四、非 android 客户端的使用

Ios , actionscript, c，net,等语言的客户端可以使用 socket 连接服务端，然后发送和接受对应格式的 xml 消息即可，

4.0 消息分隔符 \b

完整的 xml 数据之间用 消息分隔符来分割，

发送一段 xml 时后面需要加一个消息分隔符 \b，收到一个完整消息时后面也会有 \b

4.1 客户端绑定到服务端

参数

字段名称	类型	说明
必选参数		
account	string	用户账号
deviceId	String	设备 ID，ios 则为 deviceToken
channel	string	客户端类型，取值为 android , ios , windows
可选参数		
device	String	设备型号名称
version	String	客户端应用版本号
osVersion	String	系统版本号
packageName	String	包名

```
<?xml version="1.0" encoding="UTF-8"?>
 <sent>
    <key>client_bind</key>
    <data>
        <account>xiyang</account>
```

```
        <channel>android</channel>
        <deviceId>764310254498</deviceId>
        <device>HTC 8088</device>
        <version>1.1.0</version>
        <osVersion>4.4.2</osVersion>

    </data>
    </sent>\b
```

则用 socket 向服务端发送 此 xml 报文即可

注意，发送一条完整的消息 xml 字符串后面 必须加上\b 特殊字符，它将作为一条完整消息的分界线，不能缺少

注意，客户端要设置 socket 的延迟发送为 false

绑定成功后返回如下 xml 说明绑定成功

```
<?xml version="1.0" encoding="UTF-8"?>
 <reply>
    <key>client_bind</key>
    <code>200</code>
</reply>\b
```

4.2 客户端收到消息

接收到的消息报文,如果一次接到多个消息，将会用\b 作为消息分隔界限,当取到\b 时 意味着一条完整的消息结尾

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
<mid>2cd79d3bfe4b43ce8bf7e2b9a38796e1</mid>
<action>2</action>
<title></title>
<content>hello world!</content>
<sender>system</sender>
```

```
<receiver>abc</receiver>
<timstamp>1343243455553</timstamp>
</message>\b
```

action 以及 content 定义参考接口文档 9.0

4.3 客户端心跳

心跳由服务的主动向客户端发起，间隔为 5 分钟一次，用户检查客户端是否还在线，服务的发起心跳请求命令为 S_H_RQ 当客户端收到心跳请求命令字符的时候应立即回应 C_H_RS 字符给服务器，否则服务的将会断开连接，注意回复的时候加上分隔符 参见 4.0

4.5 JAVA socket 客户代码实例

附上 java socket 使用代码实例

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
public class Client {

    static Socket socket;
    public static void main(String[] args) throws UnknownHostException,
IOException {

        socket = new Socket("127.0.0.1",23456);
        socket.setTcpNoDelay(true);
        sendBindRequest();//发送绑定请求

        while(true){
            receiveMessage();//接受消息
        }
    }
```

```
    public static void sendBindRequest() throws IOException
    {
        PrintWriter write = new PrintWriter(new OutputStreamWriter
(socket.getOutputStream()),true);

        StringBuffer buffer = new StringBuffer();
        buffer.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
        buffer.append("<sent>");
        buffer.append("<key>client_bind</key>");
        buffer.append("<data>");
        buffer.append("<account>xiyang</account>");
        buffer.append("<channel>java</channel>");
        buffer.append("<device>JVM</device>");
        buffer.append("</data>");
        buffer.append("</sent>");
        buffer.append('\b');
        write.print(buffer.toString());
        write.flush();

        System.out.println("发送绑定成功:"+buffer.toString());
    }


    public static void receiveMessage() throws IOException
    {

        byte[] buf = new byte[2048];
        int len = 0;
        InputStream input = socket.getInputStream();
        while((len = input.read(buf))!=-1){
            String data  = new String(buf,0,len,"UTF-8");
            System.out.println("收到服务端返回:"+data);
        }

    }
}
```