通过 GSM 发送通知接口说明 V1.1

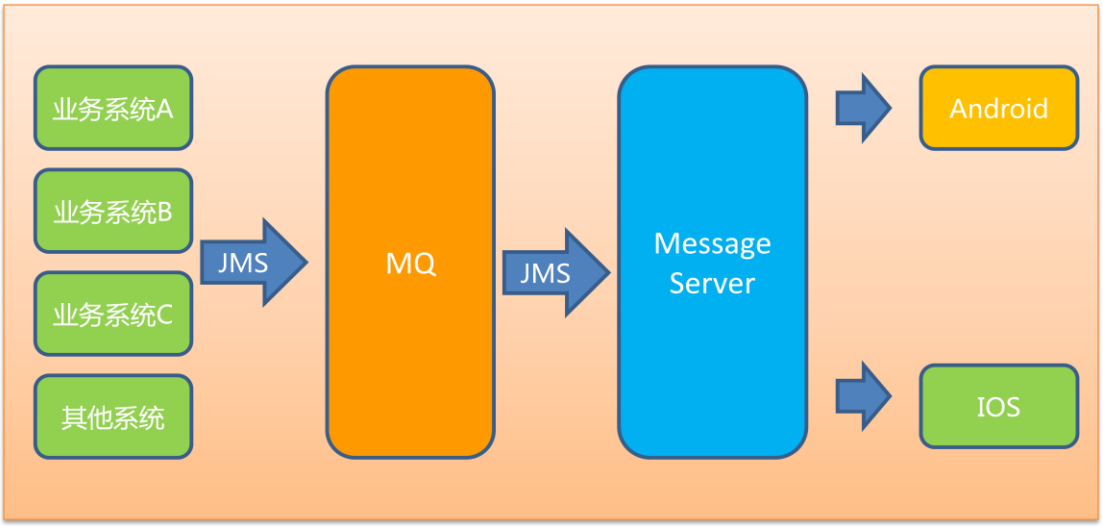
版本	作者	修改记录	备注
1.0	张明绩	2014-10-16	创建
1.1	张明绩	2015-03-20	更新

目录

一、 通知发送说明

因各个系统不维护手机端的数据，各个系统要发送通知到手机端需要经过 GSM，考虑到系统架构及性能问题在 V1.0 版本上进行改进，各个系统发送通知不再经过 GSM，直接通过 M只 MQ 发送通知。

1. JMS 方式



二、 发送参数说明

1. JMS 方式参数说明

参数名称	参数说明	参数类型	备注
companyId	公司 Id	String	通过 Oauth 可以获得
employeeId	员工编号	String	通过 Oauth 可以获得
title	通知标题	String	
content	通知内容	String	
type	通知类型	String	101：会议通知 102：升级通知 103：促销通知 104：专项通知 105：员工通知
sender	发送者	String	工程名称

代码示例：

采用 JMS 与 spring 集成的方式

发送类：

```
package com.gome.gsm.service.system.test;

import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.Queue;
import javax.jms.Session;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.jms.core.MessageCreator;

/**
 *
 * @author Zhang.Mingji
 * @date 2014年10月16日上午11:02:59
 * @Copyright(c) gome inc Gome Co.,LTD
 */
public class NotificationMQSenderTest {

   private static final Logger logger =
LoggerFactory.getLogger(NotificationMQSenderTest.class);

   private JmsTemplate jmstemplate;
   private Queue queue;


   public JmsTemplate getJmstemplate() {
      return jmstemplate;
   }

   public void setJmstemplate(JmsTemplate jmstemplate) {
      this.jmstemplate = jmstemplate;
   }

   public Queue getQueue() {
      return queue;
   }

   public void setQueue(Queue queue) {
```

```
        this.queue = queue;
    }

    public void send(final String message){
        this.jmstemplate.send(queue, new MessageCreator() {

            @Override
            public Message createMessage(Session session)
throws JMSException {
                return session.createTextMessage(message);
            }
        });



        logger.info("-------发送消息到通知队列成功----------");
    }
}
```

调用类:

```
package com.gome.gsm.service.system.test;

import java.util.HashMap;
import java.util.Map;

import com.gome.gsm.util.JsonUtil;

/**
 *
 * @author Zhang.Mingji
 * @date 2014年10月16日上午11:07:50
 * @Copyright(c) gome inc Gome Co.,LTD
 */
public class NotificationMQSenderClient {

    public static void main(String[] args) throws Exception
{

        //开发中需要使用spring或其他容器注入,这里方便测试直接采用new
的方式，仅提供参考
        NotificationMQSenderTest sender = new
NotificationMQSenderTest();
        Map<String,Object> map = new HashMap<String,Object>();
```

```
        map.put("companyId", "10000");
        map.put("employeeId", "100002");

        map.put("title", "通知标题");


        map.put("content", "通知内容");

        map.put("type", "105");
        map.put("sender", "ygzz");
List<Map<String,Object>> list = new
ArrayList<Map<String,Object>>();
list.add(map);
        String message =
JsonUtil.javaObjectToJsonString(list);
        sender.send(message);
    }

}
```

Spring 配置：（MQ 地址需要根据 MQ 服务器进行更改）

```
<bean id="jmsConnectionFactory"
class="org.apache.activemq.ActiveMQConnectionFactory">
    <property name="brokerURL" value="${mq.url}" />
    <property name="useAsyncSend" value="true" />
  </bean>
  <bean id="pooledJmsConnectionFactory"
class="org.apache.activemq.pool.PooledConnectionFactory"
    destroy-method="stop">
    <property name="connectionFactory"
ref="jmsConnectionFactory" />
    <property name="maxConnections"
value="${mq.maxJmsConnections}" />
    <property name="maximumActive"
value="${mq.maxJmsActive}" />
  </bean>
  <bean id="jmsTemplate"
class="org.springframework.jms.core.JmsTemplate">
    <property name="connectionFactory"
ref="pooledJmsConnectionFactory" />
  </bean>

<bean id="sendNotificationQueue"
class="org.apache.activemq.command.ActiveMQQueue">
    <constructor-arg value="gsm.message.push.queue" />
  </bean>
```

```
  <bean id="notificationSender" class="
com.gome.gsm.service.system.test. NotificationMQSenderTest
">
    <property name="jmsTemplate"
ref="jmsTemplate"></property>
    <property name="queue"
ref="sendNotificationQueue"></property>
  </bean>
```

注意：一条消息对应一个人（一个 map）