IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);

      AppMessage message = new AppMessage();

      NotificationTemplate template = NotificationTemplateDemo();

      message.IsOffline = false;
```

```
        message.OfflineExpireTime = 1000 * 3600 * 12;
        message.Data = template;
        List<String> appIdList = new List<string>();
        appIdList.Add(APPID);
        List<String> phoneTypeList = new List<string>();
        phoneTypeList.Add("ANDROID");
        phoneTypeList.Add("IOS");
        List<String> provinceList = new List<string>();
        provinceList.Add("浙江");
        provinceList.Add("上海");
        message.AppIdList = appIdList;

        AppConditions cdt = new AppConditions();
        cdt.addCondition(AppConditions.PHONE_TYPE,
phoneTypeList,AppConditions.OptType.or);
        cdt.addCondition(AppConditions.REGION,
provinceList,AppConditions.OptType.or);
        message.Conditions =cdt;
        String pushResult = push.pushMessageToApp(message);
    }
public static NotificationTemplate NotificationTemplateDemo()
    {
        NotificationTemplate template = new NotificationTemplate();
        template.AppId = APPID;
        template.AppKey = APPKEY;
        template.Title = "";
        template.Text = "";
        template.Logo = "";
        template.LogoURL = "";
        template.TransmissionType = 1;
        template.TransmissionContent = "";
        template.IsRing = true;
        template.IsVibrate = true;
        template.IsClearable = true;


        Style1 style1 = new Style1();
        style1.Title = "style1";
        style1.Text = "style1";
        style1.Logo = "push.png";
        style1.LogoUrl = "";

        template.setStyle(style1);

        Console.WriteLine(template.getTransparent());
        return template;
    }
```

10. 短信补量推送接口

10.1 说明

针对推送，在消息有效时间内，对未收到推送消息的用户进行短信补发，既提升消息触达又节省成本

10.2 接入流程

10.2.1 获取应用参数

进入个推开发者中心后，个推消息推送模块,注册app,获取 appId、appKey 和 masterSecret 参数（妥善保管）。

10.2.2 模板报备

进行短信补量推送前，必须在个推申请开通短信补量并报备短信模板。短信模板审核通过后，才能进行短信补量推送。短信模板报备审核的日期为1-2个工作日。

10.2.3 模板示例

模板ID	模板内容	相关参数解释
10000	示例【***】尊敬的 ***name***，你的余额还剩{money}，请尽快充值。	变量格式:${name}。

10.2.4 pn绑定

进行短信补量推送前，需要调用接口绑定cid与pn。推送时获取pn，进行消息下发。其中文档中所说的pn 都是指md5之后的手机号。

10.3 短信补量补发逻辑

个推推送消息下发时，对于在线用户，直接通过sdk通道下发，对于离线用户消息将存储在离线空间，等到用户设定的短信补发时间，将开始对已绑定pn的用户进行短信补发。 对于在补发前，经过个推通道成功下发消息的用户，将不再进行短信补发。

10.4 cid与pn绑定接口

10.4.1 接口描述

绑定cid和pn的关系，用户短信补量根据cid查询pn下发短信。

10.4.2 功能代码示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String PN = "";

private static void bindCidPnDemo()
    {
        Encoding encoding = new UTF8Encoding(true);
        IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
        Dictionary<string, String> dict = new Dictionary<string, String>();
        System.Security.Cryptography.MD5CryptoServiceProvider
md5Hasher = new
System.Security.Cryptography.MD5CryptoServiceProvider();
        byte[] data = md5Hasher.ComputeHash(encoding.GetBytes(PN));
        StringBuilder sBuilder = new StringBuilder();
        for (int i = 0; i < data.Length; i++)
        {
            sBuilder.Append(data[i].ToString("x2"));
        }
```

```
        dict.Add(CLIENTID, sBuilder.ToString());
        string res = push.bindCidPn(APPID, dict);
        Console.WriteLine(res);
    }
```

10.4.3 具体参数

参数	类型	是否必填	最大长度	描述
appId	String	是	32	个推分配给开发者的ID
cidAndPn	Dict	是	50	key是cid value是pnmd5【32位小写】

10.4.4 返回值说明

参数	类型	是否必填	描述	示例值	其他
result	String	是	响应码	0	详细code码含义。参加文档最后【附录一】
batchRet	String	否	返回每个cid的绑定结果	[{"code":"0","cid":"xxxx"}]	

