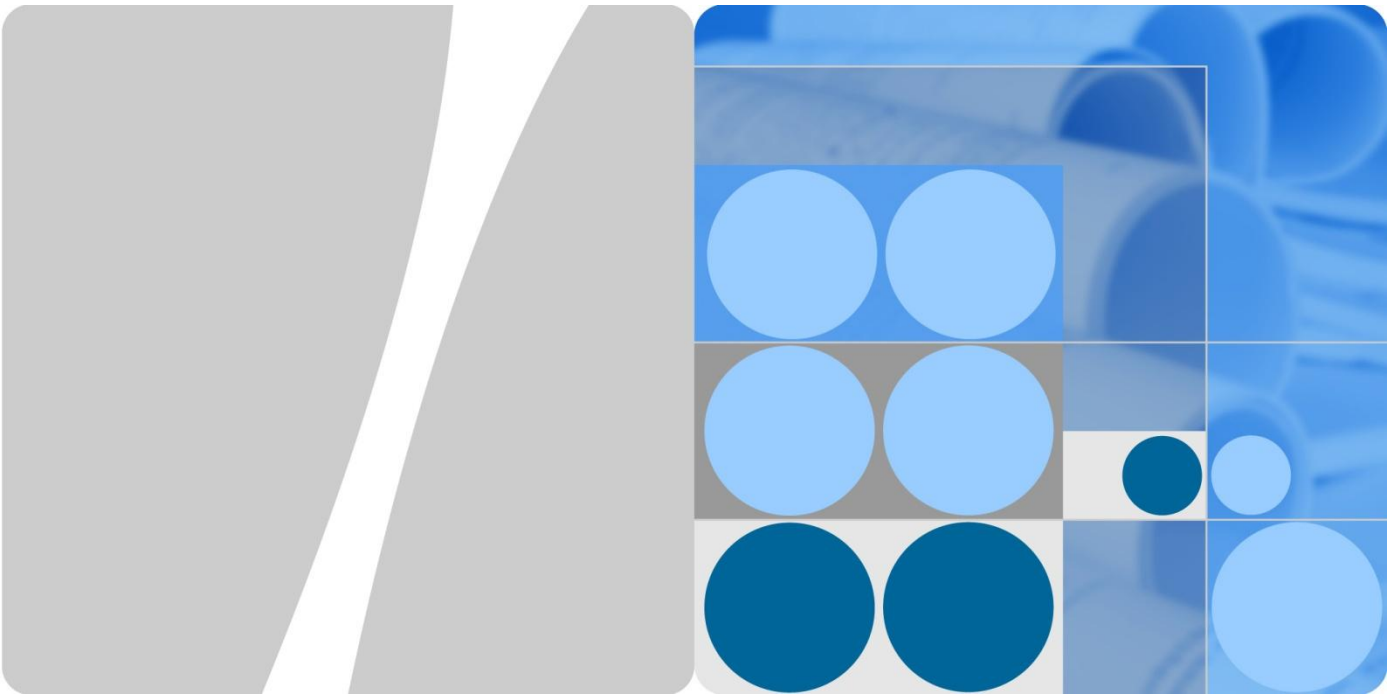


eSpace UC 服务端开放接口描述

注意：所有代码仅供学习参考

文档版本 1.02

发布日期 2012-06-28

华为技术有限公司

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

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

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://enterprise.huawei.com/

客户服务邮箱： uccisv@huawei.com

客户服务电话： 4008229999

前 言

概述

本文档用于 eSpace UC 服务端功能集成开发，本文档所描述接口适用于 eSpace UC V100R002C01 及以后的版本。

读者对象

本文档（本指南）主要适用于以下工程师：

ISV 集成开发工程师

术语表

术语	说明
ISV	Independent Software Vendors（独立软件开发商）。
HTTP	Hyper Text Transport Protocol(超文本传输协议)。一种详细规定了浏览器和万维网服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议。
UC	Unified Communications（统一通信）。
BMU	Business Management Unit（业务管理单元）。eSpace UC 服务器端的网元
URL	Uniform Resource Locator（统一资源定位符）。也被称为网页地址，是因特网上标准的资源的地址。
JDK	Java Development Kit（Java 开发工具包）。
Java SE	Java Standard Edition（Java 标准版）。
JRE	Java Runtime Environment（Java 运行环境）。运行 JAVA 程序所必须的环境的集合，包含 JVM 标准实现及 Java 核心类库。
CTD	Click To Dialog（点击拨号）。
XML	extensible markup language（可扩展标记语言）。用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。

版权说明

本产品使用了以下商业软件，用户应自行付费获得相关商业软件权利人的合法授权。

jdk-6u20-windows-i586

Eclipse IDE for Java Developers

修改记录

日期	修订版本	修改描述	配套版本	作者
2012-05-18	1.01	首次发布，并应用于首次ISV培训	eSpace UC V100R002C01	刘杨、袁坤
2012-06-28	1.02	1.基于培训中反馈的一些问题和建议修改文档中的一些描述细节 2.新增两个场景，分别是同步企业通讯录（3.8章节）和单点登录（3.9章节，统一鉴权场景） 3.新增AppSoap接口描述（6章节） 4.补充附录8.4中的加密算法描述	eSpace UC V100R002C01	袁坤

目录

1 概述

本文是对 eSpace UC 服务端集成开发提供的指导文档，使用者为进行 eSpace UC 服务端集成开发的 ISV 工程师，文档的侧重点在于基本开发方法的掌握，主要描述以下几个方面内容。

1.1 开发环境准备

- 环境说明以及如何部署开发环境
- ISV 集成开发样例，详细举例如何集成 eSpace UC 服务器的接口

1.2 典型开发场景开发举例

- 典型场景的开发举例，分为场景描述，预置条件，实现步骤以及代码片段

1.3 AppXML 调用接口

- AppXML 接口为第三方请求 eSpace UC 服务器的接口
- 详细列举接口的 URL，请求、响应消息的格式及错误码和解决方法

1.4 AppXML 通知接口

- AppXML 通知接口为为回调消息接口，由 eSpace UC 服务器推送给第三方提供的接口
- 详细列举接口的 URL，请求消息的格式及开发样例

1.5 AppSOAP 请求接口

- eSpace UC 服务器中开放的 SOAP 接口的格式描述。

1.6 Web 集成接口

- 通过 JavaScript 初始化 eSpace 状态呈现对象，实现在 Web 页面上呈现用户状态，并支持发起即时消息、发送文件、音视频呼叫、添加为联系人的操作

1.7 附录

- 包括 XML 消息的构造与解析，HTTP 协议交互的举例，常见的返回码解释以及文档中涉及到的加解密方法

2 开发环境准备

2.1 环境说明

本文说明的开发环境准备是指基于 Java 语言的开发环境搭建与配置，如果 ISV 厂家使用非 Java 语言实现我们开放接口调用，则可以跳过本章内容。

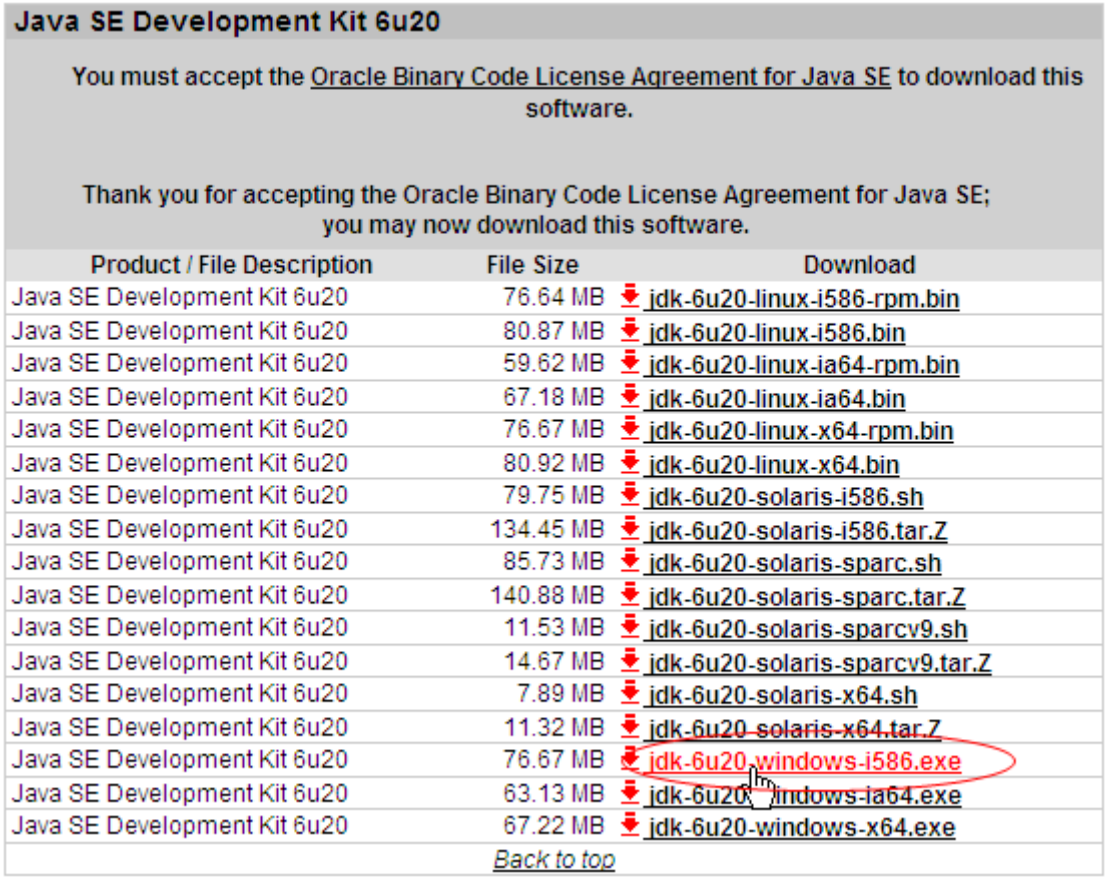
JDK 需要 1.5.0 以上版本即可满足要求，本文使用 1.6.0 版的 JDK 来举例说明。

Java 的 IDE 开发环境有很多，如 Eclipse、NetBeans、IntelliJ 等，这里不做限制，本文使用 Eclipse 来举例说明。

2.2 JDK 安装与配置

2.2.1 下载地址

访问 Oracle 官方网站，选择 Java SE 的下载连接，选择如下版本下载：



Java SE Development Kit 6u20

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Java SE Development Kit 6u20	76.64 MB	jdk-6u20-linux-i586-rpm.bin
Java SE Development Kit 6u20	80.87 MB	jdk-6u20-linux-i586.bin
Java SE Development Kit 6u20	59.62 MB	jdk-6u20-linux-ia64-rpm.bin
Java SE Development Kit 6u20	67.18 MB	jdk-6u20-linux-ia64.bin
Java SE Development Kit 6u20	76.67 MB	jdk-6u20-linux-x64-rpm.bin
Java SE Development Kit 6u20	80.92 MB	jdk-6u20-linux-x64.bin
Java SE Development Kit 6u20	79.75 MB	jdk-6u20-solaris-i586.sh
Java SE Development Kit 6u20	134.45 MB	jdk-6u20-solaris-i586.tar.Z
Java SE Development Kit 6u20	85.73 MB	jdk-6u20-solaris-sparc.sh
Java SE Development Kit 6u20	140.88 MB	jdk-6u20-solaris-sparc.tar.Z
Java SE Development Kit 6u20	11.53 MB	jdk-6u20-solaris-sparcv9.sh
Java SE Development Kit 6u20	14.67 MB	jdk-6u20-solaris-sparcv9.tar.Z
Java SE Development Kit 6u20	7.89 MB	jdk-6u20-solaris-x64.sh
Java SE Development Kit 6u20	11.32 MB	jdk-6u20-solaris-x64.tar.Z
Java SE Development Kit 6u20	76.67 MB	jdk-6u20-windows-i586.exe
Java SE Development Kit 6u20	63.13 MB	jdk-6u20-windows-ia64.exe
Java SE Development Kit 6u20	67.22 MB	jdk-6u20-windows-x64.exe

Back to top

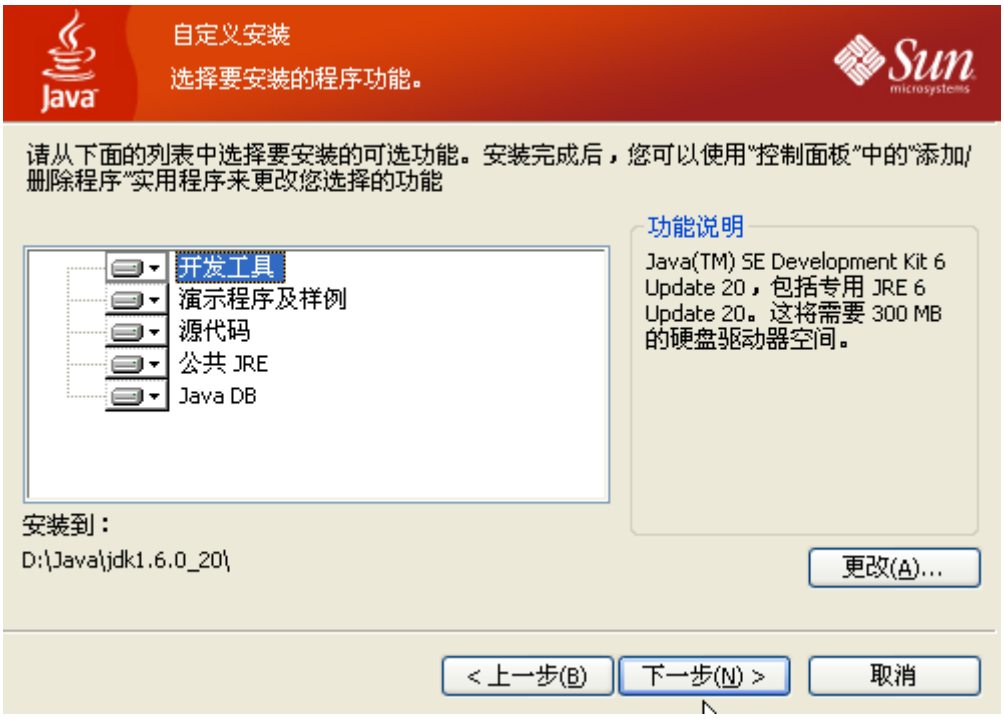
参考连接：

http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u20-oth-JPR

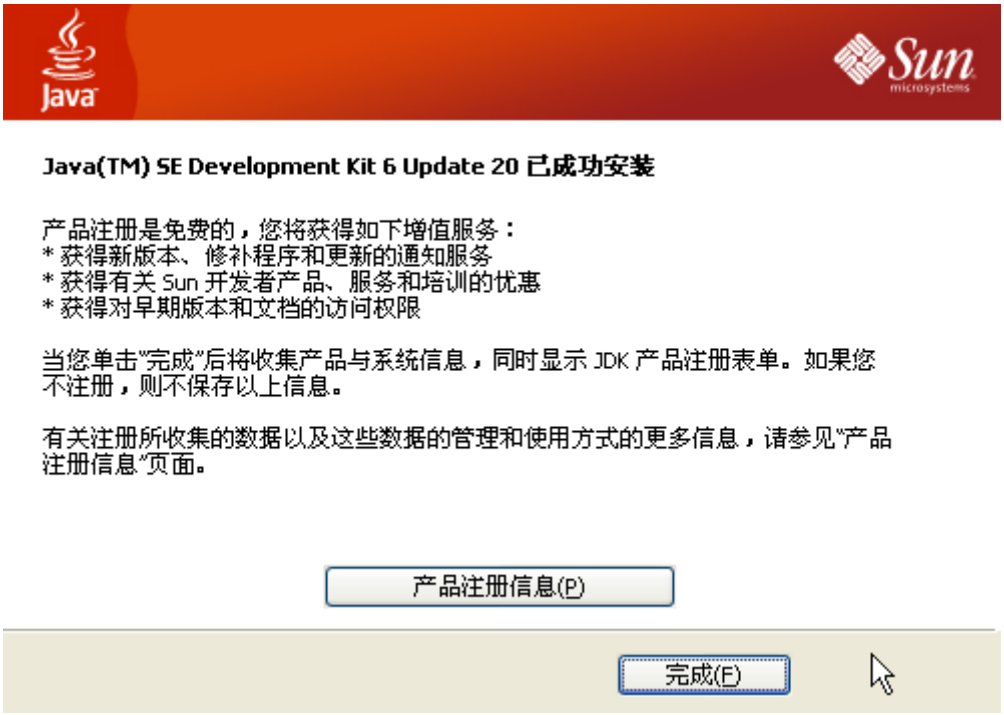
2.2.2 安装

下载完成后双击 jdk-6u20-windows-i586.exe 根据提示安装。

比如我们把 JDK 安装到 D:\Java\jdk1.6.0_20\目录，请记住这个路径，后面配置 JDK 的时候会用到。

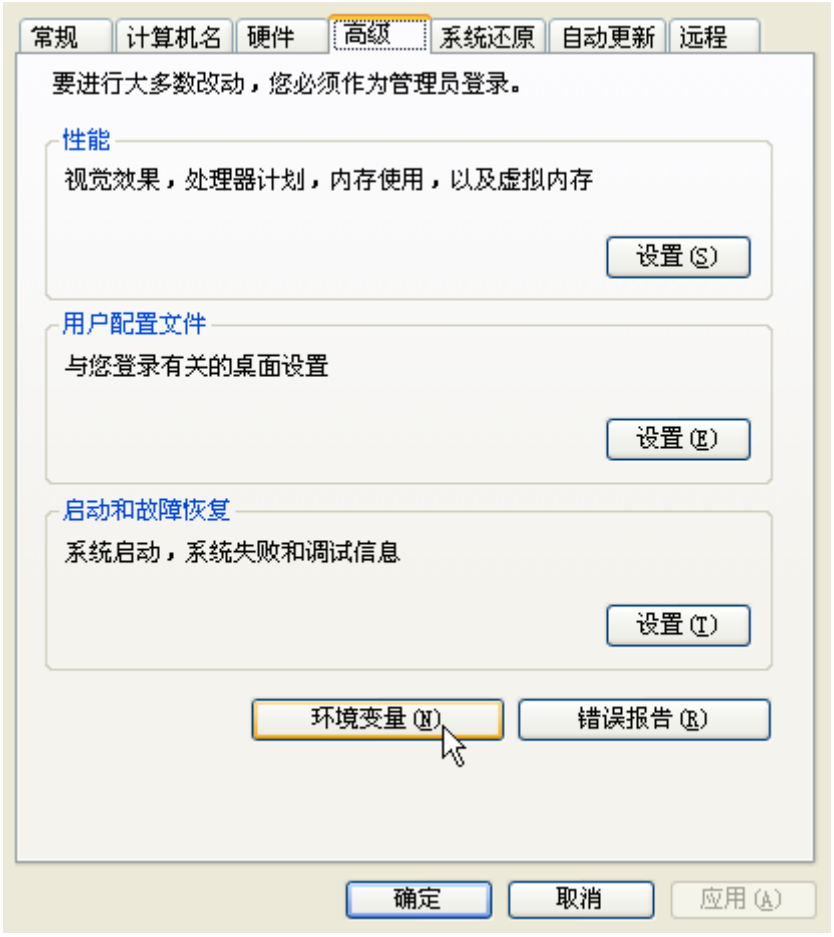


完成安装。



2.2.3 配置

打开“系统属性”窗口，选择“高级”页签，单击“环境变量”



在系统变量中新增 JAVA_HOME 环境变量，变量值为刚才 JDK 的安装路径。

2.3 Eclipse 安装与配置

2.3.1 下载地址

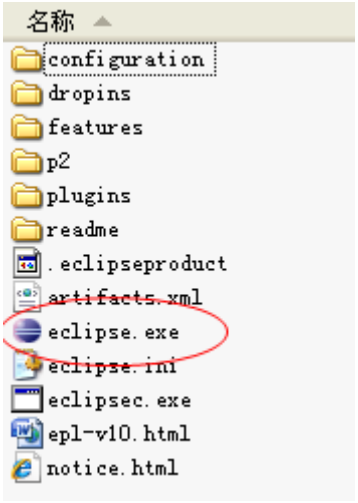
访问 Eclipse 官方网站，下载 Eclipse IDE for Java Developers.

参考连接：

http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/indigo/SR2/eclipse-java-indigo-SR2-win32.zip

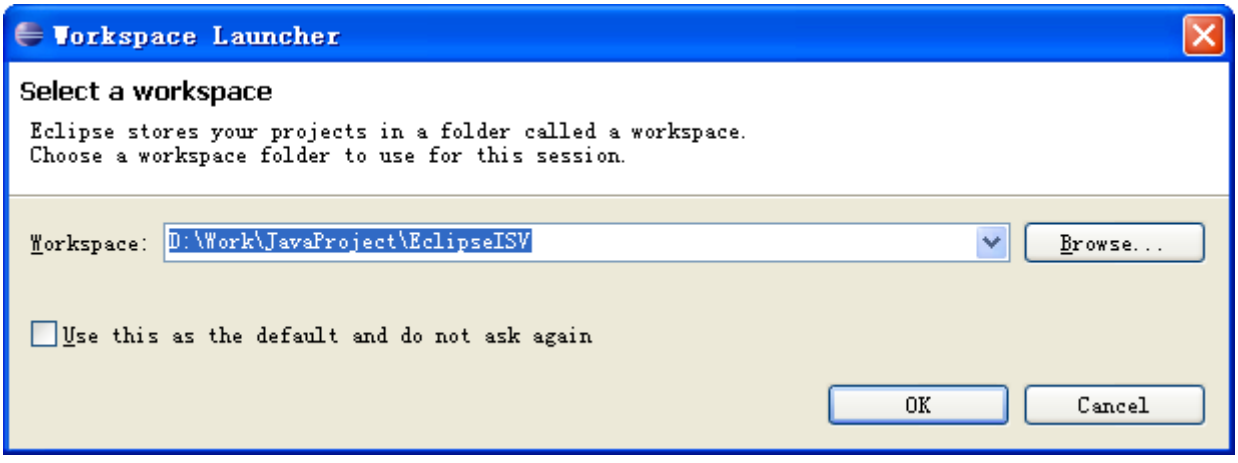
2.3.2 安装

下载后的压缩包解压到任何目录直接可以使用，双击 eclipse.exe 直接打开。



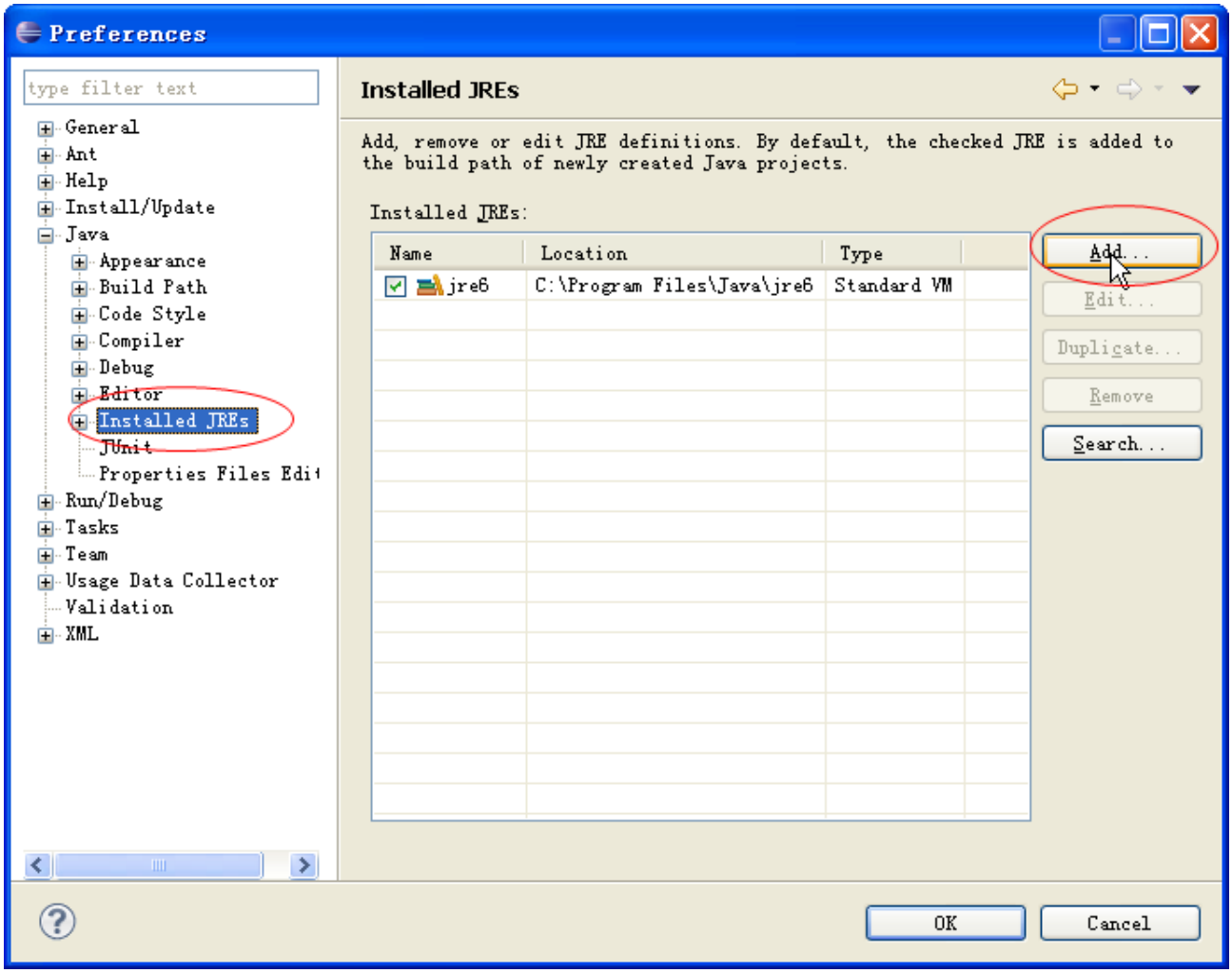
2.3.3 配置

启动 Eclipse 后，Eclipse 会首先询问你设定工程路径，我们可以为我们的 ISV 开发单独设定一个目录来存放我们开发的集成工程，本文设定为 D:\Work\JavaProject\EclipseISV，如果你不希望下次启动再次询问，请勾选“Use this as the default and do not ask again”