10.4.5 注意事项

1）目前接口最多支持一次绑定50个cid与pn的对应关系。绑定后会返回每个cid的绑定结果。

2）同个应用下pn和cid是一对一的关系，否则原关系会被替换

10.5 cid与pn解绑接口

10.5.1 接口描述

用户通过接口传递cid列表, 可以批量解绑与之相对应的pn的关系。

10.5.2 功能代码示例

```
private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String CLIENTID = "";
private static void unbindCidPnDemo()
    {
      IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
      List<string> list = new List<string>();
      list.Add(CLIENTID);
      string res = push.unbindCidPn(APPID, list);
      Console.WriteLine("----------------------------------------------");
      Console.WriteLine(res);
    }
```

10.5.3 具体参数

参数	类型	是否必填	最大长度	描述
appId	String	是	32	开发者ID
cids	list	是	50	客户端身份ID列表

10.5.4 返回值说明

参数	类型	是否必填	描述	示例值	其他
result	String	是	响应码	0	详细code码含义。参加文档最后【附录一】
batchRet	String	否	返回每个cid的绑定结果	[{"code":"0","cid":"xxxx"}]	

10.5.5 注意事项

1）目前接口最多支持一次解绑50个cid

10.6 pn查询接口

10.6.1 接口描述

用户通过接口传递cid列表, 可以查询与之相对的pn,接口会批量返回绑定关系。

10.6.2 用法举例

```
private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String CLIENTID = "";
private static void queryCidPnDemo()
    {
      IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
      List<string> cidList = new List<string>();
      cidList.Add(CLIENTID);
      string res = push.queryCidPn(APPID, cidList);
      Console.WriteLine("----------------------------------------------");
      Console.WriteLine(res);
    }
```

10.6.3 具体参数

参数	类型	是否必填	最大长度	描述
appId	String	是	32	开发者ID
cidList	list	是	50	客户端身份ID列表

10.6.4 返回值说明

参数	类型	是否必填	描述	示例值	其他
result	String	是	响应码	0	详细code码含义。参加文档最后【附录一】
batchRet	String	否	返回每个cid的绑定结果	[{"code":"0","cid":"xxxx"}]	

10.6.5 注意事项

1）目前接口最多支持一次解绑50个cid

10.7.1 接口列表

接口定义	说明
pushMessageToApp	对应用的所有用户群发推送消息
pushMessageToSingle	向单个clientid或别名用户推送消息
pushMessageToList	上传clientid或别名列表，对列表中所有clientid或别名用户进行消息推送
注：详细的推送逻辑参考个推官网。这里只是补充短信推送的逻辑 参考地址：http://docs.getui.com/getui/server/php/push//	

10.7.2 支持短信模板的个推消息模板

消息模板	中文含义	是否支持短信	其他

	模板		
Notification	通知模板	是	消息模板具体含义参考：http://docs.getui.com/getui/server/php/template/
LinkTemplate	网页模板	是	
NotyPopLoadTemplate	下载模板	是	
TransmissionTemplate	透传模板	是	

10.7.3 短信推送模板参数

成员名或方法名	类型	是否必填	描述	示例值	其他
smsTemplateId	String	是	推送的短信模板ID（短信模板说明）	smsMessage.smsTemplateId = "xxx"	具体使用，参考下面的代码中LinkTemplate中使用短信模板的示例。其他模板如此相同
smsContent	map	否	推送的短信模板中占位符的内容。	smsContent = {} smsContent["name"] = "zhangsan" smsContent["money"] = "0000" smsMessage.smsContent = smsContent	
offlineSendtime	long	是	推送后多久进行短信补发（单位：ms）	smsMessage.offlineSendtime = 1000	

isApplink	boolean	否	推送的短信模板中是否选用APPLink进行推送。	smsMessage.isApplink = True
url	String	否	推送的短信模板中的APPLink链接地址。	smsMessage.url = "www.baidu.com"
payload	String	否	推送的短信模板中的APPLink自定义字段。	smsMessage.payload = "自定义"

10.7.4 pushMessageToApp 完整示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String CLIENTID = "";
private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";

      private static void pushMessageToApp(){
         IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);

         AppMessage message = new AppMessage();
```

```
        // 设置群推接口的推送速度，单位为条/秒，仅对
pushMessageToApp（对指定应用群推接口）有效
        message.Speed = 100;

        TransmissionTemplate template =  TransmissionTemplateDemo();
com.igetui.api.openservice.igetui.sms.SmsInfo smsMessage = new
com.igetui.api.openservice.igetui.sms.SmsInfo();
        Dictionary<string, string> smscount = new Dictionary<string,
string>();
        smscount.Add("name", "");
        smscount.Add("code", "");
        smscount.Add("hour","");

        smsMessage.Payload = "";
        smsMessage.SmsContent = smscount;
        smsMessage.SmsTemplateId = "";
        smsMessage.IsApplink = false;
        smsMessage.Url = "www.bai";
        smsMessage.OfflineSendtime = 1;
        template.setSmsInfo(smsMessage);
        message.IsOffline = false;
        message.OfflineExpireTime = 1000 * 3600 * 12;
        message.Data = template;
        List<String> appIdList = new List<string>();
        appIdList.Add(APPID);