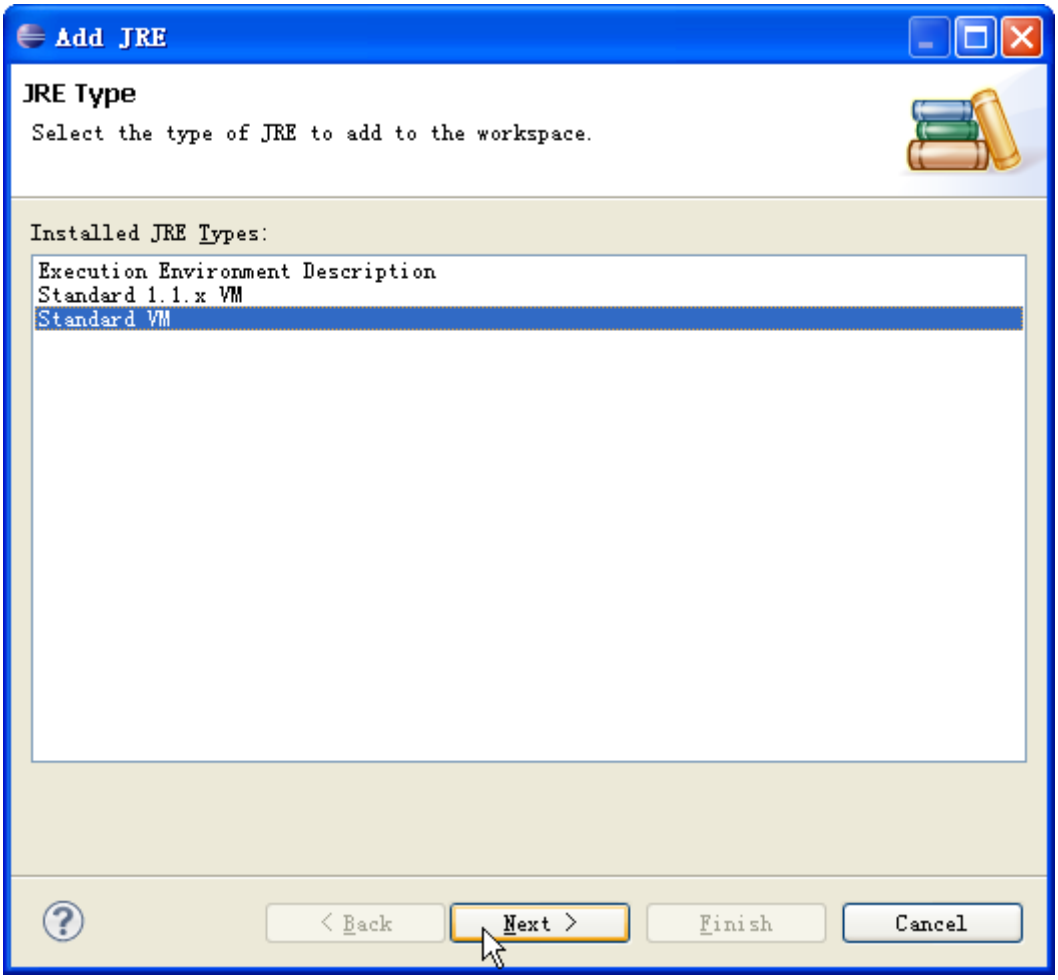


打开设置窗口

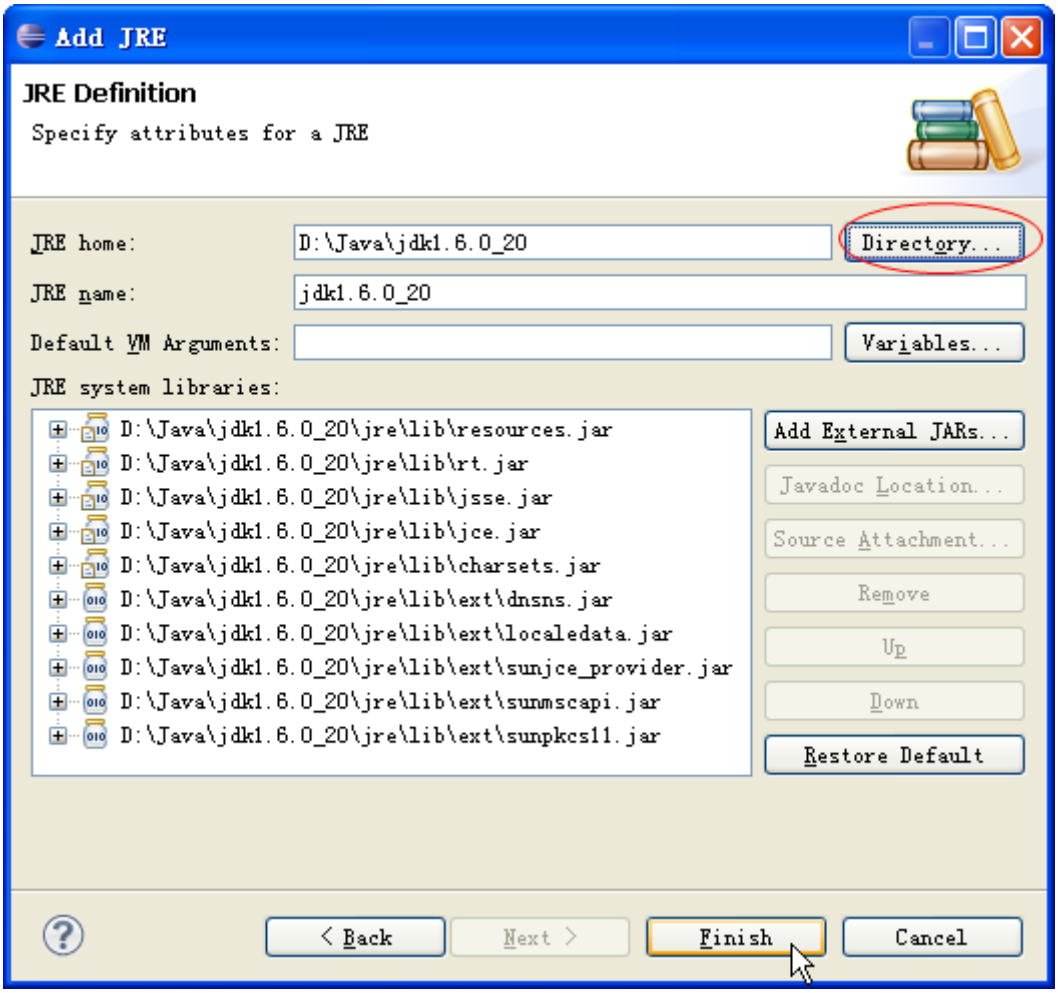
选择设置“Installed JREs”，单击“Add”。



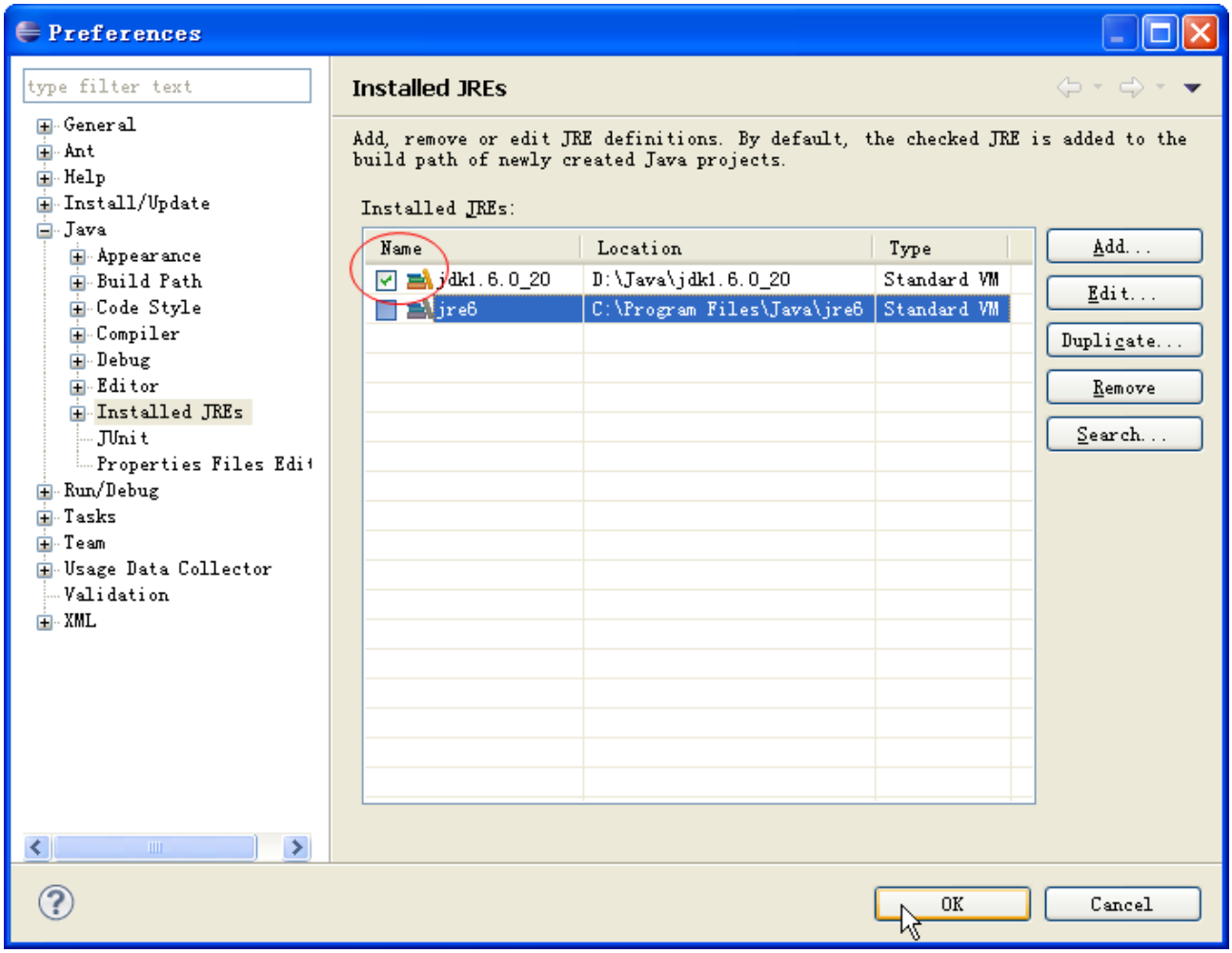
选择“Standard VM”，单击“Next”。



单击“Directory”选择刚才安装 JDK 的路径“D:\Java\jdk1.6.0_20”，如下图所示，单击完成。



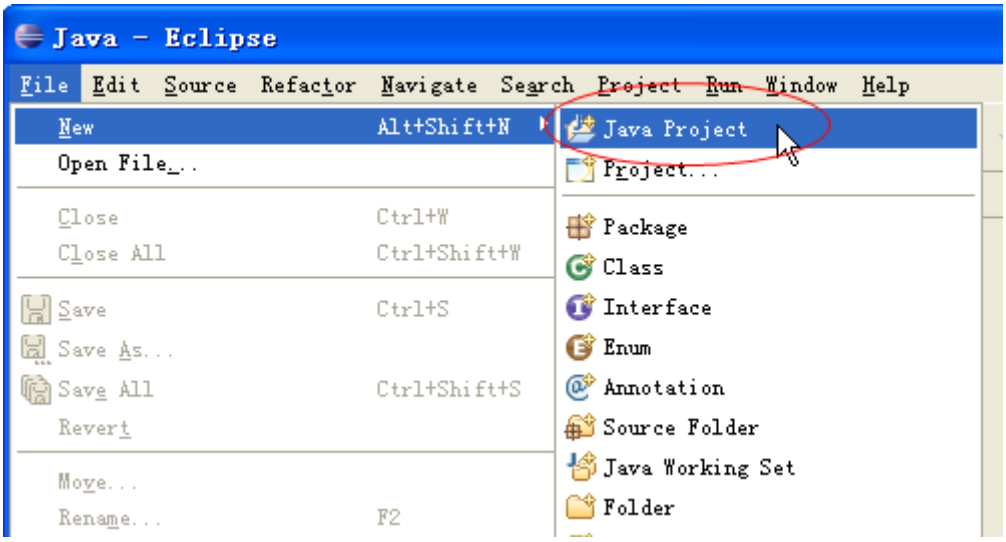
勾选刚才 JDK 作为默认使用的 JDK，如下图所示：



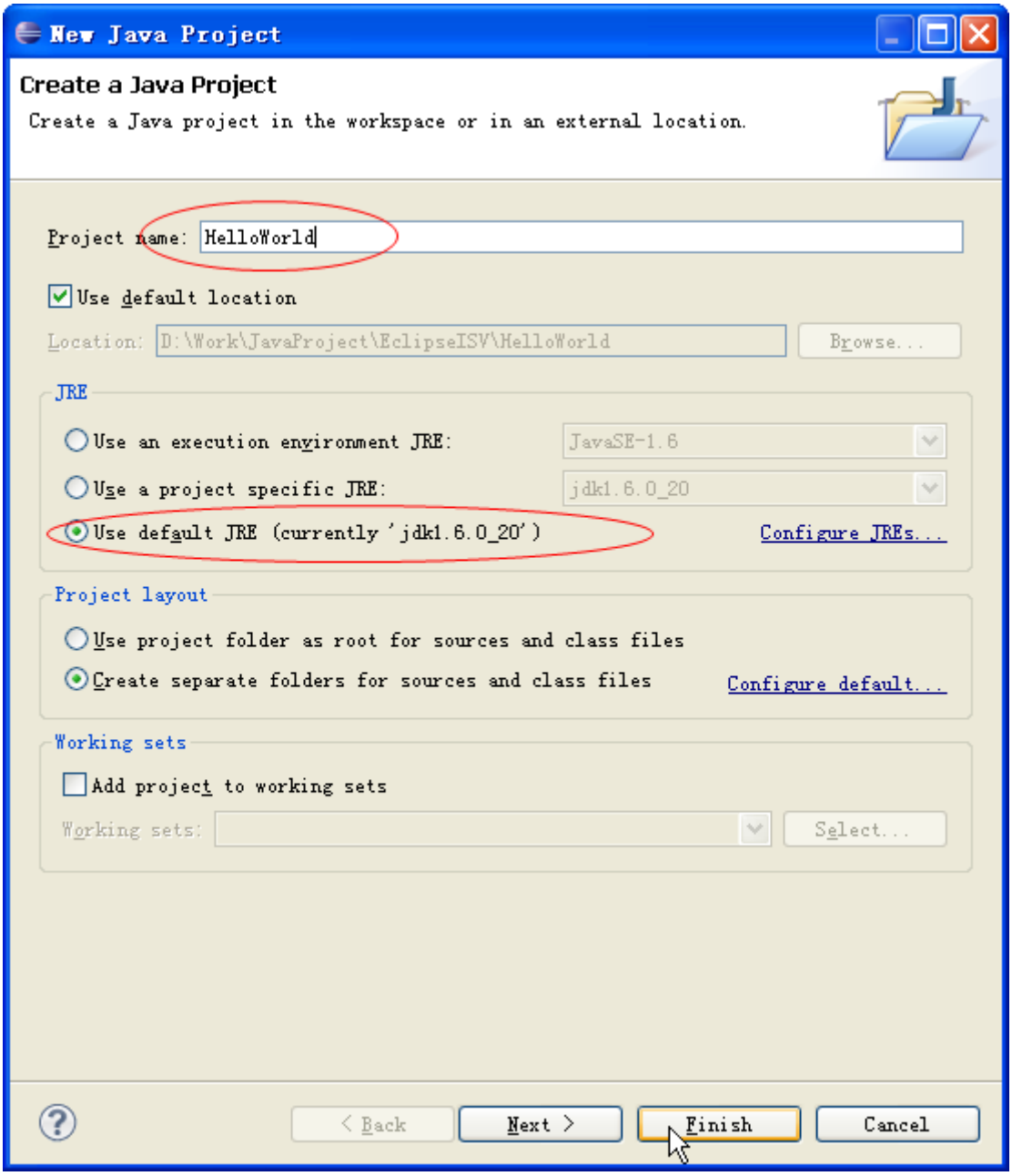
2.4 新建第一个 Java 工程

2.4.1 新建 Java 工程

通过 eclipse 菜单新建一个叫“HelloWorld”的 Java 工程

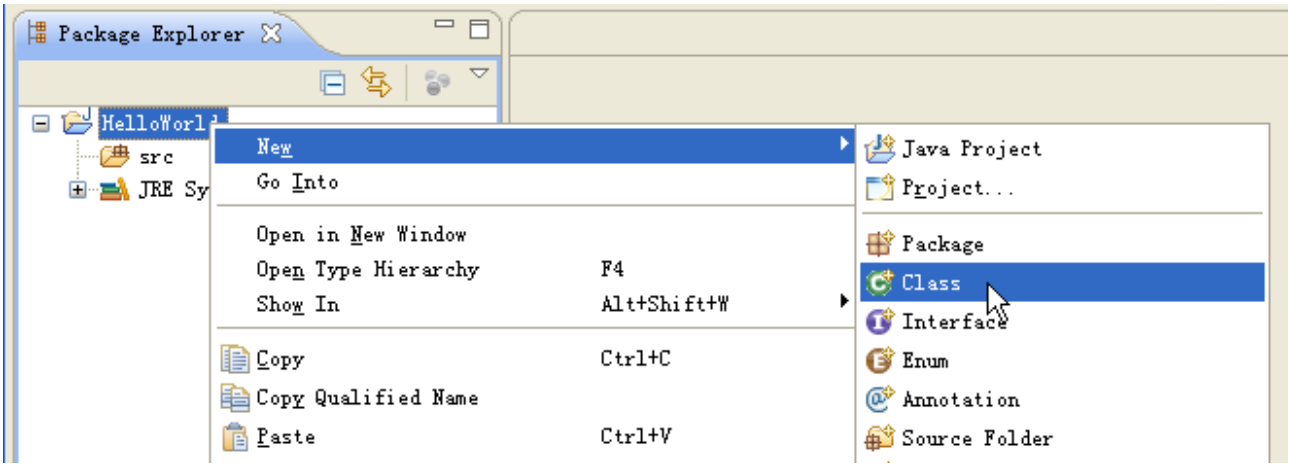


选择使用刚才添加的默认 JRE 作为开发 JDK。

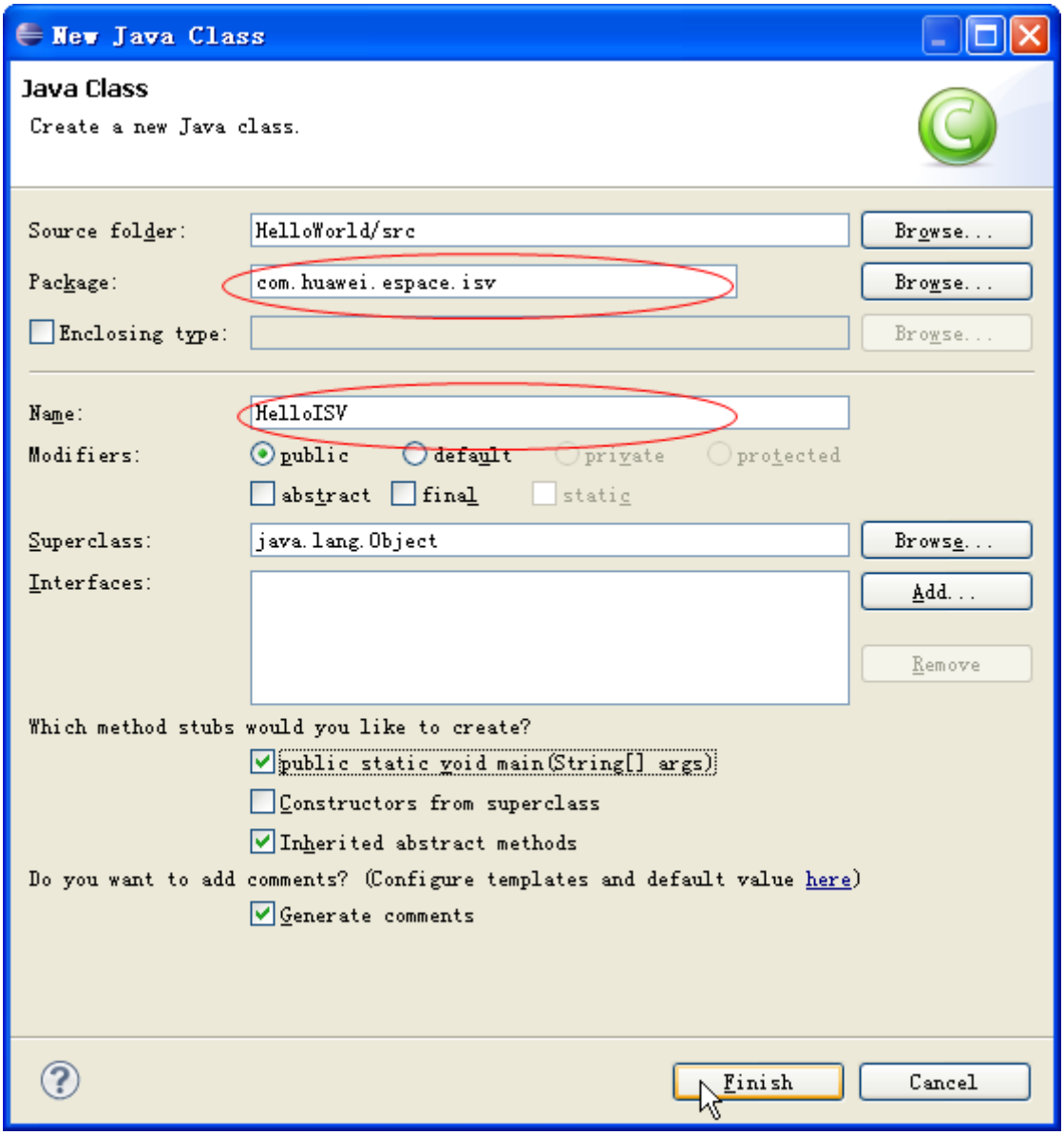


2.4.2 新建 Java 类

鼠标右键单击刚才新建的工程名称，通过菜单选择新建“Class”。



设定“Package”，这个是 Java 的包名；设定“Name”，这个是给 Java 类设定一个名字。

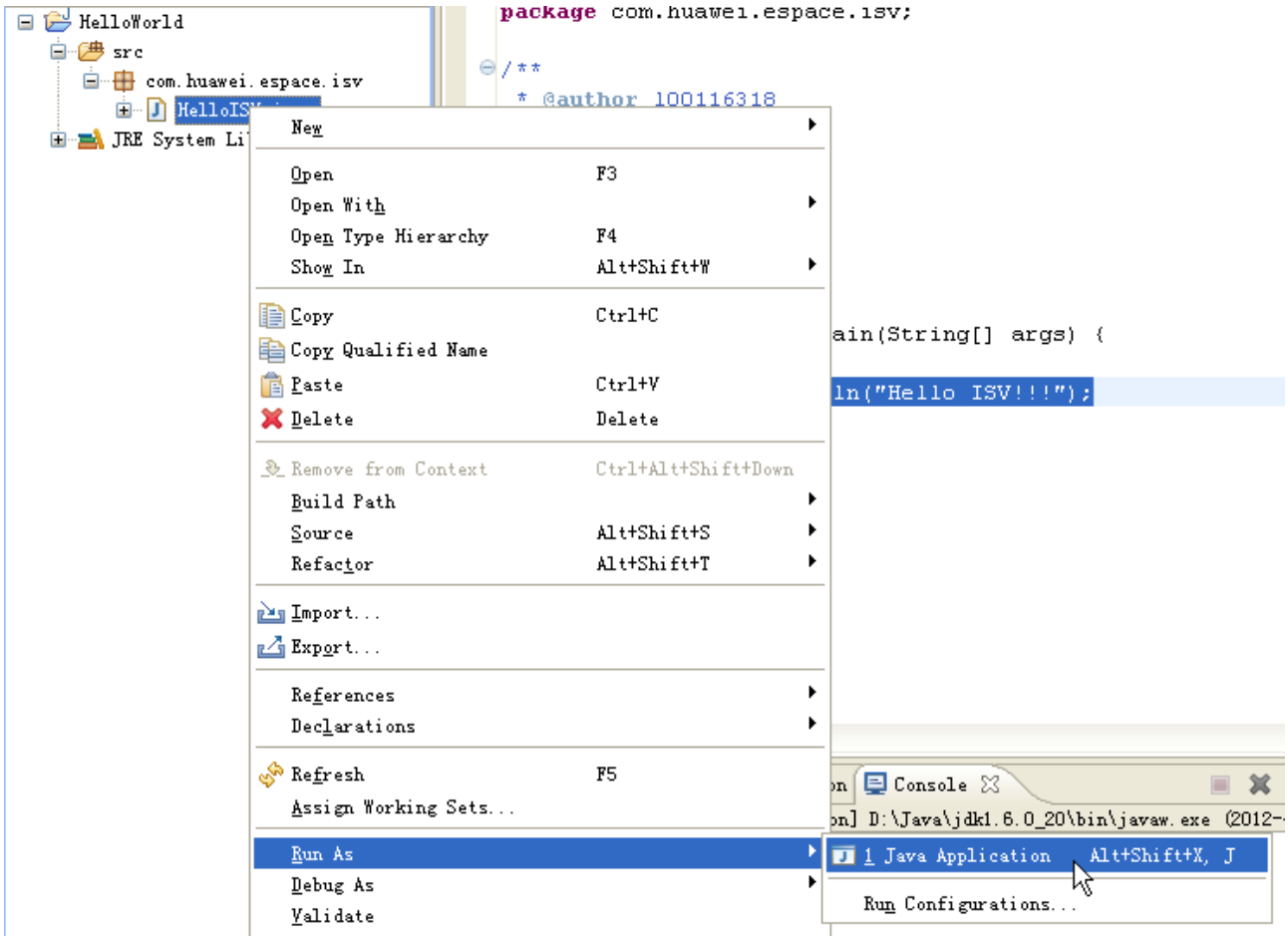


2.4.3 编写代码并运行

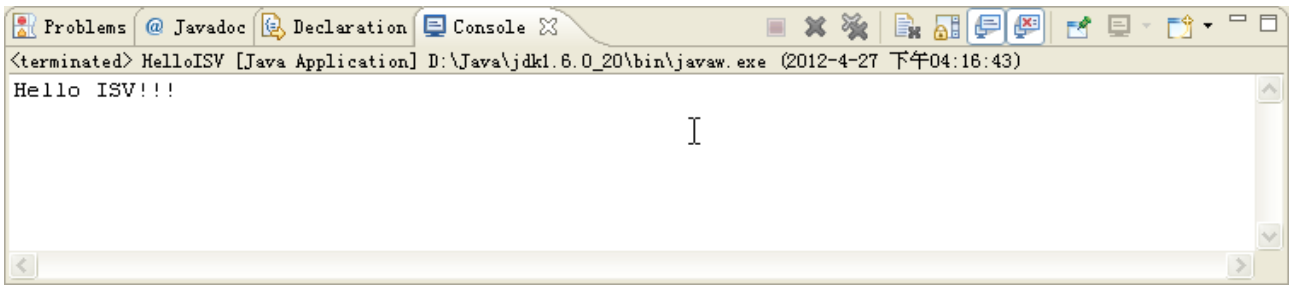
在 main 方法里面新增一行用于打印输出的代码并保存：

```
System.out.println("Hello ISV!!!");
```

在“HelloISV.java”文件上单击邮件，选择下图菜单运行 Java 程序。



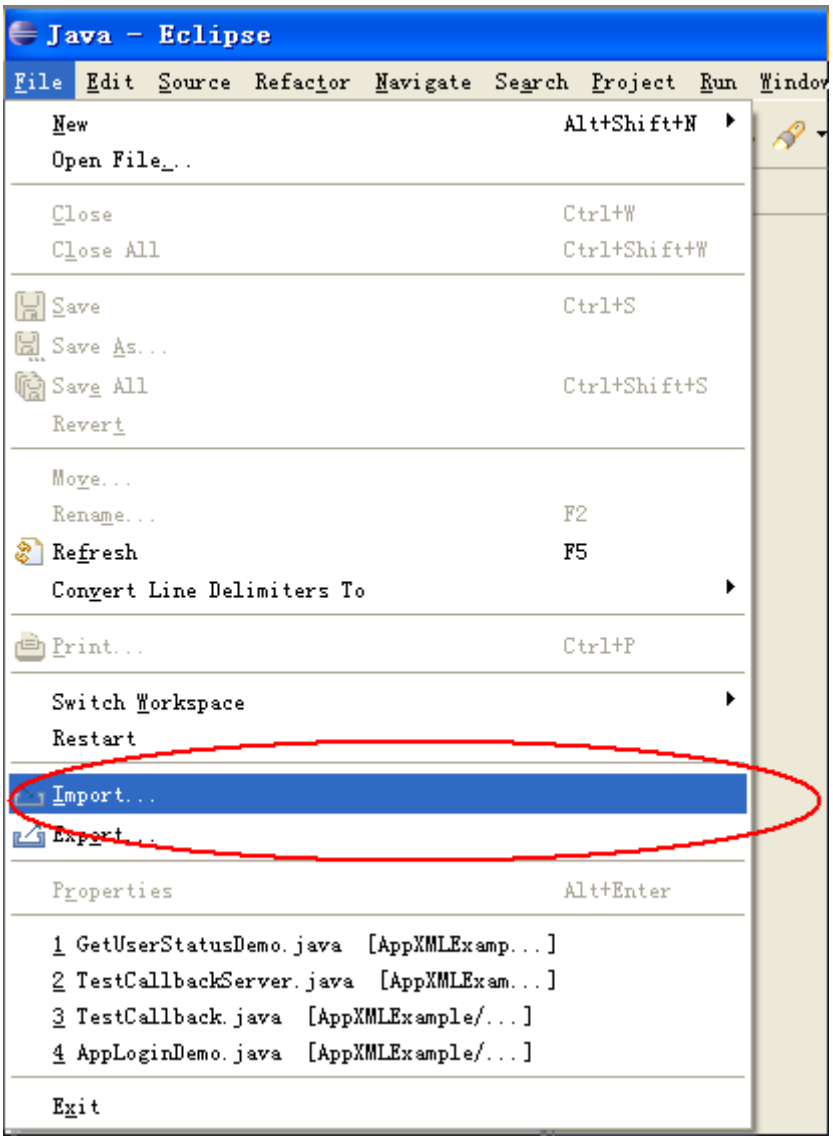
运行结果：



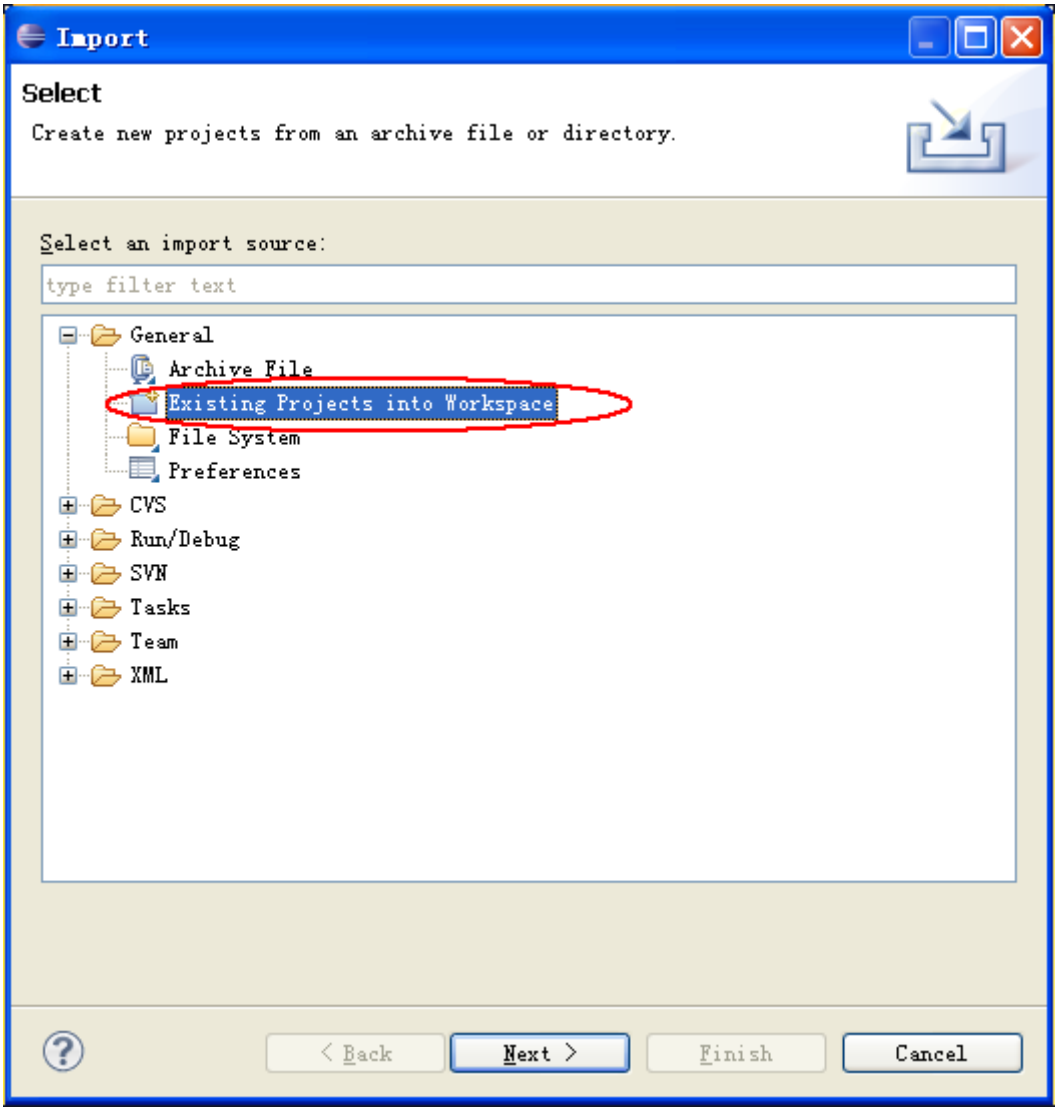
2.5 导入已有工程

2.5.1 导入已有工程

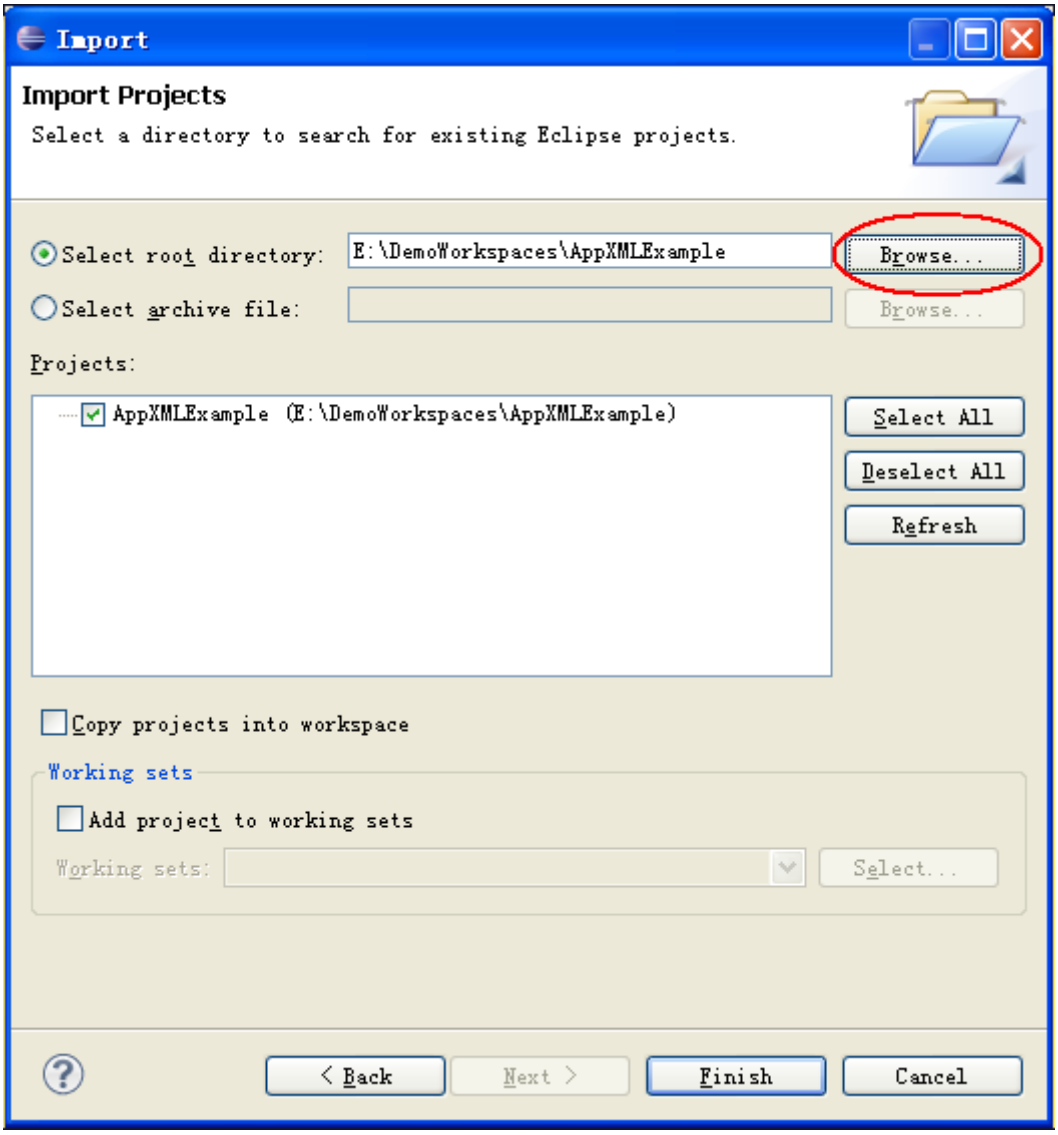
通过 Eclipse 菜单导入工程



选择“Existing Projects into Workspace”，点击 Next



选择工程路径，点击 Finish



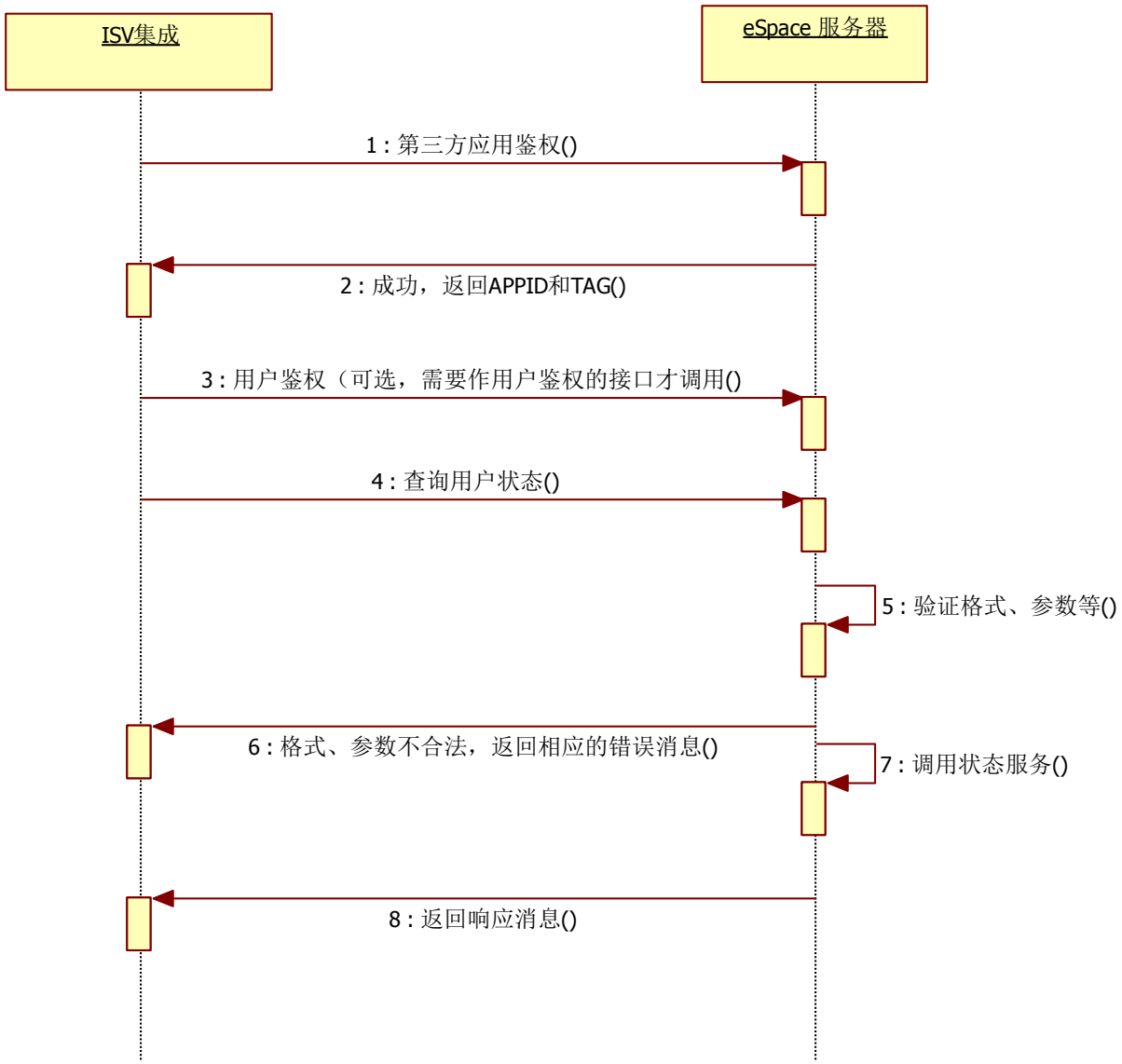
至此，工程导入完毕。

2.6 ISV 集成开发过程介绍

2.6.1 场景描述

客户方某个用户想拉起另外一个或一批用户进入电话会议，这时需要知道当前需要进入电话会议的用户的状态是否为非通话中。

2.6.2 流程图



流程描述：1.请求第三方应用鉴权接口

2.鉴权成功，返回相应的 appid 和 tag 信息

3.请求用户鉴权接口，此步骤只有在调用其他需要作用户鉴权的借口之前才调用。

4.请求查询用户状态接口

5.验证请求消息的格式、参数信息等

6.如果验证不通过，即格式、参数等不合法，返回相应的错误码和错误消息

7.验证通过，作查询数据库，推送通知等接口所对应的操作

8.返回成功或失败的响应消息

2.6.3 第三方应用鉴权

本文档中的接口都是以 HTTP 协议交互的，交互举例可以参考本文档 8.2 附录 B: HTTP 协议交互举例。以查询用户状态为例编写，首先查询用户状态依赖于第三方应用鉴权，所以先写第三方应用鉴权代码（消息格式见本文档 4.1.2 和 4.1.3 章节）。

场景描述

客户需要针对 eSpace UC 服务器作集成开发，而此接口用于所有第三方应用接入的鉴权。使用其他接口前，必须请求此接口获取 tag 后才能使用其他接口。tag 有过期机制，如果使用接口时返回码为 93006，则说明 tag 过期，需要使用此接口重新申请 tag。

预置条件

- eSpace UC 服务器部署并调试完成
- 获取 eSpace UC 服务器 IP 地址和端口
- 管理员已经为应用分配接入账号及密码

实现步骤

- 构造请求 XML 消息，消息定义请参考本文档 4.1.2 章节。
- 发送请求消息，并获取响应消息。
- 记录获取的 tag 值，为后续其它接口调用时使用。

```
public static void main(String[] args)
{
    new AppLoginDemo().appLogin("mobile","mobile");
}
```

相关依赖方法如下：

```
public AppBean appLogin(String guid,String pwd)
{
    System.out.println("-----------------------------------------第三方应用登录开始-------------------------------------------------------");
    System.out.println("输入参数为：guid="+ guid + ",pwd=" + pwd);

    //封装XML请求
    Document document = DocumentHelper.createDocument();
    Element message = document.addElement("message");
    Element head = message.addElement("head");
    head.addElement("guid").setText(guid);
    Element body = message.addElement("body");
    Element params = body.addElement("params");
    params.addElement("pwd").setText(pwd);
    //注册第三方回调接口地址
    params.addElement("callbackurl")
        .setText("http://ip:port/serviceName/callback");

    //获取请求URL
    String url = URLUtils.getURL("appLogin");
    System.out.println("请求地址：" + url);
    System.out.println("请求消息内容 :" + document.asXML());

    //请求第三方应用鉴权接口
```

```
        String msg = XmlService.httpPostRequest(document.asXML(), url);


        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        //解析响应消息头
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        AppBean ab = new AppBean();
        if ("0".equals(retCode))
        {
            //获取相应消息
            ab.setAppid(XmlService.getElementText(msgDoc,
"/message/body/params/appid"));
            ab.setTag(XmlService.getElementText(msgDoc,
"/message/body/params/tag"));
        }
        else
        {
            System.out.println("*********querystate state failed cause is " +
retContext);

        }
        System.out.println("-----------------------------------------获取结果为
-------------------------------------------------------------");
        System.out.println("appid="+ab.getAppid()+",tag=" + ab.getTag());
        return ab;
    }

        //获取请求URL
    public static String getURL(String method)
    {
        String action = "";
        if ("appLogin".equals(method))
        {
            action = "loginAction.do?method=appLogin";
        }
        else if ("userLogin".equals(method))
        {
            action = "loginAction.do?method=userLogin";
        }
        else if ("call".equals(method))
        {
            action = "ctdAction.do?method=call";
```

```
        }
        else if ("batchGetUserState".equals(method))
        {
            action = "imManageAction.do?method=batchGetUserState";
        }
        else if ("imChat".equals(method))
        {
            action = "imManageAction.do?method=imChat";
        }
        else if ("queryEnterprise".equals(method))
        {
            action = "queryEnterpriseAction.do?method=queryEnterprise";
        }
        else if ("queryGroup".equals(method))
        {
            action = "queryGroupAction.do?method=queryGroup";
        }
        else if ("queryGroupMember".equals(method))
        {
            action = "queryGroupAction.do?method=queryGroupMember";
        }
        else if ("queryIMGroup".equals(method))
        {
            action = "imManageAction.do?method=queryIMGroup";
        }
        else if ("queryIMGroupMember".equals(method))
        {
            action = "imManageAction.do?method=queryIMGroupMember";
        }
        else if ("searchEmployee".equals(method))
        {
            action = "queryEnterpriseAction.do?method=searchEmployee";
        }
        else if ("sendSms".equals(method))
        {
            action = "smsAction.do?method=sendSms";
        }

        return "http://192.168.10.10/" + action;
    }
```

运行结果如下：

```
-------------------------------------------第三方应用登录开始
------------------------------------------------------------
输入参数为：guid=mobile,pwd=mobile
请求地址：http://10.166.42.201/loginAction.do?method=appLogin
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><guid>mobile</guid></head><body><params><pwd>mobile</pwd></params
></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><appid>11</appid><tag>46DE16414B265E260761EF8F
4C19FA1E</tag><eid>1</eid></params></body></message>
-------------------------------------------获取结果为
------------------------------------------------------------
appid=11,tag=46DE16414B265E260761EF8F4C19FA1E
```