        List<String> phoneTypeList = new List<string>();
            phoneTypeList.Add("");
            phoneTypeList.Add("");
        List<String> provinceList = new List<string>();
            provinceList.Add("浙江");
            provinceList.Add("上海");
            provinceList.Add("北京");

        List<String> tagList = new List<string>();
            tagList.Add("开心");

        message.AppIdList = appIdList;
        message.PhoneTypeList = phoneTypeList;
        message.ProvinceList = provinceList;
        message.TagList = tagList;



        String pushResult = push.pushMessageToApp(message);
        System.Console.WriteLine("----------------------------------------
-----");
        System.Console.WriteLine("服务端返回结果：" + pushResult);
```

```
    }
    //透传模板动作内容
    public  static TransmissionTemplate TransmissionTemplateDemo()
    {
      TransmissionTemplate template = new TransmissionTemplate();
      template.AppId = APPID;
      template.AppKey = APPKEY;
      template.TransmissionType = 1;
      //透传内容
      template.TransmissionContent = "透传内容";
      String begin = "2015-03-06 14:36:10";
      String end = "2015-03-06 14:46:20";
      template.setDuration(begin, end);

      return template;
    }
}
```

10.7.5 pushMessageToSingle 完整示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String CLIENTID = "";
private static void pushMessageToSingleToSMS()
     {
       IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);
       TransmissionTemplate template = TransmissionTemplateDemo();
       com.igetui.api.openservice.igetui.sms.SmsInfo smsMessage = new
com.igetui.api.openservice.igetui.sms.SmsInfo();
       Dictionary<string, string> smscount = new Dictionary<string,
string>();
```

```
            smscount.Add("name", "");
            smscount.Add("code", "");
            smscount.Add("hour","");

            smsMessage.Payload = "";
            smsMessage.SmsContent = smscount;
            smsMessage.SmsTemplateId = "";
            smsMessage.IsApplink = false;
            smsMessage.Url = "www.bai";
            smsMessage.OfflineSendtime = 1;
            template.setSmsInfo(smsMessage);
            SingleMessage message = new SingleMessage();
            message.IsOffline = true;
            // 离线有效时间
            message.OfflineExpireTime = 1000 * 3600 * 12;
            message.Data = template;
            com.igetui.api.openservice.igetui.Target target = new
com.igetui.api.openservice.igetui.Target();
            target.appId = APPID;
            target.clientId = CLIENTID;
            string ret = push.pushMessageToSingle(message, target);
            Console.WriteLine("-----------------------------------------------");
            Console.WriteLine(ret + "pushsms");
        }
  public  static TransmissionTemplate TransmissionTemplateDemo()
        {
            TransmissionTemplate template = new TransmissionTemplate();
            template.AppId = APPID;
            template.AppKey = APPKEY;
            template.TransmissionType = 1;
            //透传内容
            template.TransmissionContent = "透传内容";
            String begin = "2015-03-06 14:36:10";
            String end = "2015-03-06 14:46:20";
            template.setDuration(begin, end);

            return template;
        }
```

10.7.6 pushMessageToList 完整示例

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

```
using System.Text;
using Google.ProtocolBuffers;
using com.gexin.rp.sdk.dto;
using com.igetui.api.openservice;
using com.igetui.api.openservice.igetui;
using com.igetui.api.openservice.igetui.template;
using System.Net;

private static String HOST = "http://sdk.open.api.igexin.com/apiex.htm";
private static String APPID = "";
private static String APPKEY = "";
private static String MASTERSECRET = "";
private static String CLIENTID = "";
private static String CLIENTID1 = "";
private static String CLIENTID2 = "";
 private static void PushMessageToList()
      {
        // 推送主类（方式1，不可与方式2共存）
        IGtPush push = new IGtPush(HOST, APPKEY, MASTERSECRET);

        ListMessage message = new ListMessage();

        NotificationTemplate template = TransmissionTemplateDemo();
        // 用户当前不在线时，是否离线存储,可选
        message.IsOffline = true;
        // 离线有效时间，单位为毫秒，可选
        message.OfflineExpireTime = 1000 * 3600 * 12;
        message.Data = template;
 com.igetui.api.openservice.igetui.sms.SmsInfo smsMessage = new
com.igetui.api.openservice.igetui.sms.SmsInfo();
        Dictionary<string, string> smscount = new Dictionary<string,
string>();
        smscount.Add("name", "");
        smscount.Add("code", "");
        smscount.Add("hour","");

        smsMessage.Payload = "1234";
        smsMessage.SmsContent = smscount;
        smsMessage.SmsTemplateId = "";
        smsMessage.IsApplink = false;
        smsMessage.Url = "";
        smsMessage.OfflineSendtime = 1;
        template.setSmsInfo(smsMessage);
        List<com.igetui.api.openservice.igetui.Target> targetList = new
List<com.igetui.api.openservice.igetui.Target>();
        com.igetui.api.openservice.igetui.Target target1 = new
com.igetui.api.openservice.igetui.Target();
```

```
            target1.appId = APPID;
            target1.clientId = CLIENTID;

            com.igetui.api.openservice.igetui.Target target2 = new
com.igetui.api.openservice.igetui.Target();
            target2.appId = APPID;
            target2.clientId = CLIENTID1;

            com.igetui.api.openservice.igetui.Target target3 = new
com.igetui.api.openservice.igetui.Target();
            target3.appId = APPID;
            target3.clientId = CLIENTID2;

            targetList.Add(target1);
            targetList.Add(target2);
            targetList.Add(target3);

            String contentId = push.getContentId(message);
            String pushResult = push.pushMessageToList(contentId, targetList);
            System.Console.WriteLine("----------------------------------------
-----");
            System.Console.WriteLine(pushResult);
        }

    public  static TransmissionTemplate TransmissionTemplateDemo()
        {
            TransmissionTemplate template = new TransmissionTemplate();
            template.AppId = APPID;
            template.AppKey = APPKEY;
            template.TransmissionType = 1;
            //透传内容
            template.TransmissionContent = "透传内容";
            String begin = "2015-03-06 14:36:10";
            String end = "2015-03-06 14:46:20";
            template.setDuration(begin, end);

            return template;
        }
```

10.7.7 返回值说明

参数	类型	是否必填	描述	示例值
result	String	是	响应码	result=ok

contentId	String	否	任务ID	contentId=xxxxxxx
smsResult	String	否	短信提交结果	smsResult=accept
smsResultMsg	String	否	短信提交失败结果	smsResultMsg=模板不存在

10.7.8 推送返回的完整结果示例

结果	示例	说明
短信补量提交成功	{result=ok, contentId=xxxxx, smsResult=accept}	消息推送成功，短信提交成功。
短信补量提交失败	{result=ok, smsResultMsg=模板不存在, contentId=xxxxxxx, smsResult=error}	消息推送成功，短信提交失败。

10.7.9 注意事项

1）短信参数填错，将不会进行短信补量，但是通过个推的sdk推送的消息会正常下发

2）短信补量的开始时间必须在个推通道的离线时间内，且不大于服务端限制值（目前设置是3天）。

3）短信补发的整个内容的总长度不能超过70字。

10.8 pn与cid操作的相关状态码

状态码	含义
0	成功
1	cid不存在
2	cid与appid不匹配
3	不能覆写低级别的PN绑定
4	应用下该PN已经绑定更活跃的cid
5	cid为空
6	pn为空
7	该cid没有绑定PN值
8	其他原因导致的失败

11. 推送结果返回值

具体返回值请查询下表

正确返回	返回码	结果说明
	successed_online	用户在线，消息在线下发
	successed_offline	用户离线，消息存入离线系统
	Ok	发送成功
	details	返回用户状态的详细信息
	contentId	任务ID（当result值为ok时，有此字段）

错误返回	返回码	结果说明
	Error	请求信息填写有误
	action_error	未找到对应的action动作
	appkey_error	Appkey填写错误
	domain_error	填写的域名错误或者无法解析
	sign_error	Appkey与ClientId不匹配，鉴权失败
	AppidNoMatchAppKey	appid和鉴权的appkey不匹配
	PushMsgToListOrAppTimesOverLimit	群推次数超过最大值
	PushTotalNumOverLimit	推送个数总数超过最大值
	AppIdNoUsers	该AppId下的用户总数为0
	SendError	消息推送发送错误
	SynSendError	报文发送错误
	flow_exceeded	接口消息推送流量已超限
	TargetListIsNullOrSizeIs0	推送target列表为空
	PushTotalNumOverLimit	推送消息个数总数超限
	TokenMD5NoUsers	target列表没有有效的clientID
	NullMsgCommon	未找到contentId对应的任务
	TaskIdHasBeanCanceled	任务已经被取消
	AppidError	clientid绑定的appid与推送的appid不符
	successed_ignore	无效用户，消息丢弃
	TokenMD5Error	clientID填写有误

SendError	消息发送错误
AppidNoAppSecret	appid未找到对应的appSecret
OtherError	未知错误，无法判定错误类型