2.6.4 用户登录鉴权

场景描述

第三方应用的用户鉴权。先请求此接口登录，再使用其他接口。客户需要使用 AppServer 提供的接口，而这些接口在调用的时候，会对当前用户是否登录进行鉴权。

预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值
- 登录的用户必须是 BMU 上已存在的 UC 用户

实现步骤

- 构造请求 XML 消息，消息定义请参考本文档 4.2.2 章节
- 发送请求消息，并获取响应消息

代码片段

```
    public static void main(String[] args)
    {
       AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");

        //用户登录
        new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),"isv10004",
//参数说明参见本文档8.4.1 章节
EncryptUtils.encode("123456", "isv10004"),"20120428000000");
    }
```

依赖方法片段如下

```
    public UserBean userLogin(String appid,String tag,String accounts,String
password,String timestamp)
    {
        System.out.println("-----------------------------------------用户登录开始
-----------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为：\n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("password:");
        sb.append(password);
        sb.append("\n");
        sb.append("timestamp:");
        sb.append(timestamp);
        System.out.println(sb.toString());
```

```
        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("password").setText(password);
        params.addElement("timestamp").setText(timestamp);
        //params.addElement("clienttype").setText("1");

        String url = URLUtils.getURL("userLogin");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        UserBean ub = new UserBean();
        if ("0".equals(retCode))
        {
            ub.setBindno(XmlService.getElementText(msgDoc,
"/message/body/params/bindno"));
            ub.setFuncid(XmlService.getElementText(msgDoc,
"/message/body/params/funcid"));
            ub.setUcpin(XmlService.getElementText(msgDoc,
"/message/body/params/ucpin"));
            ub.setMaxsmsnum(XmlService.getElementText(msgDoc,
"/message/body/params/maxsmsnum"));
            ub.setMaxconfnum(XmlService.getElementText(msgDoc,
"/message/body/params/maxconfnum"));
            ub.setIsupdate(XmlService.getElementText(msgDoc,
"/message/body/params/isupdate"));
            ub.setServerip(XmlService.getElementText(msgDoc,
"/message/body/params/serverip"));
            ub.setServerport(XmlService.getElementText(msgDoc,
"/message/body/params/serverport"));
            ub.setServerdomain(XmlService.getElementText(msgDoc,
"/message/body/params/serverdomain"));
```

```
            ub.setCountrycode(XmlService.getElementText(msgDoc,
"/message/body/params/countrycode"));
            ub.setUseragent(XmlService.getElementText(msgDoc,
"/message/body/params/useragent"));
            ub.setVoipnum(XmlService.getElementText(msgDoc,
"/message/body/params/voipunm"));
            ub.setVoippin(XmlService.getElementText(msgDoc,
"/message/body/params/voippin"));
            ub.setOutgoingacccode(XmlService.getElementText(msgDoc,
"/message/body/params/outgoingacccode"));
            ub.setStaffaccount(accounts);
        }
        else
        {
            System.out.println(retContext);
        }

System.out.println("------------------------------------------------------------用户
的属性为----------------------------------------");
        System.out.println("Bindno:" + ub.getBindno());
        System.out.println("Funcid:" + ub.getFuncid());
        System.out.println("Ucpin:" + ub.getUcpin());
        System.out.println("Maxsmsnum:" + ub.getMaxsmsnum());
        System.out.println("Maxconfnum:" + ub.getMaxconfnum());
        System.out.println("Isupdate:" + ub.getIsupdate());
        System.out.println("Serverip:" + ub.getServerip());
        System.out.println("Serverport:" + ub.getServerport());
        System.out.println("Serverdomain:" + ub.getServerdomain());
        System.out.println("Countrycode:" + ub.getCountrycode());
        System.out.println("Useragent:" + ub.getUseragent());
        System.out.println("Voipnum:" + ub.getVoipnum());
        System.out.println("Voippin:" + ub.getVoippin());
        System.out.println("Outgoingacccode:" + ub.getOutgoingacccode());

        return ub;
    }
```

运行结果如下

```
-----------------------------------------第三方应用登录开始
------------------------------------------------------------
输入参数为：guid=mobile,pwd=mobile
请求地址：http://10.166.42.201/loginAction.do?method=appLogin
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><guid>mobile</guid></head><body><params><pwd>mobile</pwd></params
></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><appid>11</appid><tag>46DE16414B265E260761EF8F
4C19FA1E</tag><eid>1</eid></params></body></message>
-----------------------------------------获取结果为
------------------------------------------------------------
appid=11,tag=46DE16414B265E260761EF8F4C19FA1E
-----------------------------------------用户登录开始
------------------------------------------------------------
输入参数为：
appid:11
```

```
tag:46DE16414B265E260761EF8F4C19FA1E
accounts:isv10004
password:WEFFBQUG
timestamp:20120428000000
请求地址：http://10.166.42.201/loginAction.do?method=userLogin
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>46DE16414B265E260761EF8F4C19FA1E</tag><acco
unts>isv10004</accounts></head><body><params><password>WEFFBQUG</password><times
tamp>20120428000000</timestamp></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><bindno>20004</bindno><funcid>1010000111100000
00000000000000011000101001000000000000000000000100110000010000000000000000000000
00000000000000000000000000000000</funcid><ucpin/><maxconfnum>16</maxconfnum><max
smsnum>30</maxsmsnum><isupdate>1</isupdate><serverip>10.166.42.86</serverip><ser
verport>5060</serverport><serverdomain>espace.huawei.com</serverdomain><countryc
ode/><useragent>Huawei
SoftCoPCA</useragent><voippin/><voipunm>20004pc</voipunm><outgoingacccode>12530<
/outgoingacccode></params></body></message>
------------------------------------------------------------用户的属性为
----------------------------------------
Bindno:20004
Funcid:10100001111000000000000000000000110001010010000000000000000000010011000001
0000000000000000000000000000000000000000000000000000000000000000
Ucpin:
Maxsmsnum:30
Maxconfnum:16
Isupdate:1
Serverip:10.166.42.86
Serverport:5060
Serverdomain:espace.huawei.com
Countrycode:
Useragent:Huawei SoftCoPCA
Voipnum:20004pc
Voippin:
Outgoingacccode:12530
```

2.6.5 查询用户状态

消息格式见本文档 4.8.2 和 4.8.3 章节，如下：

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        new GetUserStatusDemo().batchGetUserState(ab.getAppid(),ab.getTag(),
"isv10004","isv10004;isv10005");
    }
```

相关依赖方法如下：

```
    //批量查询用户状态
    public List<StatuBean> batchGetUserState(String appid,String tag,String
accounts,String staffaccount)
    {
        List<StatuBean> sbs = new ArrayList<StatuBean>();
        System.out.println("-----------------------------------------批量查询用户
状态开始------------------------------------------------------");
```

```
        System.out.println("输入参数为：appid="+ appid + ",tag=" + tag + ",accounts="
+ accounts + ",staffaccount=" + staffaccount);

        //请求状态查询方法
        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);//基本上每个接口都会有相同的消
息头，所以重复代码，继续封装

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("staffaccount").setText(staffaccount);

        String url = URLUtils.getURL("batchGetUserState");
        System.out.println("请求地址：" + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);
        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容：" + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

      //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");
        String[] state;
        String[] staffacount;
        if ("0".equals(retCode))
        {
            //获取响应状态信息
            List<DefaultElement> elems1 =
msgDoc.selectNodes("/message/body/paramlist/bean/status");
            state = new String[elems1.size()];
            for (int i=0; i<elems1.size();i++)
            {
                DefaultElement elem = elems1.get(i);
                String imstate = elem.getText();
                state[i] = imstate;
            }
            //获取响应账号信息
            List<DefaultElement> elems2 =
msgDoc.selectNodes("/message/body/paramlist/bean/staffaccount");
            staffacount = new String[elems2.size()];
            for (int i=0; i<elems2.size();i++)
            {
```

```
                DefaultElement elem = elems2.get(i);
                String user = elem.getText();
                staffacount[i] = user;
            }

            for (int i=0;i<staffacount.length;i++)
            {
                StatuBean sb = new StatuBean();
                sb.setStaffaccount(staffacount[i]);
                sb.setStatus(state[i]);
                sbs.add(sb);
            }
        }
        else
        {
            System.out.println(retContext);
        }

        System.out.println("-----------------------------------------用户状态为
------------------------------------------------------------");
        for(int i=0;i<sbs.size();i++)
        {
            System.out.println("第"+(i+1)+"个用户");

System.out.println("staffaccount="+sbs.get(i).getStaffaccount()+",status="+sbs.g
et(i).getStatus());
        }

        return sbs;
    }

    //封装消息头
    public static void getHead(Element messageEle,String appid,String tag,String
accounts)
    {
        Element head = messageEle.addElement("head");

        Element appidelemnt = head.addElement("appid");
        appidelemnt.setText(appid);

        Element tagelemnt = head.addElement("tag");
        tagelemnt.setText(tag);

        Element accountelemnt = head.addElement("accounts");
        accountelemnt.setText(accounts);

    }
```

```
-----------------------------------------批量查询用户状态开始
------------------------------------------------------
输入参数为:
appid=11,tag=46DE16414B265E260761EF8F4C19FA1E,accounts=isv10004,staffaccount=isv
```

```
10004;isv10005
请求地址：http://10.166.42.201/imManageAction.do?method=batchGetUserState
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>46DE16414B265E260761EF8F4C19FA1E</tag><acco
unts>isv10004</accounts></head><body><params><staffaccount>isv10004;isv10005</st
affaccount></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><paramlist
type="employeeList"><bean><staffaccount>isv10004</staffaccount><status>0</status
><statusinfo/></bean><bean><staffaccount>isv10005</staffaccount><status>0</statu
s><statusinfo/></bean></paramlist></body></message>
-----------------------------------------用户状态为
----------------------------------------------------------------
第1个用户
staffaccount=isv10004,status=0
第2个用户
staffaccount=isv10005,status=0
```

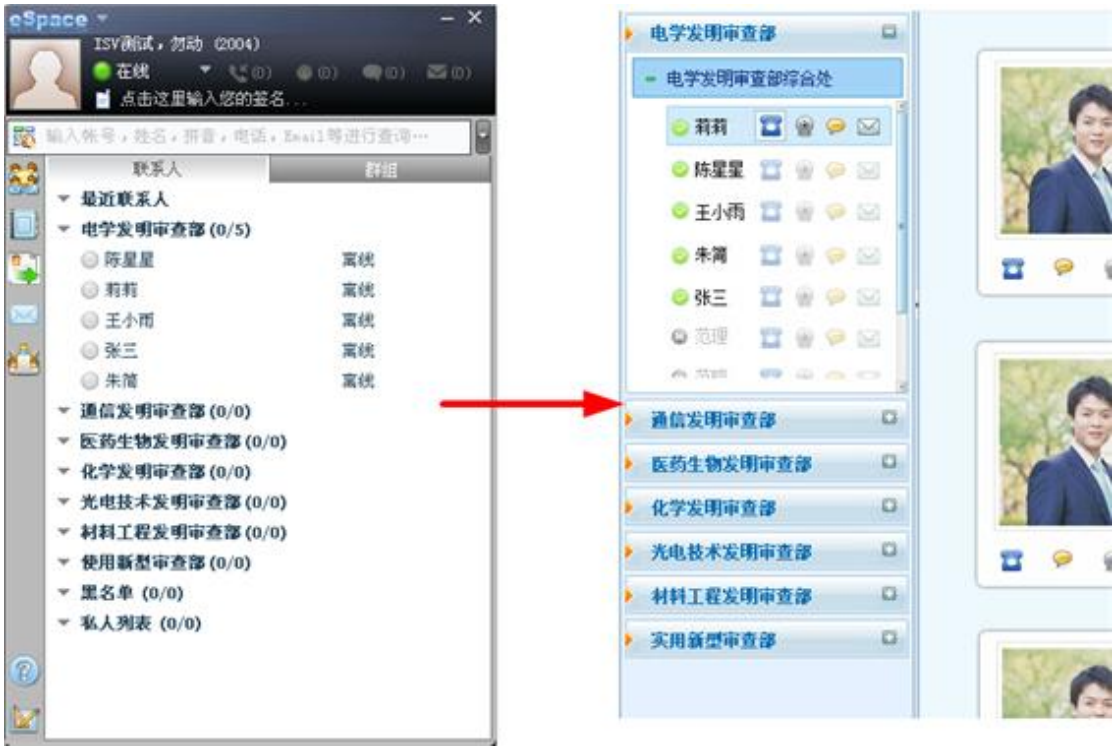
3 典型场景开发举例

说明：本章节需要用到对 XML 格式的消息进行构造与解析，详细方法参考本文档 8.1 附录 A: XML 消息的构造与解析。

3.1 查询好友信息和群组信息

3.1.1 场景描述

客户希望可以把查询好友集成到自己的企业系统中，当员工登录以后，可以查询到自己所有的好友，好友分组，群组，群成员。如下图中，左为 eSpace 客户端中的好友及好友分组，右为某企业集成后的效果。



此场景使用的接口有：查询用户好友列表，查询用户好友分组列表，查询群组列表，查询群组成员列表。

3.1.2 预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值
- 已经调用用户鉴权接口（用户已登录）

3.1.3 实现步骤

步骤 1 获取好友分组信息。

步骤 2 获取好友信息。

步骤 3 获取群组信息。

步骤 4 获取群组成员信息

查询用户好友列表代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new QueryGroupMemberDemo().queryGroupMember(ab.getAppid(),ab.getTag(),
ub.getStaffaccount(),"1","10","1");

    }
```

依赖方法片段如下

```
    public void queryGroupMember(String appid,String tag,String accounts,String
groupid,String pagecount,String pagenum)
    {
        System.out.println("-----------------------------------------查询好友分组
成员列表开始-------------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为：\n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("groupid:");
        sb.append(groupid);
        sb.append("\n");
        sb.append("pagecount:");
        sb.append(pagecount);
        sb.append("\n");
        sb.append("pagenum:");
        sb.append(pagenum);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("groupid").setText(groupid);
        params.addElement("pagecount").setText(pagecount);
        params.addElement("pagenum").setText(pagenum);

        String url = URLUtils.getURL("queryGroupMember");
        System.out.println("请求地址：" + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容：" + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
```

```
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        if ("0".equals(retCode))
        {
            String total = XmlService.getElementText(msgDoc,
"/message/body/params/total");
            String sum = XmlService.getElementText(msgDoc,
"/message/body/params/sum");
            String memberid = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/memberid");
            String name = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/name");
            String staffaccount = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffaccount");


System.out.println("------------------------------------------------------------好友
列表为----------------------------------------");

            System.out.println("memberid:" + memberid);
            System.out.println("name:" + name);
            System.out.println("staffaccount:" + staffaccount);
            System.out.println("...");
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
------------------------------------------查询好友列表开始
------------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
groupid:1
pagecount:10
pagenum:1
请求地址：http://10.166.42.201/queryGroupAction.do?method=queryGroupMember
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><groupid>1</groupid><pagecount>10</
pagecount><pagenum>1</pagenum></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head>0<retcode></retcode><retcontext>操作成功
</retcontext></head><body><params><total>24</total><sum>10</sum></params><paraml
ist type="personalList"><bean><memberid>1</memberid><name>ISV测试，勿动
</name><staffaccount>isv10003</staffaccount><sex>1</sex><mobile></mobile><homeph
one></homephone><fax></fax><email></email><bindno>20003</bindno><shortphone><sho
rtphone><officephone></officephone></bean></paramlist></body></message>
------------------------------------------好友列表为
```

```
------------------------------------------------------
memberid:1
name: ISV测试，勿动
staffaccount:isv10003
…
```

查询用户好友分组列表代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new QueryGroupDemo().queryGroup(ab.getAppid(),ab.getTag(),
ub.getStaffaccount(),"10","1");
    }
```

依赖方法片段如下

```
    public void queryGroup(String appid,String tag,String accounts,String
pagecount,String pagenum)
    {
        System.out.println("-----------------------------------------查询好友分组
列表开始-------------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为: \n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("pagecount:");
        sb.append(pagecount);
        sb.append("\n");
        sb.append("pagenum:");
        sb.append(pagenum);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("pagecount").setText(pagecount);
        params.addElement("pagenum").setText(pagenum);

        String url = URLUtils.getURL("queryGroup");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);
```

```
        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        if ("0".equals(retCode))
        {
            String total = XmlService.getElementText(msgDoc,
"/message/body/params/total");
            String sum = XmlService.getElementText(msgDoc,
"/message/body/params/sum");
            String groupid = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/groupid");
            String name = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/name");
            String count = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/count");


System.out.println("------------------------------------------------------------好友
分组列表为----------------------------------------");

            System.out.println("groupid:" + groupid);
            System.out.println("name:" + name);
            System.out.println("count:" + count);
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-------------------------------------------查询好友分组列表开始
------------------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
pagecount:10
pagenum:1
请求地址: http://10.166.42.201/queryGroupAction.do?method=queryGroup
```

```
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><pagecount>10</pagecount><pagenum>1
</pagenum></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><total>3</total><sum>3</sum></params><paramlis
t
type="personalGroupList"><bean><groupid>1</groupid><name>group</name><count></co
unt></bean></paramlist></body></message>
groupid:1
name: group
count: 3
...
```

查询群组列表代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new QueryIMGroupDemo().queryIMGroup(ab.getAppid(),ab.getTag(),
ub.getStaffaccount());
    }
```

依赖方法片段如下

```
    public void queryIMGroup(String appid,String tag,String accounts)
    {
        System.out.println("-----------------------------------------查询群组列表
开始-------------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为：\n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");

        String url = URLUtils.getURL("queryIMGroup");
        System.out.println("请求地址：" + url);
        System.out.println("请求消息内容 :" + document.asXML());
```

```
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容：" + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        //UserBean ub = new UserBean();
        if ("0".equals(retCode))
        {
            String total = XmlService.getElementText(msgDoc,
"/message/body/params/total");
            String sum = XmlService.getElementText(msgDoc,
"/message/body/params/sum");
            String gid = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/gid");
            String count = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/count");
            String name = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/name");

System.out.println("------------------------------------------------------------群组
列表信息为-----------------------------------------");

            System.out.println("gid:" + gid);
            System.out.println("name:" + name);
            System.out.println("count:" + count);

            System.out.println("...");
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-----------------------------------------查询群组列表开始
------------------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
```

```
accounts:isv10004

请求地址：http://10.166.42.201/imManageAction.do?method=queryIMGroup
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params/></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><total>1</total></params><paramlist
type="groupList"><bean><gid>121</gid><name>ISV测试
</name><count>1</count></bean></paramlist></body></message>
------------------------------------------------------------群组列表信息为
----------------------------------------
gid:121
name:ISV测试
count:1
...
```

查询群组成员列表代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new QueryIMGroupMemberDemo().queryIMGroupMember(ab.getAppid(),
ab.getTag(),ub.getStaffaccount(),"1");

    }
```

依赖方法片段如下

```
    public void queryIMGroupMember(String appid,String tag,String accounts,String
gid)
    {
        System.out.println("------------------------------------------查询群组成员
列表开始------------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为：\n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("gid:");
        sb.append(gid);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);
```

```
        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("gid").setText(gid);

        String url = URLUtils.getURL("queryIMGroupMember");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        //UserBean ub = new UserBean();
        if ("0".equals(retCode))
        {
            String total = XmlService.getElementText(msgDoc,
"/message/body/params/total");
            String sum = XmlService.getElementText(msgDoc,
"/message/body/params/sum");
            String headid = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/headid");
            String staffaccount = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffaccount");
            String name = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/name");
            String signature = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/signature");


System.out.println("------------------------------------------------------------群组
成员信息为----------------------------------------");

            System.out.println("headid:" + headid);
            System.out.println("name:" + name);
            System.out.println("staffaccount:" + staffaccount);
            System.out.println("signature:" + signature);
            System.out.println("...");
        }
        else
        {
```

```
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-----------------------------------------查询群组成员列表开始
-----------------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
gid:121
请求地址:http://10.166.42.201/imManageAction.do?method=queryIMGroupMember
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><gid>121</gid></params></body></mes
sage>
响应消息内容:<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><total>1</total></params><paramlist
type="memberList"><bean><staffaccount>isv10004</staffaccount><name>ISV使用，勿动
</name><headid/><signature><![CDATA[]]></signature></bean></paramlist></body></m
essage>
-----------------------------------------------------------群组成员信息为
----------------------------------------
headid:
name:ISV使用，勿动
staffaccount:isv10004
signature:
...
```

3.2 搜索企业通讯录

3.2.1 场景描述

客户希望可以从企业通讯录中条件搜索出员工信息。

此场景下包括查询企业通讯录，查询个人信息和查询用户状态

如下图，查询条件输入“liuyang”，从企业通讯录中查询出所有含“liuyang”信息的员工详细信息，标红部分为员工当前的 eSpace 状态。

3.2.2 预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值

3.2.3 实现步骤

步骤 1 搜索企业通讯录。

步骤 2 查询个人信息。

步骤 3 查询用户状态。

查询企业通讯录代码片段

```
    public static void main(String[] args)
    {
        / //第三方应用登录
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");

        //用户登录
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");

        //查询企业通讯录
        new SearchEmployeeDemo().searchEmployee(ab.getAppid(),ab.getTag(),
"isv10004","isv1000","10","1");

    }
```

依赖方法片段如下

```
   public void searchEmployee(String appid,String tag,String accounts,String
condition,String pagecount,String pagenum)
    {
        System.out.println("-----------------------------------------查询企业通讯
录开始-----------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为：\n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("condition:");
        sb.append(condition);
        sb.append("\n");
        sb.append("pagecount:");
        sb.append(pagecount);
        sb.append("\n");
        sb.append("pagenum:");
        sb.append(pagenum);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("condition").setText(condition);
        params.addElement("pagecount").setText(pagecount);
        params.addElement("pagenum").setText(pagenum);

        String url = URLUtils.getURL("searchEmployee");
        System.out.println("请求地址：" + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容：" + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }
```

```
      //获取返回码和返回提示信息
       String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
       String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

       //UserBean ub = new UserBean();
       if ("0".equals(retCode))
       {
           String total = XmlService.getElementText(msgDoc,
"/message/body/params/total");
           String sum = XmlService.getElementText(msgDoc,
"/message/body/params/sum");
           String staffid = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffid");
           String staffaccount = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffaccount");
           String name = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/name");
           String staffno = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffno");
           String mobile = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/mobile");

System.out.println("------------------------------------------------------------查询
信息为-----------------------------------------");

           System.out.println("staffid:" + staffid);
           System.out.println("name:" + name);
           System.out.println("staffaccount:" + staffaccount);
           System.out.println("staffno:" + staffno);
           System.out.println("mobile:" + mobile);
           System.out.println("...");
       }
       else
       {
           System.out.println(retContext);
       }
   }
```

运行结果如下

```
------------------------------------------查询企业通讯录开始
------------------------------------------------------------
输入参数为:
appid:11
tag:62944B652996BF35892F52C81A1C3D4C
accounts:isv10004
condition:isv1000
pagecount:10
pagenum:1
请求地址: http://10.166.42.243/queryEnterpriseAction.do?method=searchEmployee
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>62944B652996BF35892F52C81A1C3D4C</tag><acco
unts>isv10004</accounts></head><body><params><condition>isv1000</condition><page
count>10</pagecount><pagenum>1</pagenum></params></body></message>
```

```
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><total>6</total><sum>6</sum></params><paramlis
t
type="employeeList"><bean><staffid>8</staffid><name><![CDATA[isv10000]]></name><
staffaccount>isv10000</staffaccount><staffno/><sex>3</sex><mobile/><homephone/><
fax/><email/><bindno>2000</bindno><shortphone/><officephone/><credit><![CDATA[]]
></credit><underwrite><![CDATA[]]></underwrite><voip/><otherphone/><addr><![CDAT
A[]]></addr><zip/><seat><![CDATA[]]></seat></bean><bean><staffid>9</staffid><nam
e><![CDATA[isv10001]]></name><staffaccount>isv10001</staffaccount><staffno/><sex
>3</sex><mobile/><homephone/><fax/><email/><bindno>2001</bindno><shortphone/><of
ficephone/><credit><![CDATA[]]></credit><underwrite><![CDATA[]]></underwrite><vo
ip/><otherphone/><addr><![CDATA[]]></addr><zip/><seat><![CDATA[]]></seat></bean>
<bean><staffid>10</staffid><name><![CDATA[isv10002]]></name><staffaccount>isv100
02</staffaccount><staffno/><sex>3</sex><mobile/><homephone/><fax/><email/><bindn
o>2002</bindno><shortphone/><officephone/><credit><![CDATA[]]></credit><underwri
te><![CDATA[]]></underwrite><voip/><otherphone/><addr><![CDATA[]]></addr><zip/><
seat><![CDATA[]]></seat></bean><bean><staffid>11</staffid><name><![CDATA[isv1000
3]]></name><staffaccount>isv10003</staffaccount><staffno/><sex>3</sex><mobile/><
homephone/><fax/><email/><bindno>2003</bindno><shortphone/><officephone/><credit
><![CDATA[]]></credit><underwrite><![CDATA[]]></underwrite><voip/><otherphone/><
addr><![CDATA[]]></addr><zip/><seat><![CDATA[]]></seat></bean><bean><staffid>12<
/staffid><name><![CDATA[isv10004]]></name><staffaccount>isv10004</staffaccount><
staffno/><sex>3</sex><mobile/><homephone/><fax/><email/><bindno>2004</bindno><sh
ortphone/><officephone/><credit><![CDATA[]]></credit><underwrite><![CDATA[]]></u
nderwrite><voip/><otherphone/><addr><![CDATA[]]></addr><zip/><seat><![CDATA[]]><
/seat></bean><bean><staffid>13</staffid><name><![CDATA[isv10005]]></name><staffa
ccount>isv10005</staffaccount><staffno/><sex>3</sex><mobile/><homephone/><fax/><
email/><bindno>2005</bindno><shortphone/><officephone/><credit><![CDATA[]]></cre
dit><underwrite><![CDATA[]]></underwrite><voip/><otherphone/><addr><![CDATA[]]><
/addr><zip/><seat><![CDATA[]]></seat></bean></paramlist></body></message>
------------------------------------------------------------查询信息为
----------------------------------------
staffid:8
name:isv10000
staffaccount:isv10000
staffno:
mobile:
...
```

搜索个人信息代码片段

```
public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");

        //搜索个人信息
        new SearchEmployeeDemo().searchEmployee(ab.getAppid(),ab.getTag(),
ub.getStaffaccount(),"isv10001","10","1");

    }
```

依赖方法片段如下

```
    public void searchEmployee(String appid,String tag,String accounts,String
condition,String pagecount,String pagenum)
```

```
    {
        System.out.println("-----------------------------------------查询个人信息
开始-------------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为：\n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("condition:");
        sb.append(condition);
        sb.append("\n");
        sb.append("pagecount:");
        sb.append(pagecount);
        sb.append("\n");
        sb.append("pagenum:");
        sb.append(pagenum);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("condition").setText(condition);
        params.addElement("pagecount").setText(pagecount);
        params.addElement("pagenum").setText(pagenum);

        String url = URLUtils.getURL("searchEmployee");
        System.out.println("请求地址：" + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容：" + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
```

```
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");

        //UserBean ub = new UserBean();
        if ("0".equals(retCode))
        {
            String total = XmlService.getElementText(msgDoc,
"/message/body/params/total");
            String sum = XmlService.getElementText(msgDoc,
"/message/body/params/sum");
            String staffid = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffid");
            String staffaccount = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffaccount");
            String name = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/name");
            String staffno = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/staffno");
            String mobile = XmlService.getElementText(msgDoc,
"/message/body/paramlist/bean/mobile");

System.out.println("------------------------------------------------------------个人
信息为----------------------------------------");

            System.out.println("staffid:" + staffid);
            System.out.println("name:" + name);
            System.out.println("staffaccount:" + staffaccount);
            System.out.println("staffno:" + staffno);
            System.out.println("mobile:" + mobile);
            System.out.println("...");
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
------------------------------------------查询个人信息开始
------------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
condition:isv10001
pagecount:10
pagenum:1
请求地址: http://10.166.42.201/queryEnterpriseAction.do?method=searchEmployee
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><condition>isv10001</condition><pag
ecount>10</pagecount><pagenum>1</pagenum></params></body></message>
响应消息内容: <?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head><body><params><total>1</total><sum>1</sum></params><paramlis
t type="employeeList"><bean><staffid>2366</staffid><name><![CDATA[ISV使用，勿
```

```
动]]></name><staffaccount>isv10001</staffaccount><staffno/><sex>3</sex><mobile/>
<homephone/><fax/><email/><bindno>20000</bindno><shortphone/><officephone/><cred
it><![CDATA[]]></credit><underwrite><![CDATA[]]></underwrite><voip/><otherphone/
><addr><![CDATA[]]></addr><zip/><seat><![CDATA[]]></seat></bean></paramlist></bo
dy></message>
------------------------------------------------------------个人信息为
----------------------------------------
staffid:2366
name:ISV使用，勿动
staffaccount:isv10001
staffno:
mobile:
...
```

查询用户状态代码片段及运行结果见本文档 2.6.4 章节

3.3 即时消息发送会议通知

3.3.1 场景描述

小 A 预定了某个会议，需要通过企业 OA 给与会人小 C 发送会议通知，他的选择可以有四种：发送即时消息给小 C；发送短信通知小 C；直接通过 CTD 呼叫小 C；以公告的形式通知小 C。

这就分别使用了即时消息，发送短信，公告和 CTD 呼叫四个接口。本场景为以即时消息发送会议通知。

3.3.2 预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值
- 已经调用用户鉴权接口（用户已登录）

3.3.3 实现步骤

步骤 1 构造请求 XML 消息，消息定义请参考本文档 4.11.2 章节。

步骤 2 发送请求消息，并获取响应消息。

即时消息代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new IMChatDemo().imChat(ab.getAppid(),ab.getTag(),ub.getStaffaccount(),
"isv10003",RTFUtilities.createRTFText("即时消息测试"));

    }
```

依赖方法片段如下

```
    public void imChat(String appid,String tag,String accounts,String
```

```
staffaccount,String content)
    {
        System.out.println("-----------------------------------------即时消息开始
-----------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为: \n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("staffaccount:");
        sb.append(staffaccount);
        sb.append("\n");
        sb.append("content:");
        sb.append(content);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("staffaccount").setText(staffaccount);
        params.addElement("content").setText(content);

        String url = URLUtils.getURL("imChat");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");
```

```
        if ("0".equals(retCode))
        {
            System.out.println("发送成功");
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-----------------------------------------即时消息开始
-----------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
staffaccount:isv10003
content:即时消息测试
请求地址：http://10.166.42.201/imManageAction.do?method=imChat
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><staffaccount>isv10003</staffaccoun
t><content>即时消息测试</content></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head></message>
发送成功
```

3.4 通过短信发送会议通知

3.4.1 场景描述

小 A 预定了某个会议，需要通过企业 OA 给与会人小 C 发送会议通知，他的选择可以有四种：发送即时消息给小 C；发送短信通知小 C；直接通过 CTD 呼叫小 C；以公告的形式通知小 C。

这就分别使用了即时消息，发送短信，公告和 CTD 呼叫四个接口。本场景以短信的方式发送会议通知。

3.4.2 预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值
- 已经调用用户鉴权接口（用户已登录）、
- 系统具备短信发送能力

3.4.3 实现步骤

步骤 1 构造请求 XML 消息，消息定义请参考本文档 4.12.2 章节。

步骤 2 发送请求消息，并获取响应消息。

短信发送代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");

SendSMS(ab.getAppid(),ab.getTag(),ub.getStaffaccount(),"20004","18758188884","发
送短信测试","1");


    }
```

依赖方法片段如下

```
    public static void SendSMS(String appid,String tag,String accounts,String
sender,String arrrecvs,String content,String needreturn)
    {
        System.out.println("-------------------------------------------发送短信开始
----------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为: \n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("sender:");
        sb.append(sender);
        sb.append("\n");
        sb.append("arrrecvs:");
        sb.append(arrrecvs);
        sb.append("\n");
        sb.append("content:");
        sb.append(content);
        sb.append("\n");
        sb.append("needreturn:");
        sb.append(needreturn);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("sender").setText(sender);
        params.addElement("arrrecvs").setText(arrrecvs);
        params.addElement("content").setText(content);
```

```
        params.addElement("needreturn").setText(needreturn);

        String url = URLUtils.getURL("sendSms");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");


        if ("0".equals(retCode))
        {
            System.out.println("发送成功");
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-----------------------------------------发送短信开始
------------------------------------------------------
输入参数为:
appid:11
tag:8F7D094FE09E319F7D5EC9DD92B5A257
accounts:isv10004
sender:20004
arrrecvs:18758188884
content:发送短信测试
needreturn:1
请求地址: http://10.166.42.243/smsAction.do?method=sendSms
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>8F7D094FE09E319F7D5EC9DD92B5A257</tag><acco
unts>isv10004</accounts></head><body><params><sender>20004</sender><arrrecvs>187
58188884</arrrecvs><content>发送短信测试
</content><needreturn>1</needreturn></params></body></message>
响应消息内容: <?xml version="1.0" encoding="UTF-8"?>
```

```
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head></message>
发送成功
```

3.5 通过 CTD 呼叫发送会议

3.5.1 场景描述

小 A 预定了某个会议，需要通过企业 OA 给与会人小 C 发送会议通知，他的选择可以有四种：发送即时消息给小 C；发送短信通知小 C；直接通过 CTD 呼叫小 C；以公告的形式通知小 C。

这就分别使用了即时消息，发送短信，公告和 CTD 呼叫四个接口。本场景以 CTD 呼叫的方式通知。

注：CTD 是用于向某个号码发起呼叫。通过服务器接口触发 U19XX 的呼叫业务，分别拉起主叫和被叫，并建立主叫和被叫的媒体通道，最终实现主叫和被叫的语音通话

CTD 的优势：接口形式简单，不需要使用协议栈和媒体库这样复杂的 API 进行开发调测就可以实现两方通话的业务，易于集成开发和实现

3.5.2 预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值
- 已经调用用户鉴权接口（用户已登录）

3.5.3 实现步骤

步骤 1 构造请求 XML 消息，消息定义请参考本文档 4.7.2 章节。

步骤 2 发送请求消息，并获取响应消息。

CTD 呼叫代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new CTDDemo().CTD(ab.getAppid(),ab.getTag(),ub.getStaffaccount(),
"30003","30004");

    }
```

依赖方法片段如下

```
    public void CTD(String appid,String tag,String accounts,String caller,String
callee)
    {
        System.out.println("-----------------------------------------CTD开始
-----------------------------------------------------");
```

```
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为: \n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("caller:");
        sb.append(caller);
        sb.append("\n");
        sb.append("callee:");
        sb.append(callee);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("caller").setText(caller);
        params.addElement("callee").setText(callee);

        String url = URLUtils.getURL("call");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容: " + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");


        if ("0".equals(retCode))
        {
            System.out.println("发送成功");
```

```
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-----------------------------------------CTD开始
-----------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
caller:30003
callee:30004
请求地址：http://10.166.42.201/ctdAction.do?method=call
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><caller>30003</caller><callee>30004
</callee></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head></message>
发送成功
```

3.6 通过公告发送会议

3.6.1 场景描述

小 A 预定了某个会议，需要通过企业 OA 给与会人小 C 发送会议通知，他的选择可以有四种：发送即时消息给小 C；发送短信通知小 C；直接通过 CTD 呼叫小 C；以公告的形式通知小 C。

这就分别使用了即时消息，发送短信，公告和 CTD 呼叫四个接口。本场景以发送公告的方式通知。

3.6.2 预置条件

- 已经调用第三方应用鉴权接口并获取 tag 值
- 已经调用用户鉴权接口（用户已登录）

3.6.3 实现步骤

步骤 1 构造请求 XML 消息，消息定义请参考本文档 4.7.2 章节。

步骤 2 发送请求消息，并获取响应消息。

CTD 呼叫代码片段

```
    public static void main(String[] args)
    {
        AppBean ab = new AppLoginDemo().appLogin("mobile", "mobile");
        UserBean ub = new UserLoginDemo().userLogin(ab.getAppid(),ab.getTag(),
"isv10004",EncryptUtils.encode("123456", "isv10004"),"20120428000000");
        new CTDDemo().CTD(ab.getAppid(),ab.getTag(),ub.getStaffaccount(),
"30003","30004");

    }
```

依赖方法片段如下

```
    public void CTD(String appid,String tag,String accounts,String caller,String
callee)
    {
        System.out.println("-----------------------------------------CTD开始
-----------------------------------------------------");
        StringBuffer sb = new StringBuffer();
        sb.append("输入参数为: \n");
        sb.append("appid:");
        sb.append(appid);
        sb.append("\n");
        sb.append("tag:");
        sb.append(tag);
        sb.append("\n");
        sb.append("accounts:");
        sb.append(accounts);
        sb.append("\n");
        sb.append("caller:");
        sb.append(caller);
        sb.append("\n");
        sb.append("callee:");
        sb.append(callee);
        System.out.println(sb.toString());

        //按照接口要求把参数封装成xml格式
        Document document = DocumentHelper.createDocument();
        Element message = document.addElement("message");

        URLUtils.getHead(message, appid, tag, accounts);

        Element body = message.addElement("body");
        Element params = body.addElement("params");
        params.addElement("caller").setText(caller);
        params.addElement("callee").setText(callee);

        String url = URLUtils.getURL("call");
        System.out.println("请求地址: " + url);
        System.out.println("请求消息内容 :" + document.asXML());
```

```
        String msg = XmlService.httpPostRequest(document.asXML(), url);

        Document msgDoc = null;
        try
        {
            msgDoc = DocumentHelper.parseText(msg);
            System.out.println("响应消息内容：" + msgDoc.asXML());
        }
        catch (DocumentException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println("*********send querystate request failed");
        }

       //获取返回码和返回提示信息
        String retCode = XmlService.getElementText(msgDoc,
"/message/head/retcode");
        String retContext = XmlService.getElementText(msgDoc,
"/message/head/retcontext");


        if ("0".equals(retCode))
        {
            System.out.println("发送成功");
        }
        else
        {
            System.out.println(retContext);
        }
    }
```

运行结果如下

```
-----------------------------------------CTD开始
-----------------------------------------------------
输入参数为:
appid:11
tag:4590E276CF3DD30A52100242C29042FF
accounts:isv10004
caller:30003
callee:30004
请求地址：http://10.166.42.201/ctdAction.do?method=call
请求消息内容 :<?xml version="1.0" encoding="UTF-8"?>
<message><head><appid>11</appid><tag>4590E276CF3DD30A52100242C29042FF</tag><acco
unts>isv10004</accounts></head><body><params><caller>30003</caller><callee>30004
</callee></params></body></message>
响应消息内容：<?xml version="1.0" encoding="UTF-8"?>
<message><head><retcode>0</retcode><retcontext>操作成功
</retcontext></head></message>
发送成功
```

3.7 Web 呈现用户状态

3.7.1 场景描述

在企业网站的通讯录页面上，可以显示 eSpace 状态，并支持添加联系人、发起即时消息等功能。由于状态呈现功能组件是被集成在 eSpace 客户端中的，所有该功能只在装有 eSpace 客户端的 PC 机上能够正确运行。使用 Web 页面状态呈现，需要允许 IE 浏览器加载 ActiveX 控件，eSpace 状态在页面上的呈现效果见下图：

3.7.2 预置条件

- eSpace UC 服务器部署并调试完成
- eSpace UC 服务器存在与企业网站通讯录联系人同名的账户
- 客户 PC 机上已正确安装 eSpace 客户端
- 客户 PC 机 IE 浏览器安全设置允许 ActiveX 控件加载。
- 拥有企业网站的版本发布/更新权限

3.7.3 实现步骤

步骤 1 在企业网站页面上调用 Demo 中 Js 脚本的 OneSpaceStatsCtrl 方法，传入正确的帐号、电话、姓名。参数说明参考本文档 7.1 章节

步骤 2 发布修改后的企业网站版本。

3.8 同步企业通讯录

3.8.1 场景描述

企业中有员工入职或者离职，需要向企业本身的员工系统中添加或者删除员工的信息，同时可以通过 eSpace UC 服务器的添加，修改，删除账号接口，把入职或者离职的员工信息同步到 eSpace UC 服务器中，实现企业与 eSpace UC 的深度集成。

同步通讯录包括 5 个接口：
1、SIP 放号。账号的 UC 类型需要用到 UC 号码，因此可以通过 SIP 放号接口进行 UC 号码的添加操作。
2、添加账号。账号有两个类型，UC 账号和非 UC 账号。
3、删除账号。
4、修改账号。
5、查询账号。

3.8.2 预置条件

1、 添加时开通 UC

- 已配置 SOAP 鉴权用户
- U19XX 网关已部署
- BMU 中存在 UC 号码

2、 添加时不开通 UC

- 已配置 SOAP 鉴权用户

3、 删除账号

- 已配置 SOAP 鉴权用户
- BMU 中已存在需要删除的账号

4、 修改账号

- 已配置 SOAP 鉴权用户
- BMU 中已存在需要修改的账号

5、 查询账号

- 已配置 SOAP 鉴权用户

3.8.3 实现步骤

1、 添加时开通 UC

步骤 1 调用用户服务中的连续批量 SIP 放号接口，接口格式见本文档 6.2.1 章节

步骤 2 调用账号服务中的添加账号信息接口，接口格式见本文档 6.3.1 章节

步骤 3 参数 accountsXml 中<UCService>节点的值为 1，并且<UCPhone>节点中指定已存在 UC 号码。

2、 添加时不开通 UC

步骤 1 调用账号服务中的添加账号信息接口，接口格式见本文档 6.3.1 章节

步骤 2 参数 accountsXml 中<UCService>节点的值为 0

3、 删除账号

步骤 1 调用账号服务中的添加账号信息接口，接口格式见本文档 6.3.3 6.3.1 章节

4、 修改账号

步骤 1 调用账号服务中的添加账号信息接口，接口格式见本文档 6.3.2 6.3.3 章节

5、 查询账号

步骤 1 调用账号服务中的添加账号信息接口，接口格式见本文档 6.3.4 6.3.3 章节

批量 SIP 放号代码片段

```
        UserServiceService user = new UserServiceServiceLocator();
        //注册请求地址
        String url = "http://10.166.42.243/services/UserService";
        UserService_PortType userService =
            user.getUserService(new URL(url));

        //sip放号XML
        String addXML = "<accounts>" +
            "<sipues>" +
            "<isJointUser>0</isJointUser>" +
            "<eid>1000</eid>" +
            "<eidStep>1</eidStep>" +
            "<dn>1000</dn>" +
            "<dnStep>1</dnStep>" +
            "<eidCount>5</eidCount>" +
            "<terType>2</terType>" +
            "<authType>0</authType>" +
            "<authByIp></authByIp>" +
            "<authByPwd></authByPwd>" +
            "<userPriorLevel>0</userPriorLevel>" +
            "<autoAddPrefix>1</autoAddPrefix>" +
            "</sipues>" +
            "</accounts>";
        String result = "";

        //请求SIP放号接口
        result = userService.addBatchAccounts("admin", "huawei", "000000", addXML);
        System.out.println("操作结果为：");
        System.out.println(result);
```

运行结果如下

```
        ----------------批量SIP放号开始------------------
输入的参数为:
peerName:admin
authPwd:huawei
comCode:000000
accountsXml:<accounts><sipues><isJointUser>0</isJointUser><eid>1000</eid><eidSte
p>1</eidStep><dn>1000</dn><dnStep>1</dnStep><eidCount>5</eidCount><terType>2</te
rType><authType>0</authType><authByIp></authByIp><authByPwd></authByPwd><userPri
```

```
orLevel>0</userPriorLevel><autoAddPrefix>1</autoAddPrefix></sipues></accounts>
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.4
Host: 10.166.42.243
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1679
操作结果为:
<result><retCode>0</retCode><retMsg>Add success
all.</retMsg><retContent><sipues><sipue><dn>1000</dn><eid>1000</eid><ip>127.0.0.
1</ip><authByIp></authByIp><authByPwd></authByPwd><code>0</code><msg>Add sipue
success</msg></sipue><sipue><dn>1001</dn><eid>1001</eid><ip>127.0.0.1</ip><authB
yIp></authByIp><authByPwd></authByPwd><code>0</code><msg>Add sipue
success</msg></sipue><sipue><dn>1002</dn><eid>1002</eid><ip>127.0.0.1</ip><authB
yIp></authByIp><authByPwd></authByPwd><code>0</code><msg>Add sipue
success</msg></sipue><sipue><dn>1003</dn><eid>1003</eid><ip>127.0.0.1</ip><authB
yIp></authByIp><authByPwd></authByPwd><code>0</code><msg>Add sipue
success</msg></sipue><sipue><dn>1004</dn><eid>1004</eid><ip>127.0.0.1</ip><authB
yIp></authByIp><authByPwd></authByPwd><code>0</code><msg>Add sipue
success</msg></sipue></sipues></retContent></result>
```

添加账号代码片段

```
        AccountServiceService account = new AccountServiceServiceLocator();

        //注册请求地址
        String url = "http://10.166.42.243/services/AccountService";
        AccountService_PortType accountService =
            account.getAccountService(new URL(url));

        String[] result;

        //添加账号xml
        String accountXML = "<Accounts>" +
            "<Account>" +
            "<UserAccount>isv10006</UserAccount>" +
            "<Name>isv10007</Name>" +
            "<DomainCode>000000</DomainCode>" +
            "<RoleName>普通用户</RoleName>" +
            "<Password>123456</Password>" +
            "<SelfPhone></SelfPhone>" +
            "<RestrictPhone></RestrictPhone>" +
            "<UCService>1</UCService>" +
            "<UCPhone>1000</UCPhone>" +
            "<IsSecret>1</IsSecret>" +
            "<DepCode>00</DepCode>" +
            "<Gender>0</Gender>" +
            "<EmployId>ykf19072</EmployId>" +
            "<Duty>职务1</Duty>" +
            "<OtherPhone>1</OtherPhone>" +
            "<CellPhone>2</CellPhone>" +
            "<Fax>3</Fax>" +
            "<EMail>4@2.com</EMail>" +
            "<Personurl>5</Personurl>" +
            "<Postalcode>6</Postalcode>" +
            "<Address>7</Address>" +
```

```
            "<Office>8</Office>" +
            "</Account>" +
            "</Accounts>";

        //请求添加账号接口
        result = accountService.addAccount("admin", "huawei", "000000", accountXML);
        System.out.println("操作的结果为:");
        System.out.println(result[0] + "," + result[1]);
```

运行结果如下

```
----------------添加账号开始------------------
输入的参数为:
peerName:admin
authPwd:huawei
comCode:000000
accountsXml:<Accounts><Account><UserAccount>isv10006</UserAccount><Name>isv10007
</Name><DomainCode>000000</DomainCode><RoleName>普通用户
</RoleName><Password>123456</Password><SelfPhone></SelfPhone><RestrictPhone></Re
strictPhone><UCService>1</UCService><UCPhone>1000</UCPhone><IsSecret>1</IsSecret
><DepCode>00</DepCode><Gender>0</Gender><EmployId>ykf19072</EmployId><Duty>职务
1</Duty><OtherPhone>1</OtherPhone><CellPhone>2</CellPhone><Fax>3</Fax><EMail>4@2
.com</EMail><Personurl>5</Personurl><Postalcode>6</Postalcode><Address>7</Addres
s><Office>8</Office></Account></Accounts>
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.4
Host: 10.166.42.243
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1679
操作的结果为:
0,<Result Msg="OK."></Result>
```

查询账号代码片段

```
        AccountServiceService account = new AccountServiceServiceLocator();

        //注册请求地址
        String url = "http://10.166.42.243/services/AccountService";
        AccountService_PortType accountService =
            account.getAccountService(new URL(url));

        String[] result;
        result = accountService.queryAccounts("admin", "huawei", "000000", 1,10,
"2");
        System.out.println("操作的结果为:");
        System.out.println(result[0] + "," + result[1]);
```

运行结果如下

```
------------查询账号开始-----------
输入的参数为:
peerName:admin
authPwd:huawei
comCode:000000
```

```
pageIndex:1
pageSize:10
queryCondition:2
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.4
Host: 10.166.42.243
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1679
操作的结果为:
0,<Result Msg="Successful."><Page currentIndex="1" sumpage="1"
totalRecordNum="2"></Page><Accounts><Account><UserAccount>isv00002</UserAccount>
<Name>isv00002</Name><DomainCode>000000</DomainCode><RoleName>Normal
User</RoleName><SelfPhone>1001</SelfPhone><RestrictPhone></RestrictPhone><UCServ
ice>1</UCService><UCPhone>1001</UCPhone><DepCode>0</DepCode><Gender>-1</Gender><
EmployId></EmployId><Duty></Duty><OtherPhone></OtherPhone><CellPhone></CellPhone
><Fax></Fax><EMail></EMail><Personurl></Personurl><Postalcode></Postalcode><Addr
ess></Address><Office></Office></Account><Account><UserAccount>isv10006</UserAcc
ount><Name>isv10007</Name><DomainCode>000000</DomainCode><RoleName>Normal
User</RoleName><SelfPhone></SelfPhone><RestrictPhone></RestrictPhone><UCService>
0</UCService><UCPhone></UCPhone><DepCode>0</DepCode><Gender>0</Gender><EmployId>
ykf19072</EmployId><Duty>职务
1</Duty><OtherPhone>1</OtherPhone><CellPhone>2</CellPhone><Fax>3</Fax><EMail>4@2
.com</EMail><Personurl></Personurl><Postalcode>6</Postalcode><Address>7</Address
><Office>8</Office></Account></Accounts></Result>
```

3.9 单点登录

3.9.1 场景描述

单点登录能力包括两个场景：

场景 1：一个是 eServer 可以调用第三方接口完成用户名和密码的鉴权认证。

场景 2：另一个反向鉴权接口。客户端登录时获取 eServer 当前会话的唯一标示 TOKEN，当用户单击插件图标，插件组合一个携带 TOKEN 的 URL 访问 OA，OA 拿着这个 TOKEN 到 eSpace UC 服务器上进行鉴权，通过后直接打开 OA 界面，无需再次登录。

注：场景 1:已经支持，场景 2 目前还没实现。

3.9.2 预置条件

场景 1 的预置条件：

eSpace UC 服务器版本配套。

3.9.3 实现步骤

配置 eServer 安装目录/IMServer/cfg/imserver_cfg.xml 中以下部分

```
        <config>
```

```
        ......
        <IMServer>
            ......
            <authFlag>1</authFlag>
            <authType>0</authType>
            <authInitFile>./cfg/w3delegate_client.properties</authInitFile>
        </IMServer>
    </config>
```

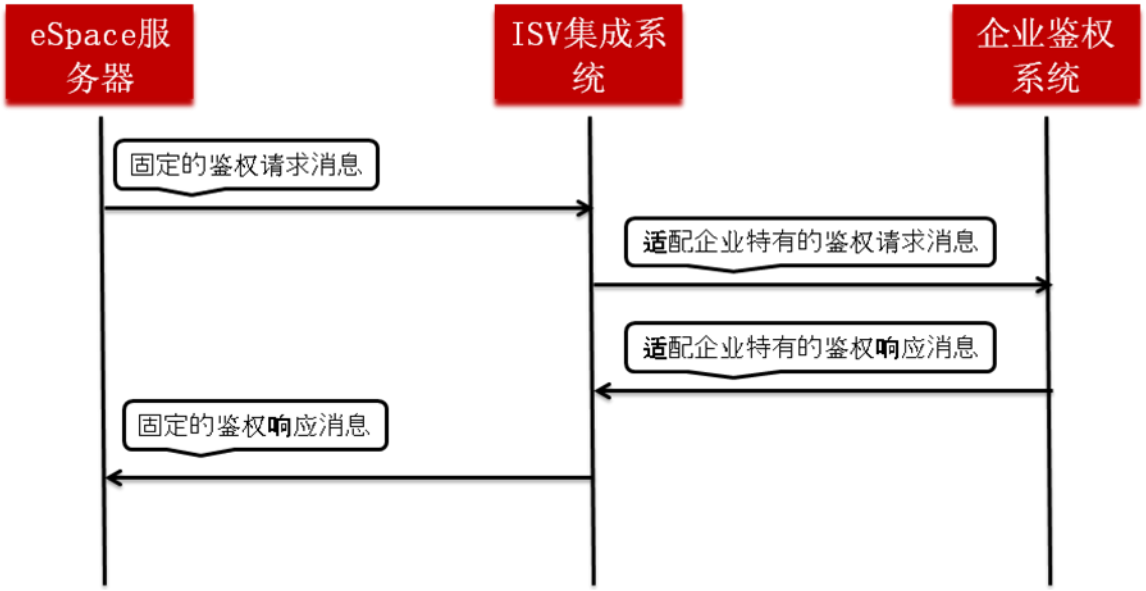
说明：authFlag：鉴权方式，0本地数据库鉴权，1第三方鉴权
authType：0
authInitFile：配置文件路径。

步骤 1 配置 w3delegate_client.properties 中鉴权地址。此文件有可能不存在，如果不存在需要 ISV 创建这个文件。文件内容为：

```
w3delegate.provider.url = http://10.166.42.122:18080/w3Delegate/login
```

注：等号右边的内容 ISV 集成实现的鉴权服务地址，这个需要按照 ISV 集成实际情况修改。
如果使用我们提供的完整 demo 作为鉴权服务器的话，URL 中要加入“auth”用于区分是鉴权(auth)还是回调（callback）。如：http://ip:port/auth

步骤 2 编写鉴权服务器代码，并实现 ISV 与企业鉴权系统的对接。流程如下图：



请参考完整 Demo 工程（AppXMLExample）中 com.huawei.espace.isv.callback.ThirdPartyAuthMessageHandler 对于 **ISV 集成系统**的简单实现。运行 com.huawei.espace.isv.test.TestCallbackServer 启动服务，鉴权地址：http://ip:port/auth

以下样例代码是处理鉴权请求消息，根据账号最后一个字符是否为偶数来判断是否鉴权成功。

```
public void onMsg(String message)
{
    try
    {
```

```
        Document doc = DocumentHelper.parseText(message);
        String username = XmlService.getElementText(doc,
"/message/body/params/username");
        String password = XmlService.getElementText(doc,
"/message/body/params/password");

        if (null == username || username.length() == 0 || null == password ||
password.length() == 0)
        {
            return;
        }

        String realAccounts = username.trim();
        realAccounts = realAccounts.substring(0, realAccounts.indexOf('@'));

        password = EncryptUtils.passwordDecode(password);

        char lastChar = realAccounts.charAt(realAccounts.length()-1);
        int i = Integer.valueOf(lastChar);
        if (i % 2 == 0)
        {
            isAuth = true;
        }

System.out.println("------------------------------------------------------------第三
方鉴权信息-----------------------------------------");
        System.out.println("username:" + realAccounts);
        System.out.println("password:" + password);
        System.out.println("result:" + isAuth);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

以下代码是根据鉴权结果来构造鉴权响应消息

```
public String getResponseMessage()
{
    String respMsg = null;
    try
    {
        Document doc = DocumentHelper.createDocument();
        Element root = doc.addElement("message");
        Element head = root.addElement("head");

        head.addElement("appid").setText("");
        head.addElement("tag").setText("");
        head.addElement("retcode").setText(isAuth ? "0" : "-1");
        head.addElement("retcontext").setText("");

        root.addElement("body");

        respMsg = doc.asXML();
    }
    catch (Exception e)
    {
```

```
        respMsg = null;
    }
    return respMsg;
}
```

3.9.4 请求消息

请求消息格式	<?xml version="1.0" encoding="“UTF-8” ?> <message> <head> <appid></appid> <eid></eid> <tag></tag> </head> <body> <params> <username>y1907901@sysdomain</username> <password><![CDATA[f9dc2827e799ee42]]></password> <usertype></usertype> <clienttype>eConsole</clienttype> </params> </body> </message> *说明：此消息为 eServer 发送给第三方服务器，请求登录鉴权的，head 中无内容，不需要处理*

参数说明	名称	说明	数据类型	长度（字节）
	username	鉴权账号,@sysdomain 对于 ISV 来说无意义，只需要保留@前的账号进行鉴权。	String	
	password	鉴权密码	String	

3.9.5 响应消息

请求消息格式	<?xml version="1.0" encoding="UTF-8" ?> <message> <head> <appid></appid> <tag></tag> <retcode>0</retcode> <retcontext></retcontext> </head> <body> </body> </message>

参数说明	名称	说明	数据类型	长度（字节）
	retcode	所有参数中，只有 retcode 有用，其他可以不填写。Retcode：操作返回码，0 为鉴权成功，其他失败。	String	

4 AppXML 调用接口

说明：AppXML 接口为第三方请求 eSpace UC 服务器的接口。

4.1 第三方应用接入鉴权接口

4.1.1 接口 URL

登录地址：http://ip:port/loginAction.do?method=appLogin

登出地址：http://ip:port/loginAction.do?method=appLogout

4.1.2 请求消息

<table>
<tr><td>请求消息格式</td><td colspan="5"><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<guid>服务唯一标识</guid>
</head>
<body>
<params>
<pwd>服务密码</pwd>
<callbackurl>http://ip:port/serviceName</callbackurl>
</params>
</body>
</message></td></tr>
<tr><td rowspan="4">参数说明</td><td>名称</td><td>说明</td><td>必填</td><td>数据类型</td><td>长度</td></tr>
<tr><td>guid</td><td>服务唯一标识</td><td>是</td><td>String</td><td>32</td></tr>
<tr><td>pwd</td><td>服务密码</td><td>是</td><td>String</td><td>64</td></tr>
<tr><td>callbackurl</td><td>注册回调地址</td><td>是</td><td></td><td></td></tr>
</table>

4.1.3 响应消息

<table>
<tr><td>响应消息格式</td><td colspan="4"><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<retcode>消息返回码</retcode>
<retcontext>返回消息体</retcontext>
</head>
<body>
<params>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<eid>企业 id</eid>
</params>
</body>
</message></td></tr>
<tr><td rowspan="6">参数说明</td><td>名称</td><td>说明</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>retcode</td><td>返回消息码，0 代表成功</td><td>String</td><td>--</td></tr>
<tr><td>retcontext</td><td>返回消息体，记录成功消息或者失败具体信息</td><td>String</td><td>--</td></tr>
<tr><td>appid</td><td>第三方服务 id</td><td>Long</td><td>10</td></tr>
<tr><td>tag</td><td>第三方服务鉴权字串</td><td>String</td><td>--</td></tr>
<tr><td>eid</td><td>企业 id</td><td>Long</td><td>10</td></tr>
</table>

4.2 用户登录接口

4.2.1 接口 URL

登录地址：http://ip:port/ /loginAction.do?method=userLogin

登出地址：http://ip:port/ /loginAction.do?method=userLogout

4.2.2 请求消息

<table>
<tr><td>请求消息格式</td><td><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>
</head>
<body></td></tr>
</table>

<table>
<tr><td></td><td colspan="5"><params>
<password></password>
<timestamp></timestamp>
</params>
</body>
</message></td></tr>
<tr><td rowspan="3">参数说明</td><td>名称</td><td>说明</td><td>必填</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>accounts</td><td>用户名</td><td>是</td><td>String</td><td>64</td></tr>
<tr><td>password</td><td>密码，密码需要加密传输</td><td>是</td><td>String(BASE64 加密，加解密指导参考本文档 8.4 附录 D:加密)</td><td>--</td></tr>
<tr><td></td><td>timestamp</td><td>客户端个人通讯录最新修改时间，时间格式：yyyymmddhhmmss</td><td>是</td><td>String</td><td>--</td></tr>
</table>

4.2.3 响应消息

<table>
<tr><td>响应消息格式</td><td><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<retcode>消息返回码</retcode>
<retcontext>返回消息体</retcontext>
</head>
<body>
<params>
<bindno>13445543443</bindno>
<funcid>00001001001001001001010010010000000000001</funcid>
<ucpin></ucpin>
<maxconfnum></maxconfnum>
<maxsmsnum></maxsmsnum>
<isupdate></isupdate>
<serverip></serverip>
<serverport></serverport>
<serverdomain></serverdomain></td></tr>
</table>

<table>
<tr><td></td><td colspan="4"><countrycode></countrycode>
<useragent></useragent>
<voipunm></voipunm>
<voippin></voippin>
<outgoingacccode></outgoingacccode>
</params>
</body>
</message></td></tr>
<tr><td rowspan="15">参数说明</td><td>名称</td><td>说明</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>retcode</td><td>返回码（详细说明见下一节）</td><td>String</td><td></td></tr>
<tr><td>retcontext</td><td>返回消息体，记录成功消息或者失败具体信息</td><td>String</td><td></td></tr>
<tr><td>bindno</td><td>绑定号码</td><td>String</td><td></td></tr>
<tr><td>funcid</td><td>功能位</td><td>String</td><td></td></tr>
<tr><td>ucpin</td><td>语音功能密码</td><td>String</td><td></td></tr>
<tr><td>maxconfnum</td><td>最大会议与会人数量</td><td>String</td><td></td></tr>
<tr><td>maxsmsnum</td><td>最大短信接收方数量</td><td>String</td><td></td></tr>
<tr><td>isupdate</td><td>是否更新个人通讯录：
0 表示不更新；
1 表示更新</td><td>String</td><td></td></tr>
<tr><td>serverip</td><td>sip 注册的软交换地址</td><td>String</td><td></td></tr>
<tr><td>serverport</td><td>sip 注册的软交换端口</td><td>String</td><td></td></tr>
<tr><td>serverdomain</td><td>sip 注册的域名</td><td>String</td><td></td></tr>
<tr><td>countrycode</td><td>sip 国家码</td><td>String</td><td></td></tr>
<tr><td>useragent</td><td>客户端类型</td><td>String</td><td></td></tr>
<tr><td>voipunm</td><td>软帐号</td><td>String</td><td></td></tr>
<tr><td>voippin</td><td>软密码</td><td>String</td><td></td></tr>
<tr><td>outgoingacccode</td><td>去话接入码</td><td>String</td><td></td></tr>
</table>

4.2.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释
- 下表为本接口特有错误代码及解决方法

错误码	说明	解决方法
93018	用户登录失败（LICENSE 过期/目前系统中 UC or PHONECLIENT 数量超过 LICENSE 许可/当前用户没有手机客户端权限）	上传合法 LICENSE 至 BMU。 给当前用户增加手机客户端权限或者更换具有手机客户端权限的用户登录。
93214	用户不存在	更换用户
93227	用户账号被暂停	更换用户或者更改用户状态
93219	用户账号被锁定	更换用户或者更改用户状态

4.3 查询用户好友分组列表

4.3.1 接口 URL

接口地址：http://ip:port/ /queryGroupAction.do?method=queryGroup

4.3.2 请求消息

请求消息格式

```
<?xml version="1.0" encoding=“UTF-8” ?>
<message>
 <head>
  <appid>服务 id</appid>
  <tag>第三方服务鉴权密码</tag>
<accounts></accounts>
 </head>
 <body>

  <params>

   <pagecount>20</pagecount>
   <pagenum>1</pagenum>
  </params>
 </body>
</message>
```

参数说明

名称	说明	必填	数据类型	长度（字节）
accounts	用户账号	是	String	64

	pagecount	每页记录上限，必须是非零正整数	是	Long	
	pagenum	当前页数。必须是非零正整数	是	Long	

4.3.3 响应消息

响应消息格式	<?xml version="1.0" encoding=“UTF-8” ?> <message> <head> <retcode></retcode> <retcontext></retcontext> </head> <body> <params> <total>24</total> <sum>3</sum> </params> <paramlist type="personalGroupList"> <bean> <groupid>号码</groupid> <name>0</name> <count></count> </bean> ... </paramlist> </body> </message>				
参数说明	名称	说明	数据类型	长度（字节）	
	retcode	返回码（详细说明见下一节）	String		
	retcontext	返回消息体，记录成功消息或者失败具体信息	String		
	total	该用户个人通讯录中分组的总数	Long		
	sum	当前分页消息中分组的数量	Long		
	groupid	分组 ID	Long		
	name	分组名称	String		
	count	分组下成员的总数	Long		

4.3.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.4 查询用户好友列表

4.4.1 接口 URL

接口地址：http://ip:port/ /queryGroupAction.do?method=queryGroupMember

4.4.2 请求消息

请求消息格式	<?xml version="1.0" encoding="UTF-8" ?> <message> <head> <appid>服务 id</appid> <tag>第三方服务鉴权密码</tag> <accounts></accounts> </head> <body> <params> <groupid>1</groupid> <pagecount>20</pagecount> <pagenum>1</pagenum> </params> </body> </message>

	名称	说明	必填	数据类型	长度（字节）
参数说明	accounts	用户账号	是	String	64
	groupid	分组 ID	是	Long	
	pagecount	每页记录上限，必须是非零正整数	是	Long	
	pagenum	当前页数。必须是非零正整数	是	Long	

4.4.3 响应消息

响应消息	<?xml version="1.0" encoding="UTF-8" ?>

格式

```
<message>
 <head>
  <retcode></retcode>
  <retcontext></retcontext>
 </head>
 <body>

  <params>

   <total>24</total>
   <sum>2</sum>
  </params>
  <paramlist type="personalList">
   <bean>
    <memberid></memberid>
    <name></name>
    <staffaccount></staffaccount>
    <sex></sex>
    <mobile></mobile>
    <homephone></homephone>
    <fax></fax>
    <email></email>
<bindno></bindno>
<shortphone><shortphone>
    <officephone></officephone>
  </bean>
   ...
  </paramlist>
 </body>
</message>
```

参数说明

名称	说明	数据类型	长度（字节）
retcode	返回码（详细说明见下一节）	String	
retcontext	返回消息体，记录成功消息或者失败具体信息	String	
sum	当前分页消息中成员的数量	Long	
memberid	成员 ID	Long	
name	姓名	String	
staffaccount	用户账号	String	
sex	性别	String	
mobile	移动电话	String	
homephone	固定电话	String	

	fax	传真	String	
	email	邮件	String	
	bindno	绑定号码	String	
	shortphone	短号	String	
	officephone	办公号码	String	

4.4.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.5 查询部门及部门下员工

4.5.1 接口 URL

接口地址：http://ip:port /queryEnterpriseAction.do?method=queryEnterprise

4.5.2 请求消息

<table>
<tr><td>请求消息格式</td><td colspan="5"><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>

</head>

<body>
<params>
<departmentid></departmentid>
<pagecount>20</pagecount>
<pagenum>1</pagenum>
</params>
</body>
</message></td></tr>
<tr><td rowspan="4">参数说明</td><td>名称</td><td>说明</td><td>必填</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>accounts</td><td>用户账号</td><td>是</td><td>String</td><td></td></tr>
<tr><td>departmentid</td><td>部门 ID。-1 表示根部门</td><td>是</td><td>String</td><td></td></tr>
<tr><td>pagecount</td><td>每页记录上限，必须是非零正整数</td><td>是</td><td>Long</td><td></td></tr>
</table>

	pagenum	当前页数。必须是非零正整数	是	Long	

4.5.3 响应消息

响应消息格式	<?xml version="1.0" encoding="UTF-8" ?> <message> <head> <retcode></retcode> <retcontext></retcontext> </head> <body> <params> <total>24</total> <deptsum>2</deptsum> <employeesum>2</employeesum> <parentdept></parentdept> </params> <paramlist type="departmentList"> <bean> <deptid></deptid> <deptname></deptname> <subDeptCount></subDeptCount> <subemployeeCount></subemployeeCount> <deptcode></deptcode> </bean> ... </paramlist> <paramlist type="employeeList"> <bean> <staffid></staffid> <name></name> <staffaccount></staffaccount> <staffno></staffno> <sex></sex> <mobile></mobile> <homephone></homephone> <fax></fax> <email></email> <bindno></bindno> <shortphone></shortphone>

	<officephone></officephone> </bean> ... </paramlist> </body> </message>

参数说明	名称	说明	数据类型	长度（字节）
	retcode	返回码（详细说明见下一节）	String	
	retcontext	返回消息体，记录成功消息或者失败具体信息	String	
	deptsum	当前分页中部门的数量	Long	
	employeesum	当前分页中员工的数量	Long	
	parentdept	上一级部门 ID	Long	
	deptid	部门 ID	Long	
	deptname	部门名称	String	
	subDeptCount	子部门的个数	String	
	subemployeeCount	子部门中员工的数量	Long	
	deptcode	部门编码	String	
	staffid	员工 ID	String	
	name	姓名	String	
	staffaccount	espace 号码	String	
	staffno	员工号	String	
	sex	性别	String	
	mobile	手机	String	
	homephone	固定电话	String	
	fax	传真	String	
	email	邮件	String	
	bindno	绑定号码	String	
	shortphone	短号	String	
	officephone	办公号码	String	

4.5.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释
- 下表为本接口特有错误代码及解决方法

错误码	说明	解决方法

93218	部门不存在	更换请求参数中的部门 ID

4.6 搜索企业通讯录

4.6.1 接口 URL

接口地址：http://ip:port/queryEnterpriseAction.do?method=searchEmployee

4.6.2 请求消息

请求消息格式	<?xml version="1.0" encoding="“UTF-8” ?> <message> <head> <appid>服务 id</appid> <tag>第三方服务鉴权密码</tag> <accounts></accounts> </head> <body> <params> <condition></condition> <pagecount>20</pagecount> <pagenum>1</pagenum> </params> </body> </message>

参数说明	名称	说明	必填	数据类型	长度（字节）
	accounts	用户账号	是	String	
	condition	查询条件	是	String	
	pagecount	每页记录上限，必须是非零正整数	是	Long	
	pagenum	当前页数。必须是非零正整数	是	Long	

4.6.3 响应消息

响应消息格式	<?xml version="1.0" encoding="“UTF-8” ?> <message> <head> <retcode></retcode> <retcontext></retcontext>

```
</head>

<body>
  <params>
    <total>100</total>
    <sum>20</sum>
  </params>
  <paramlist type="employeeList">
    <bean>
      <staffid></staffid>
      <name></name>
      <staffaccount></staffaccount>
      <staffno></staffno>
      <sex></sex>
      <mobile></mobile>
      <homephone></homephone>
      <fax></fax>
      <email></email>
<bindno></bindno>
<shortphone></shortphone>
<officephone></officephone>
<credit></ credit >
<underwrite></ underwrite>
<voip></ voip>
<otherphone></otherphone>
<addr></addr>
<zip></zip>
      <sear></seat>
    </bean>
    ...
  </paramlist>
 </body>
</message>
```

参数说明

名称	说明	数据类型	长度（字节）
retcode	返回码（详细说明见下一节）	String	
retcontext	返回消息体，记录成功消息或者失败具体信息	String	
sum	当前分页中员工的数量	Long	
staffid	员工 ID	String	
name	姓名	String	
staffaccount	espace 号码	String	

	staffno	员工号	String	
	sex	性别	String	
	mobile	手机	String	
	homephone	固定电话	String	
	fax	传真	String	
	email	邮件	String	
	bindno	绑定号码	String	
	shortphone	短号	String	
	officephone	办公号码	String	
	credit	部门	String	
	underwrite	签名	String	
	voip		String	
	otherphone	其他号码	String	
	addr	地址	String	
	zip	邮编	String	
	sear	座位号	String	

4.6.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.7 使用 CTD 呼叫

4.7.1 接口 URL

接口地址：http:// ip:port/ctdAction.do?method=call

4.7.2 请求消息

请求消息格式	<?xml version="1.0" encoding="“UTF-8” ?> <message> <head> <appid>服务 id</appid> <tag>第三方服务鉴权密码</tag> <accounts></accounts> </head>

	<body> <params> <caller>主叫号码</caller > <callee>被叫号码</callee > </params> </body> </message>

参数说明	名称	说明	必填	数据类型	长度（字节）
	appid	第三方服务 id	是	Long	10
	tag	第三方服务鉴权密码，由服务唯一标示（guid）+服务密码+服务登录时间加密生成	是	String	--
	accounts	UC 账号	是	String	--
	caller	主叫号码	是	String	32
	callee	被叫号码	是	String	32

4.7.3 响应消息

响应消息格式	<?xml version="1.0" encoding="UTF-8" ?> <message> <head> <appid>服务 id</appid> <tag>第三方服务鉴权密码</tag> <retcode>消息返回码</retcode> <retcontext>操作失败</retcontext> </head> </message>

参数说明	名称	说明	数据类型	长度（字节）
	appid	第三方服务 id	Long	10
	tag	第三方服务鉴权密码，由服务唯一标示（guid）+服务密码+服务登录时间加密生成	String	--
	retcode	返回消息码，0 代表成功。（详细说明见下一节）	String	--
	retcontext	返回消息体，记录成功消息或者失败具体信息	String	--

4.7.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.8 查询用户状态

4.8.1 接口 URL

单个查询接口地址：http://ip:port/imManageAction.do?method=getUserState

批量查询接口地址：http://ip:port/imManageAction.do?method=batchGetUserState

4.8.2 请求消息

<table>
<tr><td>请求消息格式</td><td colspan="5"><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>
</head>
<body>
<params>
<staffaccount></staffaccount>
</params>
</body>
</message></td></tr>
<tr><td rowspan="3">参数说明</td><td>名称</td><td>说明</td><td>必填</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>accounts</td><td>用户账号</td><td>是</td><td>String</td><td></td></tr>
<tr><td>staffaccount</td><td>被查询方账号。注：如果使用批量查询查询接口，被查询方账号以;(分号)分隔。</td><td>是</td><td>String</td><td></td></tr>
</table>

4.8.3 响应消息

单个查询响应消息

响应消息格式	<?xml version="1.0" encoding="UTF-8"?> <message>

<table>
<tr><td></td><td>
<head>
<retcode>消息返回码</retcode>
<retcontext>返回消息体</retcontext>
</head>
<body>
<params>
<status></status>
<statusinfo><![CDATA[出去吃饭]]></statusinfo>
</params>
</body>
</message>
</td></tr>
<tr><td rowspan="5">参数说明</td><td>名称</td><td>说明</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>retcode</td><td>返回消息码，0 代表成功（详细说明见下一节）</td><td>String</td><td></td></tr>
<tr><td>retcontext</td><td>返回消息体，记录成功消息或者失败具体信息</td><td>String</td><td></td></tr>
<tr><td>status</td><td>在线状态：
0/离线；1/在线；2/隐身（离线）；
3/忙碌；4/离开；</td><td>Long</td><td></td></tr>
<tr><td>statusinfo</td><td>自定义状态描述，自定义状态是在上述 5 种状态情况下子状态的扩展描述，如 3 状态下扩展了电话中的忙碌子状态。</td><td>String</td><td></td></tr>
</table>

批量查询响应消息

<table>
<tr><td>响应消息格式</td><td>
<?xml version="1.0" encoding="UTF-8"?>
<message>
<head>

<retcode>消息返回码</retcode>

<retcontext>返回消息体</retcontext>
</head>
<body>
<paramlist type="employeeList">
<bean>
<staffaccount></staffaccount>
<status></status>
<statusinfo><![CDATA[出去吃饭]]></statusinfo>
</bean>
….
</ paramlist >
</td></tr>
</table>

	</body> </message>			
参数说明	名称	说明	数据类型	长度（字节）
	retcode	返回消息码，0 代表成功（详细说明见下一节）	String	
	retcontext	返回消息体，记录成功消息或者失败具体信息	String	
	staffaccount	被查询方账号	String	
	status	在线状态： 0/离线；1/在线；2/隐身（离线）； 3/忙碌；4/离开；	Long	
	statusinfo	自定义状态描述，自定义状态是在上述 5 种状态情况下子状态的扩展描述，如 3 状态下扩展了电话中的忙碌子状态。	String	

4.8.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.9 查询群组列表

4.9.1 接口 URL

接口地址：http://ip:port/imManageAction.do?method=queryIMGroup

4.9.2 请求消息

请求消息格式	<?xml version="1.0" encoding=“UTF-8” ?> <message> <head> <appid>服务 id</appid> <tag>第三方服务鉴权密码</tag> <accounts></accounts> </head> <body> <params>

	</params> </body> </message>

参数说明	名称	说明	必填	数据类型	长度（字节）
	accounts	用户账号	是	String	

4.9.3 响应消息

响应消息格式	<?xml version="1.0" encoding="UTF-8" ?> <message> <head> <retcode></retcode> <retcontext></retcontext> </head> <body> <params> <total></total> </params> <paramlist type="groupList"> <bean> <gid></gid> <name></name> <count></count> </bean> ... </paramlist> </body> </message>

参数说明	名称	说明	数据类型	长度（字节）
	total	群组总数	Long	
	gid	群组 ID	Long	
	name	群组名称	String	
	count	群成员数量	Long	

4.9.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.10 查询群组成员列表

4.10.1 接口 URL

接口地址：http://ip:port/imManageAction.do?method= queryIMGroupMember

4.10.2 请求消息

<table>
<tr><td>请求消息格式</td><td colspan="5"><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>
</head>
<body>
<params>
<gid></gid>
</params>
</body>
</message></td></tr>
<tr><td rowspan="3">参数说明</td><td>名称</td><td>说明</td><td>必填</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>accounts</td><td>用户账号</td><td>是</td><td>String</td><td>64</td></tr>
<tr><td>gid</td><td>群组 iD</td><td>是</td><td>Long</td><td></td></tr>
</table>

4.10.3 响应消息

<table>
<tr><td>响应消息格式</td><td><?xml version="1.0" encoding=“UTF-8” ?>
<message>
<head>
<retcode></retcode>
<retcontext></retcontext>
</head>
<body>
<params>
<total></total>
</params>
<paramlist type="memberList">
<bean>
<staffaccount></staffaccount>
<name></name></td></tr>
</table>

<table>
<tr><td></td><td>
<headid></headid>
<signature><![CDATA[个性签名]]></signature>
</bean>
...
</paramlist>
</body>
</message>
</td></tr>
<tr><td rowspan="7">参数说明</td><td>名称</td><td>说明</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>total</td><td>好友总数</td><td>Long</td><td></td></tr>
<tr><td>staffaccount</td><td></td><td>String</td><td></td></tr>
<tr><td>name</td><td>姓名</td><td>String</td><td></td></tr>
<tr><td>headid</td><td>头像 ID</td><td>String</td><td></td></tr>
<tr><td>signature</td><td>个性签名</td><td>String</td><td></td></tr>
</table>

4.10.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.11 发送即时消息

4.11.1 接口 URL

接口地址：http://ip:port/imManageAction.do?method=imChat

4.11.2 请求消息

<table>
<tr><td>请求消息格式</td><td>
<?xml version="1.0" encoding=“UTF-8” ?>

<message>

<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>
</head>
<body>
<params>

<staffaccount></staffaccount>

<content><![CDATA[]]></content>
</td></tr>
</table>

	</params> </body> </message>

参数说明	名称	说明	必填	数据类型	长度（字节）
	accounts	用户账号	是	String	64
	staffaccount	对方账号	是	String	64
	content	即时消息内容	是	String	

4.11.3 响应消息

响应消息格式	<?xml version="1.0" encoding="UTF-8" ?> <message> <head> <retcode></retcode> <retcontext></retcontext> </head> </message>			
参数说明	名称	说明	数据类型	长度（字节）
	retcode	返回消息码，0 代表成功（详细说明见下一节）	String	
	retcontext	返回消息体，记录成功消息或者失败具体信息	String	

4.11.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.12 发送短信

4.12.1 接口 URL

接口地址：http://ip:port/smsAction.do?method=sendSms

4.12.2 请求消息

<table>
<tr><td>请求消息格式</td><td colspan="5"><?xml version="1.0" encoding=“UTF-8” ?>

<message>
<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>
</head>
<body>
<params>
<sender>短信发送人号码</sender>
<arrrecvs>短信接收人号码</arrrecvs>
<content><![CDATA[短信内容]]></content>
<needreturn>是否回执</needreturn>
</params>
</body>
</message></td></tr>
<tr><td rowspan="5">参数说明</td><td>名称</td><td>说明</td><td>必填</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>arrrecvs</td><td>接收者号码，多个接收者时用“；”隔开</td><td>是</td><td>String</td><td>--</td></tr>
<tr><td>conText</td><td>发送的短信内容</td><td>是</td><td>String</td><td>--</td></tr>
<tr><td>sender</td><td>发送者号码</td><td>是</td><td>String</td><td>32</td></tr>
<tr><td>needreturn</td><td>是否回执，0 代表不需要，1 代表需要</td><td>是</td><td>Long</td><td>1</td></tr>
</table>

4.12.3 响应消息

<table>
<tr><td>响应消息格式</td><td><?xml version="1.0" encoding=“UTF-8” ?>

<message>
<head>
<retcode>消息返回码</retcode></td></tr>
</table>

<table>
<tr><td></td><td colspan="4"><retcontext>返回消息体</retcontext>
</head>
<body></body>
</message></td></tr>
<tr><td rowspan="3">参数说明</td><td>名称</td><td>说明</td><td>数据类型</td><td>长度（字节）</td></tr>
<tr><td>retcode</td><td>返回消息码，0 代表成功。多条记录操作时，只有全部成功时，才返回 0（详细说明见下一节）</td><td>String</td><td>--</td></tr>
<tr><td>retcontext</td><td>返回消息体，记录成功消息或者失败具体信息</td><td>String</td><td>--</td></tr>
</table>

4.12.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释

4.13 发送公告

4.13.1 接口 URL

接口地址：http://ip:port/afficheAction.do?method=sendAffiche

4.13.2 请求消息

<table>
<tr><td rowspan="2">请求消息格式</td><td><?xml version="1.0" encoding=“UTF-8” ?></td></tr>
<tr><td><message>
<head>
<appid>服务 id</appid>
<tag>第三方服务鉴权密码</tag>
<accounts></accounts>
</head>
<body>
<params>
<afficheTitle></afficheTitle>
<afficheContent> </afficheContent>
<isInstancy></isInstancy>
</params>
<paramlist>
<receiver type=”user”>isv10000</receiver>
<receiver type=”user”>isv10001</receiver></td></tr>
</table>

	… </paramlist> </body> </message>				
参数说明	名称	说明	必填	数据类型	长度（字节）
	afficheTitle	公告主题	是	String	100
	afficheContent	公告内容	是	String	700
	isInstancy	是否紧急，0 代表普通通知，1 代表紧急通知	是	String	--
	receiver	接收方，type 属性为”user”	是	String	--

4.13.3 响应消息

响应消息格式	<?xml version="1.0" encoding=“UTF-8” ?> <message> <head> <retcode></retcode> <retcontext> </retcontext> </head> <body> <result></result> <detailresult></detailresult> </body> </message>			
参数说明	名称	说明	数据类型	长度（字节）
	retcode	返回消息码，0 代表成功。	String	--
	retcontext	返回消息体，记录成功消息或者失败具体信息。	String	--
	body	Retcode 为 0，且非全部接收者发送成功时添加 body 节点。		
	result	结果：“部分接收者发送成功”或“无接收者发送成功”。	String	--
	detailresult	具体结果消息：哪些接收者未发送成功。	String	--

4.13.4 本接口错误代码及解决方法

- 通用错误代码及解决方法请参考 8.3 附录 C: 常见返回码解释
- 下表为本接口特有错误代码及解决方法

错误码	说明	解决方法

95001	公告主题为空	填写公告标题。
95002	公告内容为空	填写公告内容。
95003	通知类型为空	填写通知类型。
95004	公告主题长度超过限制	更改公告主题。
95005	公告内容长度超过限制	更改公告内容。
95006	通知类型不为 0 或 1	更改通知类型为 0 或 1。
95007	接收方数量为空	填写接收方。
95008	接收方数量超过限制	更改接收方数量。
95009	获取不到 domain	检查 BMU 中是否存在 SysDomain。
95010	数据库保存公告失败	检查数据库链接。
95011	接收方重复	更改接收方。

5 AppXML 通知接口

说明：通知接口为回调消息接口。由 eSpace UC 服务器推送给第三方提供的接口

5.1 接口 URL

由第三方提供，所有回调消息都从这一个接口通知到第三方系统。

5.2 请求消息

目前只开放状态变更和即时消息接口。

请求消息格式	<?xml version="1.0" encoding=“UTF-8” ?> <message> <head> <appid>服务 id</appid> <tag>第三方服务鉴权密码</tag> </head> <body> <params> <accounts></accounts> </params> <paramlist type="notify_status">...</paramlist> <paramlist type="notify_staffInfoModified">...</paramlist> <paramlist type="notify_friendOperate">...</paramlist> <paramlist type="notify_friendResult">...</paramlist> <paramlist type="notify_groupMemberAdd">...</paramlist> <paramlist type="notify_groupMemberRemove">...</paramlist> <paramlist type="notify_groupDrop">...</paramlist> <paramlist type="notify_imChat">...</paramlist>

	<paramlist type="notify_groupChat">...</paramlist> <paramlist type="notify_smsChat">...</paramlist> <paramlist type="notify_kickout">...</paramlist> <paramlist type="notify_heartBeat">...</paramlist> <paramlist type="notify_p2pControl">...</paramlist> <paramlist type="notify_groupMemberStatus">...</paramlist> </body> </message>

参数说明	名称	说明	数据类型	长度（字节）
	notify_status	好友状态变更通知		
	notify_staffInfoModified	好友信息变更通知		
	notify_friendOperate	好友添加删除通知		
	notify_friendResult	好友操作结果通知		
	notify_groupMemberAdd	群组成员邀请通知		
	notify_groupMemberRemove	群组成员踢出通知		
	notify_groupDrop	群组删除通知		
	notify_imChat	即时消息		
	notify_groupChat	群组消息		
	notify_smsChat	短信消息		
	notify_kickout	踢人消息通知		
	notify_heartBeat	心跳失败通知		
	notify_p2pControl	P2P 请求消息通知		
	notify_groupMemberStatus	群成员状态变更		

5.3 好友状态变更通知

5.3.1 请求消息

请求消息格式	<paramlist type="notify_status"> <bean> <staffaccount></staffaccount> <status></status> <statusinfo><![CDATA[出去吃饭]]></statusinfo> </bean> </paramlist>

参数说明	名称	说明	数据类型	长度（字节）
	staffaccount	对方账号	String	
	status	对方状态	String	
	statusinfo	对方状态信息	String	

5.4 获取即时消息

5.4.1 请求消息

请求消息格式	<paramlist type="notify_imChat"> <bean> <staffaccount></staffaccount> <content><![CDATA[]]></content> </bean> </paramlist>			
参数说明	名称	说明	数据类型	长度（字节）
	staffaccount	对方账号	String	
	content	IM 消息内容	String	

5.5 回调消息开发样例

```
    public static final String NOTIFICATION_TYPE_STATUS = "notify_status";

    public static final String NOTIFICATION_TYPE_IMCHAT = "notify_imChat";
    public void onMsg(String msg)
    {
        try
        {
            Document doc = DocumentHelper.parseText(msg);

/*解析消息中所有paramlist，map的格式为
key:"notify_status",value:<bean><staffaccount></staffaccount>
<status></status><statusinfo><![CDATA[出去吃饭]]></statusinfo></bean>*/
            Map<String, List<Node>> listMap = XmlService.parseParamlist(doc);

            String accounts = XmlService.getElementText(doc,
"/message/body/params/accounts");

            Object[] typeList = listMap.keySet().toArray();
            for (Object type : typeList)
            {
                String msgType = (String)type;
                 //按type分发消息，只对状态变更和即时消息做样例代码
                if (NOTIFICATION_TYPE_STATUS.equalsIgnoreCase(msgType))
                {
                    onStatusMsg(accounts, listMap.get(msgType));
                }
                else if (NOTIFICATION_TYPE_IMCHAT.equalsIgnoreCase(msgType))
                {
                    onIMChatMsg(accounts, listMap.get(msgType));
                }
                else
                {
                    onUnknownMsg(accounts, msgType, listMap.get(msgType));
                }
```

```
            }
        }
        catch (DocumentException e)
        {
            e.printStackTrace();
        }
    }
```

依赖方法如下

```
    private void onStatusMsg(String accounts, List<Node> beans)
    {
        try
        {
            for (Node bean : beans)
            {
                //对方账号
                String staffaccount = XmlService.getElementText(bean,
"/message/body/paramlist/bean/staffaccount");
                //对方状态
                String status = XmlService.getElementText(bean,
"/message/body/paramlist/bean/status");
                //对方状态信息
                String statusinfo = XmlService.getElementText(bean,
"/message/body/paramlist/bean/statusinfo");


System.out.println("------------------------------------------------------------好友
的状态变更-----------------------------------------");
                System.out.println("accounts:" + accounts);
                System.out.println("staffaccount:" + staffaccount);
                System.out.println("status:" + status);
                System.out.println("statusinfo:" + statusinfo);
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    private void onIMChatMsg(String accounts, List<Node> beans)
    {
        try
        {
            for (Node bean : beans)
            {
                //对方账号
                String staffaccount = XmlService.getElementText(bean,
"/message/body/paramlist/bean/staffaccount");
                //发送的消息内容
                String content = XmlService.getElementText(bean,
"/message/body/paramlist/bean/content");


System.out.println("------------------------------------------------------------即时
消息-----------------------------------------");
```

```
                System.out.println("accounts:" + accounts);
                System.out.println("staffaccount:" + staffaccount);
                System.out.println("content:" + RTFUtilities.parseRTFText(content));
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    private void onUnknownMsg(String accounts, String type, List<Node> beans)
    {
        try
        {
            for (Node bean : beans)
            {

System.out.println("-------------------------------------------------------------未实
现通知消息-----------------------------------------");
                System.out.println("accounts:" + accounts);
                System.out.println("paramlist type:" + type);
                System.out.println("bean xml:" + bean.asXML());
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
```

在运行第三方应用鉴权接口前，启动回调消息接口。然后在第三方应用鉴权接口里注册回调消息接口，如：

```
    params.addElement("callbackurl").setText("http://10.166.44.236:8080");
```

以 isv10004 账号给 isv10003 账号发即时消息为例（isv10003 要登录），运行结果如下

```
   ----------------------------------------------------------即时消息
----------------------------------------
accounts:isv10003
staffaccount:isv10004
content:即时消息测试
```

6 AppSOAP 请求接口

6.1 规范说明

6.1.1 时间表示规范

表1 时间表示规范

格式	含义
yyyy	年
mm	月
dd	日
hh	时
mm	分
ss	秒

6.1.2 数据类型规范

以下提到的数据类型，按照 JDK 中的数据类型定义解释。

整型：integer

字符串数组：String[length] “[]”内为该数组的长度，length 为该长度的具体数值，无 length 时表示长度不定！

6.1.3 SoapHeader 规范

客户端在调用 BMU 的 WebService 接口时，需要在 SoapHeader 中增加用户名（节点名为 Username）和密码（节点名为 Password），BMU 接收到 Soap 请求后，解析 SoapHeader，对用户名和密码进行校验，客户端只有鉴权通过后才能调用 WebService 接口。

注：如果使用我们提供的完整 demo 是不需要处理 SoapHeader 的，因为我们已经处理好了。上面提到的只针对使用重新发布的 Soap 接口才需要进行处理。

6.2 用户服务（UserService）

接口 URL：http://*BMUUrl*/services/UserService

6.2.1 连续批量 SIP 放号

- 说明：企业中连续批量 SIP 放号
- 接口格式

  public String addBatchAccounts(String peerName,String authPwd, String comCode,String accountsXML);
- 请求信息

服务方法	addBatchAccounts				
参数	名称	说明	是否必需	数据类型	长度（字节）
	peerName	接口使用者的名称，作鉴权使用	Y	String	
	authPwd	接口鉴权密码，作鉴权使用	Y	String	
	comCode	企业域代码（集团编号），纯数字字符串	Y	String	6
	accountsXml	开户的具体号码信息，包括单个开户或批量开户，采用 XML 格式（UTF-8），具体格式如下：	Y	String	

- accountsXml 格式

```
<accounts>
  <sipues>
      <isJointUser>0</isJointUser>
      <eid>1000</eid>
      <eidStep>1</eidStep>
      <dn>1000</dn>
      <dnStep>1000</dnStep>
      <eidCount>10</eidCount>
      <terType>2</terType>
      <authType>0</authType>
      <authByIp></authByIp>
      <authByPwd></authByPwd>
      <localgwid>65535</localgwid>
      <userPriorLevel>0</userPriorLevel>
      <autoAddPrefix>1</autoAddPrefix>
  </sipues>
</accounts>
```

参数说明	名称	说明	是否必须	数据类型	长度（字节）
	isJointUser	是否联动用户，0 为非联动用户，1 为联动用户	Y	Long	1
	eid	SIP 设备标识，1~31 位数字、字母组成的字符串，当 isJointUser 为 1 时，必须为空，当 isJointUser 为 0 时，必须填写		Long	2
	eidStep	批量放号时的设备标识间隔，1~1000	Y	Long	4
	dn	待批量分配的起始号码,即话机短号，1~16 位纯数字字符串	Y	String	16
	dnStep	批量放号时的号码间隔,1~1000	Y	Long	4
	eidCount	SIP 设备数量，即号码数量，1~1024	Y	Long	4
	terType	sipue 的设备类型： 0：普通终端 2：UC 如不填，则默认为：普通终端	N	String	1
	authType	鉴权方式 0：不鉴权 1：基于 IP 鉴权 2：基于密码鉴权 3：基于 IP 与密码鉴权 如不填，则默认为：不鉴权	N	String	1
	authByIp	如果使用 IP 鉴权，请填 IP(格式如：255.255.255.255)	N	String	
	authByPwd	如果使用密码鉴权，请填 1~31 位密码字符串	N	String	31
	localgwid	当主备统一网关都故障时,本地网关可以实现本地逃生功能。只有当 softco 版本为“SOFTCO_9500_softco”时有用。默认 65535	N	String	
	userPriorLevel	用户权限级别 0：默认用户 1：正常用户 2：高级用户 3：超级用户 如不填，则默认为：默认用户	N	String	1

	autoAddPrefix	是否自动添加字冠 0：否 1：是 如不填，则默认为：否	N	String	1

- 响应消息

数据类型	
String	以 xml 格式字符串返回： <result> <retCode>0</retCode> <retMsg>Add success all.</retMsg> <retContent> <sipues> <sipue> <dn>1000</dn> <eid>1000</eid> <ip>10.166.42.243</ip> <authByIp></authByIp> <authByPwd></authByPwd> <code>0</code> <msg>Add sipue success</msg> </sipue> … </sipues> </retContent> </result> 【注】<retContent></retContent>为放号后，产生的每个号码详细信息，包含操作结果及原因。

参数说明	名称	说明	数据类型	长度(字节)
	retCode	操作返回码，0 为成功，其他为失败	String	
	retMsg	操作返结果提示信息，与返回码对应	String	
	dn	sipue 号码	String	
	eid	sipue 的 eid	String	
	ip	sipue 所在 SoftCoIp	String	
	authByIp	如果使用 IP 鉴权	String	
	authByPwd	如果使用密码鉴权	String	
	code	该 sipue 号码开户的结果码，0 为成功，其他为失败	String	
	msg	该 sipue 号码开户的结果提示信息，与返回码对应	String	

6.2.2 批量 SIP 销号

- 说明：指定企业中单个或批量销号
- 接口格式

public String deleteAccounts(String peerName, String authPwd, String comCode, String accountsXml);

- 请求信息

服务方法	deleteAccounts				
参数	名称	说明	是否必需	数据类型	长度（字节）
	peerName	接口使用者的名称，作鉴权使用	Y	String	
	authPwd	接口鉴权密码，作鉴权使用	Y	String	
	comCode	企业域代码（集团编号），纯数字字符串	Y	String	6
	accountsXml	销户的具体号码信息，包括单个销户或批量销户，采用 XML 格式（UTF-8），具体格式如下：	Y	String	

- accountsXml 格式

```
<accounts>
  <sipues>
      <sipue>
          <dn>1000</dn>
          <eid>1000</eid>
       </sipue>
       <sipue>批量销户时，同上循环填写</sipue>
  </sipues>
</accounts>
```

参数说明	名称	说明	是否必须	数据类型	长度（字节）
	dn	sipue 对应的号码，可以使用 dn 或 eid，但都要确保唯一	Y	String	
	eid	sipue 的 eid，可以使用 dn 或 eid，但都要确保唯一	Y	String	

- 响应消息

String

```
以 xml 格式字符串返回：
<result>
  <retCode>0</retCode>
  <retMsg>Delete success all.</retMsg>
  <retContent>
    <sipues>
      <sipue>
        <dn>1000</dn>
        <eid>1000</eid>
        <code>0</code>
        <msg>delete success</msg>
      </sipue>
      …
    </sipues>
```

	</retContent> </result>

参数说明	名称	说明	数据类型	长度(字节)
	retCode	操作返回码，0 为成功，其他为失败	String	
	retMsg	操作返结果提示信息，与返回码对应	String	
	dn	sipue 号码	String	
	eid	sipue 的 eid	String	
	code	该 sipue 号码开户的结果码，0 为成功，其他为失败	String	
	msg	该 sipue 号码开户的结果提示信息，与返回码对应	String	

6.3 账号服务（AccountService）

接口 URL：http://*BMUUrl*/services/AccountService

6.3.1 添加帐号信息

- 说明：提供添加 UC 账号的功能
- 接口格式

  public String[] addAccount(String peerName, String authPwd, String comCode, String accountXml);
- 请求信息

服务方法	addAccount				
参数	名称	说明	是否必需	数据类型	长度（字节）
	peerName	接口使用者的名称，作鉴权使用，双方协商	Y	String	
	authPwd	接口鉴权密码，作鉴权使用，双方协商	Y	String	
	comCode	企业域代码（集团编号），纯数字字符串，默认填 000000	Y	String	6
	accountXml	帐号信息，采用 XML 格式（UTF-8），具体格式如下：	Y	String	

- accountsXml 格式

数据类型	
String	<Accounts> <Account> <UserAccount>isv1000</UserAccount > <Name>isv1000</Name>

```
    <DomainCode>000000</DomainCode>
    <RoleName>普通用户</RoleName>
    <Password>123456</Password >
    <SelfPhone></SelfPhone>
    <RestrictPhone></RestrictPhone >
    <UCService>1</UCService>
    <UCPhone>1000</UCPhone>
    <IsSecret>0</IsSecret>
    <DepCode>00</DepCode>
    <Gender>0</Gender>
    <EmployId>yKF19079</EmployId>
    <Duty>职务</Duty>
    <OtherPhone></OtherPhone>
    <CellPhone></CellPhone>
    <Fax></Fax>
    <EMail>uccisv@huawei.com</EMail>
    <Personurl></Personurl>
    <Postalcode></Postalcode>
    <Address></Address>
    <Office></Office>
  </Account>
  <Account>...</Account>
</Accounts>
```

参数说明

名称	说明	是否必须	数据类型	长度(字节)
UserAccount	UC 账号，只能包含字母、数字、_	Y	String	2-64
Name	姓名，不能包含 "%"、"|"	Y	String	64
DomainCode	企业域编号，默认 000000	Y	String	6
RoleName	角色名称	Y	String	
Password	账号密码，6-15 个字符	Y	String	15
SelfPhone	自助服务号码。可以选择多个自助服务号码，号码间用“;”号隔开。这些号码必须已经在 U19XX 中存在	N	String	
RestrictPhone	绑定的限呼号码。多个限呼号码使用“;”号分隔。这些限呼号码必须已经在 U19XX 中存在	N	String	
UCService	是否开通 UC 业务，0 关闭，1 开通，默认不开通	Y	Long	1
UCPhone	当 UCService 开通时，必须填写需要绑定的 UC 号码，且该 UC 号码必须已经存在 U19XX 中			
IsSecret	是否保密 0：不保密 1：保密联系号码 2：完全保密，成员不可见	N		
DepCode	部门编码，如果不填默认为根部门	N	String	

	Gender	性别，0 男，1 女，如果不填为 3 未知	N	Long	
	EmployId	工号	N	String	
	Duty	职务	N	String	
	OtherPhone	其它号码	N	String	
	CellPhone	手机	N	Long	32
	Fax	传真	N	Long	20
	Email	电子邮箱	N	String	40
	Personurl	个人主页	N	String	100
	Postalcode	邮政编码	N	Long	20
	Address	联系地址	N	String	200
	Office	办工位置	N	String	50

- 返回信息

数据类型	说明
String[0]	返回字符串数组（以下标区分）： [0]返回码，0 为成功，其他为出错码；
String[1] 的 xml 格式内容	[1]返回结果以 xml 格式字符串，部分失败时，只返回失败的账号及原因，成功的账号不返回，具体格式如下： <Result Msg：”执行结果提示信息(成功提示语或出错提示语)”> <Account UC：”失败的 UC 账号标示”, Reason：”账号添加失败原因”> <Account UC：”失败的 UC 账号标示”, Reason：”账号添加失败原因”> </Result>

- 接口常见返回码见本文档 8.3.2 章节

6.3.2 修改账号信息

- 说明：提供修改 UC 账号的功能
- 接口格式

  public String[] updateAccount(String peerName, String authPwd, String comCode, String accountXml);

- 请求信息

服务方法	addAccount				
参数	名称	说明	是否必需	数据类型	长度（字节）
	peerName	接口使用者的名称，作鉴权使用，双方协商	Y	String	
	authPwd	接口鉴权密码，作鉴权使用，双方协商	Y	String	
	comCode	企业域代码（集团编号），纯数字字符串，默认填 000000	Y	String	6

	accountXml	帐号信息，采用 XML 格式（UTF-8），具体格式如下：	Y	String	

- accountsXml 格式

数据类型	
String	<Accounts> <Account> <UserAccount>isv1000</UserAccount > <Name>isv1000</Name> <DomainCode>000000</DomainCode> <RoleName>普通用户</RoleName> <Password>123456</Password > <SelfPhone></SelfPhone> <RestrictPhone></RestrictPhone > <UCService>1</UCService> <UCPhone>1000</UCPhone> <IsSecret>0</IsSecret> <DepCode>00</DepCode> <Gender>0</Gender> <EmployId>yKF19079</EmployId> <Duty>职务</Duty> <OtherPhone></OtherPhone> <CellPhone></CellPhone> <Fax></Fax> <EMail>uccisv@huawei.com</EMail> <Personurl></Personurl> <Postalcode></Postalcode> <Address></Address> <Office></Office> </Account> <Account>...</Account> </Accounts>

参数说明

名称	说明	是否必须	数据类型	长度(字节)
UserAccount	UC 账号，只能包含字母、数字、_	Y	String	2-64
Name	姓名，不能包含 "%"、"|"	Y	String	64
DomainCode	企业域编号，默认 000000	Y	String	6
RoleName	角色名称	Y	String	
Password	账号密码，6-15 个字符	Y	String	15
SelfPhone	自助服务号码。可以选择多个自助服务号码，号码间用‘;’号隔开。这些号码必须已经在 U19XX 中存在	N	String	
RestrictPhone	绑定的限呼号码。多个限呼号码使用‘;’号分隔。这些限呼号码必须已经在 U19XX 中存在	N	String	
UCService	是否开通 UC 业务，0 关闭，1 开通，默认不开通	Y	Long	1

	UCPhone	当 UCService 开通时，必须填写需要绑定的 UC 号码，且该 UC 号码必须已经存在 U19XX 中			
	IsSecret	是否保密 0：不保密 1：保密联系号码 2：完全保密，成员不可见	N		
	DepCode	部门编码，如果不填默认为根部门	N	String	
	Gender	性别，0 男，1 女，如果不填为 3 未知	N	Long	
	EmployId	工号	N	String	
	Duty	职务	N	String	
	OtherPhone	其它号码	N	String	
	CellPhone	手机	N	Long	32
	Fax	传真	N	Long	20
	Email	电子邮箱	N	Long	40
	Personurl	个人主页	N	String	100
	Postalcode	邮政编码	N	Long	20
	Address	联系地址	N	String	200
	Office	办工位置	N	String	50

- 返回信息

数据类型	说明
String[0]	返回字符串数组（以下标区分）： [0]返回码，0 为成功，其他为出错码；
String[1] 的 xml 格式内容	[1]返回结果以 xml 格式字符串，部分失败时，只返回失败的账号及原因，成功的账号不返回，具体格式如下： <Result Msg：”执行结果提示信息(成功提示语或出错提示语)”> <Account UC：”失败的 UC 账号标示”, Reason：”账号添加失败原因”> <Account UC：”失败的 UC 账号标示”, Reason：”账号添加失败原因”> </Result>

- 接口常见返回码见本文档 8.3.2 章节

6.3.3 删除账号信息

- 说明：提供删除 UC 账号功能
- 接口格式

  public String[] deleteAccount(String peerName, String authPwd, String comCode, String userAccount, String domainCode);

- 请求信息

服务方法	deleteAccount				
参数	名称	说明	是否必	数据类型	长度（字

			需		节）
	peerName	接口使用者的名称，作鉴权使用，双方协商	Y	String	
	authPwd	接口鉴权密码，作鉴权使用，双方协商	Y	String	
	comCode	企业域代码（集团编号），纯数字字符串，默认填 000000	Y	String	6
	UserAccount	帐号 Id，只能包含字母、数字、_	Y	String	
	DomainCode	企业域编号，6 位数字	Y	String	6

- 返回信息

数据类型	说明
String[0]	返回字符串数组（以下标区分）： [0]返回码，0 为成功，其他为出错码；
String[1] 的 xml 格式内容	[1]执行结果提示信息(包括成功提示语和出错提示语)。

- 接口常见返回码见本文档 8.3.2 章节

6.3.4 查询账号信息

- 说明：提供查询 UC 账号功能
- 接口格式

  public String[] queryAccounts(String peerName, String authPwd, String comCode, int pageIndex, int pageSize,String queryCondition);
- 请求信息

服务方法	queryAccounts				
参数	名称	说明	是否必需	数据类型	长度（字节）
	peerName	接口使用者的名称，作鉴权使用，双方协商	Y	String	
	authPwd	接口鉴权密码，作鉴权使用，双方协商	Y	String	
	comCode	企业域代码（集团编号），纯数字字符串，默认填 000000	Y	String	6
	pageIndex	查询起始页，默认为 0	Y	Int	
	pageSize	每页最多返回记录数，默认为 10 条	Y	Int	
	queryCondition	模糊查询条件，不允许为空，模糊匹配字段：姓名（支持简拼、全拼）、工号、电话	Y	String	

- 响应消息

数据类型	说明
String[0]	返回字符串数组（以下标区分）： [0]返回码，0 为成功，其他为出错码；
String[1]的 xml 格式内容	[1]返回结果以 xml 格式字符串，具体格式如下： <Result Msg=”执行结果提示信息(成功提示语或出错提示语)”> <Page currentIndex=”1”, sumpage=”32”></Page> <Accounts> <Account> <UserAccount>isv1000</UserAccount > <Name>isv1000</Name> <DomainCode>000000</DomainCode> <RoleName>普通用户</RoleName> <Password>123456</Password > <SelfPhone></SelfPhone> <RestrictPhone></RestrictPhone > <UCService>1</UCService> <UCPhone>1000</UCPhone> <IsSecret>0</IsSecret> <DepCode>00</DepCode> <Gender>0</Gender> <EmployId>yKF19079</EmployId> <Duty>职务</Duty> <OtherPhone></OtherPhone> <CellPhone></CellPhone> <Fax></Fax> <EMail>uccisv@huawei.com</EMail> <Personurl></Personurl> <Postalcode></Postalcode> <Address></Address> <Office></Office> </Account> <Account>...</Account> </Accounts> </Result>

参数说明

名称	说明	数据类型	长度(字节)
currentIndex	本次查询结果在总页数中的第几页	String	
sumpage	查询结果总页数	String	
UserAccount	UC 账号，只能包含字母、数字、_	String	2-64
Name	姓名，不能包含 ″%″、″|″	String	64
DomainCode	企业域编号，默认 000000	String	6
RoleName	角色名称	String	
Password	账号密码，6-15 个字符	String	15

	SelfPhone	自助服务号码。可以选择多个自助服务号码，号码间用“;”号隔开。这些号码必须已经在 U19XX 中存在	String	
	RestrictPhone	绑定的限呼号码。多个限呼号码使用“;”号分隔。这些限呼号码必须已经在 U19XX 中存在	String	
	UCService	是否开通 UC 业务，0 关闭，1 开通，默认不开通	Long	1
	UCPhone	当 UCService 开通时，必须填写需要绑定的 UC 号码，且该 UC 号码必须已经存在 U19XX 中		
	IsSecret	是否保密 0：不保密 1：保密联系号码 2：完全保密，成员不可见		
	DepCode	部门编码，如果不填默认为根部门	String	
	Gender	性别，0 男，1 女，如果不填为 3 未知	Long	
	EmployId	工号	String	
	Duty	职务	String	
	OtherPhone	其它号码	String	
	CellPhone	手机	Long	32
	Fax	传真	Long	20
	Email	电子邮箱	Long	40
	Personurl	个人主页	String	100
	Postalcode	邮政编码	Long	20
	Address	联系地址	String	200
	Office	办工位置	String	50

7 Web 集成接口

7.1 初始化状态呈现对象接口

通过 JavaScript 初始化 eSpace 状态呈现对象，实现在 Web 页面上呈现用户状态，并支持发起即时消息、发送文件、音视频呼叫、添加为联系人的操作。

7.1.1 接口声明

function OneSpaceStatsCtrl(account, staffno, name, elem)

7.1.2 参数说明

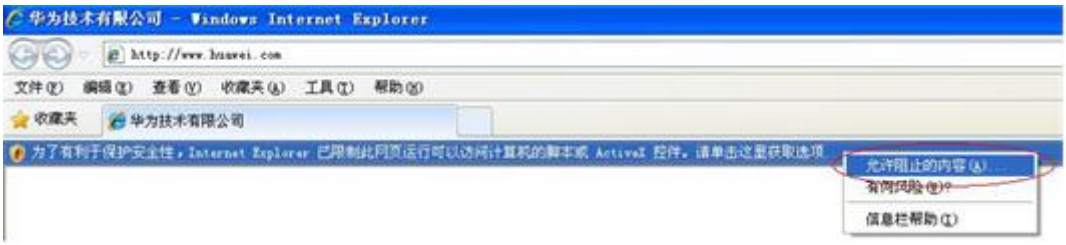
名称	说明	必填	数据类型	长度
account	要呈现状态的联系人帐号	是	String	32
staffno	联系人电话号码	是	String	32
name	联系人姓名	是	String	32
elem				

7.1.3 接口说明

该接口只支持 IE 浏览器。

接口被调用后会加载 eSpace 程序目录下的 ActiveX 控件。如果用户 PC 机没有安装 eSpace，或者在 IE 浏览器中禁用了的 ActiveX 的加载，该功能将不可用。

具体 IE 浏览器设置如下：



Domo 文件：

调用指导

首先页面中引入 js 和 css 文件

<script type="text/javascript" language="javascript" src="JavaScript/eSpaceStatus.js"></script>

<link type="text/css" href="JavaScript/Stylesheet1.css" rel="Stylesheet" />

然后在页面的控件中增加 onload 事件，调用 js 函数 OneSpaceStatsCtrl

如：

8 附录

8.1 附录 A: XML 消息的构造与解析

XML 的构造与解析有非常多的方法，也有非常多的第三方开源组件提供各种不同的 xml 构造与解析的方式，本文只挑选一个我们常用的开源组件（dom4j）来实现 xml 的构造与解析，同时我们提供一些我们常使用的一些静态方法给 ISV 开发人员参考，ISV 开发人员可以根据自己实际系统的情况来做 XML 文档的构造与解析工作。

8.1.1 XML 构造

我们以 0 章节所描述的请求消息的 xml 为例，构造如下 xml：

```
<?xml version="1.0" encoding="UTF-8" ?>
<message>
  <head>
    <guid>appAccount1</guid>
  </head>
  <body>
    <params>
      <pwd>appPassword1</pwd>
```

```
      <callbackurl>http://ip:port/serviceName</callbackurl>
    </params>
  </body>
</message>
```

实现方法如下：

```
   public static String getRequestMessage()
   {
      String reqMsg = null;
      try
      {
         Document doc = DocumentHelper.createDocument();
         Element root = doc.addElement("message");
         Element head = root.addElement("head");
         Element body = root.addElement("body");

         head.addElement("guid").setText("appAccount1");

         Element params = body.addElement("params");
         params.addElement("pwd").setText("appPassword1");
         params.addElement("callbackurl")
.setText("http://ip:port/serviceName");

         reqMsg = doc.asXML();
      }
      catch (Exception e)
      {
         reqMsg = null;
      }

      return reqMsg
    }
```

8.1.2 XML 解析

我们以 3.1.3 章节所描述的响应消息的 xml 为例，当我们完成 HTTP 消息交互后（请参考 8.2 附录 B: HTTP 协议交互举例的详细说明），我们获取如下 xml：

```
<?xml version="1.0" encoding="UTF-8" ?>
<message>
  <head>
    <retcode>0</retcode>
    <retcontext></retcontext>
  </head>
  <body>
    <params>
      <appid>app001</appid>
      <tag>jie3902i9fjajf44jfie</tag>
      <eid>eid001</eid>
    </params>
  </body>
</message>
```

我们希望获取 xml 节点中的数据，如 retcode、retcontext、appid、tag 和 eid。实现方法如下：

```
       String respMsg = getRespMessage();
       System.out.println(respMsg);
       try
       {
           Document doc = DocumentHelper.parseText(respMsg);
           String retcode = XmlService.getElementText(doc,
"/message/head/retcode");
           String retcontext = XmlService.getElementText(doc,
"/message/head/retcontext");

           String appid = XmlService.getElementText(doc,
"/message/body/params/appid");
           String tag = XmlService.getElementText(doc, "/message/body/params/tag");
           String eid = XmlService.getElementText(doc, "/message/body/params/eid");

           System.out.println(retcode);
           System.out.println(retcontext);

           System.out.println(appid);
           System.out.println(tag);
           System.out.println(eid);

       }
       catch (Exception e)
       {
           logger.error("parseMessage error: ", e);
       }
```

相关依赖方法定义如下：

```
   public static String getRespMessage()
   {
       String respMsg = null;
       try
       {
           Document doc = DocumentHelper.createDocument();
           Element root = doc.addElement("message");
           Element head = root.addElement("head");
           Element body = root.addElement("body");

           head.addElement("retcode").setText("0");
           head.addElement("retcontext").setText("");

           Element params = body.addElement("params");
           params.addElement("appid").setText("app001");
           params.addElement("tag").setText("jie3902i9fjajf44jfie");
           params.addElement("eid").setText("eid001");

           respMsg = doc.asXML();
       }
       catch (Exception e)
       {
           respMsg = null;
       }
```

```
        return respMsg;
    }

    public static String getElementText(Document doc, String singleNode)
    {
        Node node = doc.selectSingleNode(singleNode);
        if (node != null)
        {
            return node.getText();
        }
        else
        {
            return null;
        }
    }
```

8.2 附录 B: HTTP 协议交互举例

实现 HTTP 消息交互的方法很多，也有很多开源组件提供这样的功能，本文提供一个简单的方法实现 HTTP 协议同步调用方法给 ISV 开发人员作为一个参考。

8.2.1 参数说明

参数	类型	说明
msgReq	String	请求消息 XML 文本内容
url	String	eSpace UC 服务器 URL

8.2.2 返回值

调用结果	说明
成功	响应消息 XML 文本内容
失败	如果调用失败返回 null，失败原因可以通过日志查看抛出的异常

8.2.3 方法代码

```
    /** Http请求响应报文信息
     *
     * @param msgReq String
     * @param url String
     * @return String
     * @see [类、类#方法、类#成员]
     */
    public static String httpPostRequest(String msgReq, String url)
```

```
    {
        HttpURLConnection httpConnection = null;
        OutputStreamWriter out = null;
        DataInputStream dis = null;
        String data = null;

        try
        {
            URL server = new URL(url);
            httpConnection = (HttpURLConnection)server.openConnection();
            httpConnection.setRequestProperty("Content-Type", "text/xml;
charset=UTF-8");
            httpConnection.setRequestProperty("Accept",
"application/x-www-form-urlencoded");
            httpConnection.setRequestProperty("version", "100");

            httpConnection.setConnectTimeout(5000);
            httpConnection.setReadTimeout(15000);
            httpConnection.setDoInput(true);
            httpConnection.setDoOutput(true);

            out = new OutputStreamWriter(new
BufferedOutputStream(httpConnection.getOutputStream()), "UTF-8");

            out.write(msgReq);
            out.flush();

            byte[] msgBody = null;
            dis = new DataInputStream(httpConnection.getInputStream());
            int length = httpConnection.getContentLength();

            //正常设置了Content-Length的值
            if (length >= 0)
            {
                msgBody = new byte[length];
                dis.readFully(msgBody);
            }
            // 未设置值
            else
            {
                byte[] temp = new byte[1024];
                int n = 0;
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                while ((n = dis.read(temp)) != -1)
                {
                    bos.write(temp, 0, n);
                }
                msgBody = bos.toByteArray();
                bos.close();
            }
            data = new String(msgBody, "UTF-8").trim();
        }
        catch (Exception e)
        {
            data = null;
            logger.error("HTTP Post error:", e);
        }
        finally
```

```
        {
            if (null != out)
            {
                try
                {
                    out.close();
                }
                catch (IOException e)
                {
                }
                out = null;
            }

            if (null != dis)
            {
                try
                {
                    dis.close();
                }
                catch (IOException e)
                {
                }
                dis = null;
            }

            if (null != httpConnection)
            {
                try
                {
                    httpConnection.disconnect();
                }
                catch (Exception e)
                {
                }
                httpConnection = null;
            }

        }

        return data;
}
```

8.2.4 调用举例

```
        String url = "http://ip:port/loginAction.do?method=appLogin";
        String reqMsg = getRequestMessage();
        if (null != reqMsg)
        {
            String respMsg = httpPostRequest(reqMsg, url);
            if (null != respMsg)
            {
                System.out.println(reqMsg);
            }
            else
            {
```

```
            System.out.println("Error: http post failed!");
        }
    }
    else
    {
        System.out.println("Error: create request message failed!");
    }
```

8.3 附录 C: 常见返回码解释

8.3.1 http+xml 常见返回码

返回码	说明	解决方法
0	操作成功	
1	操作失败	
93001	xml 格式的消息体为空	检查请求消息格式
93002	xml 解析失败	检查请求消息格式
93003	接收的某个参数为空	检查请求消息参数
93004	接收的参数不合法	改为合法请求参数
93005	tag 生成失败	检查请求消息中 guid 与 pwd 是否正确
93006	tag 无效或匹配有误，请从新鉴权	重新请求第三方应用接入鉴权接口
93028	用户对象信息为空	重新请求用户鉴权接口
93213	用户名或密码错误	检查用户名和密码
93219	账号被锁定	更改用户状态
94001	没有可用 U19XX	检查 U19XX 连接
94002	向 U19XX 发送数据失败	检查 U19XX 连接

8.3.2 同步通讯录 SOAP 接口返回码

返回码	返回消息内容
0	OK
1	Account auth failed.
2	Account xml info is null or error.
3	There is no such domain code.
4	Address is error.
5	The employ id exists.

6	There is no such account.
7	The account exists.
8	There is no such role name.
9	UC service is turned on but phone number is not exist.
10	The gender is not set.
11	Secret is not set.
12	Email is error.
13	Duty is too long.
14	Add account error.
15	Delete account error.
16	Update account error.
17	Password is null or error.
18	Employ id is error.
19	The UC account is error.
20	The name is error.
21	The UC phone is error.
22	The UC service is not set.
23	Some self phones is not bind.
24	Some restrict phones is not bind.
25	其他错误

8.4 附录 D:加密算法

8.4.1 AppServer 用户鉴权密码加密

代码如下：

```
/**
 * 自定义加密移位算法，内部自动截断key
 * @param sourceData 用户密码
 * @param key  用户账号
 * @return
 */
public static String encryDoc(String sourceData, String key)
{
    if (key.length() > 16)
    {
        key = key.substring(0, 16);
    }
    return new BASE64Encoder().encode(encryption(sourceData, key));
}
```

相关依赖方法如下：

```
/** 加密 将Session（32位）取前16位作为Key。
```

```
     * <功能详细描述>
     * @param sourceData 用户密码
     * @param key  用户账号
     * @return
     * @see [类、类#方法、类#成员]
     */
    public static byte[] encryption(String sourceData, String key)
    {
        byte[] bKey = key.getBytes();
        byte[] bSourceData = sourceData.getBytes();
        byte[] szBuffer = new byte[bSourceData.length];

        int k = 0;
        for (int i = 0; i < bSourceData.length; i++)
        {
            szBuffer[i] = (byte)(bSourceData[i] ^ bKey[k]);
            k++;
            k = k % bKey.length;
        }

        return szBuffer;
    }
```

8.4.2 第三方鉴权密码解密

```
    public static String passwordDecode(String secret)
    {
        if (null == secret || secret.length() == 0)
        {
            return null;
        }

        String result = null;

        try
        {
            byte[] secretByte = hexstr2bytes(secret);
            byte[] kbytes = "huaweiuc_pwd".getBytes();
            SecretKeySpec key = new SecretKeySpec(kbytes, "Blowfish");
            Cipher cipher = Cipher.getInstance("Blowfish");
            cipher.init(Cipher.DECRYPT_MODE, key);
            result = new String(cipher.doFinal(secretByte));
        }
        catch (Exception ex)
        {
            result = null;
        }

        return result;
    }
```

相关依赖方法如下：

```
    private static byte[] hexstr2bytes(String _s)
    {
        byte[] bin = new byte[(_s.length() + 1) / 2];
```

```
        for (int i = 0; i < _s.length(); i += 2)
        {
            bin[i >> 1] = hexstr2byte(_s.charAt(i), _s.charAt(i + 1));
        }
        return bin;
    }

    private static byte hexstr2byte(char _c1, char _c2)
    {
        int by = hexchar2int(_c1);
        by <<= 4;
        by += hexchar2int(_c2);
        return (byte)by;
    }

    private static int hexchar2int(char _v)
    {
        return Character.digit(_v, 16);
    }
```