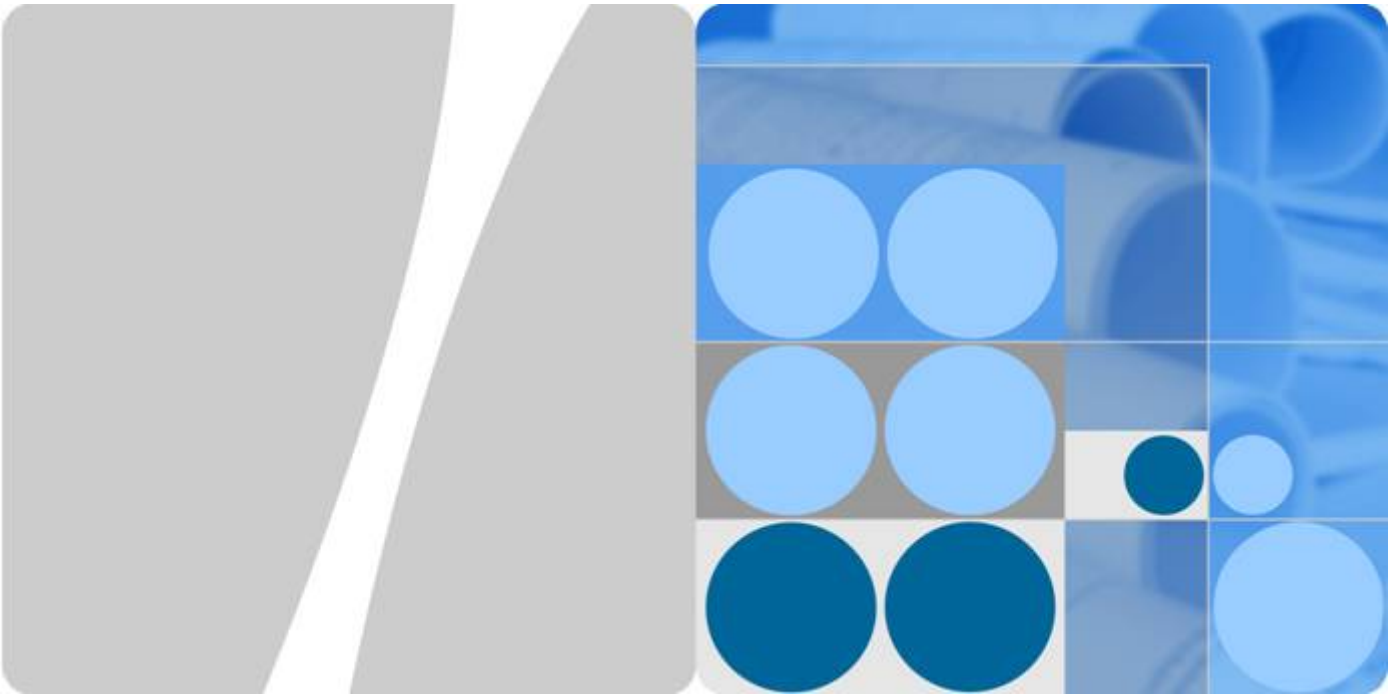


eSDK CloudEC

6.1.T2

开发指南 (REST, USM)

文档版本　01

发布日期　2018-06-08

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://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 eSDK CloudEC 简介

eSDK CloudEC 是什么？

eSDK CloudEC在第三方应用开发过程中，扮演软件中间件的角色。eSDK CloudEC为开发者在构建应用程序、组织业务系统和它的IP通信基础设施之间搭建起桥梁。eSDK CloudEC可以使商业应用程序开发人员轻松地将融合通信的能力集成到业务系统中，且无需了解过多的通信网络及基础设施的技术知识，包括这些网络是如何工作的。

eSDK CloudEC 的 Web Services 是什么？

eSDK CloudEC提供基于WEB API开发模式，开发者基于Web Services为第三方业务系统提供CloudEC能力，eSDK CloudEC提供Web Services接口，并通过内部机制尽量屏蔽CloudEC系统升级所带来的变化，同时提供Parlay X形式的接口。

eSDK CloudEC 的 Web Services 提供什么？

开发者可以使用以下几种方式调用eSDK EC的Web Services接口。

- 与应用开发环境集成，如Eclipse + CXF工具可将Web Services接口直接转换成Java接口。
- 根据WSDL定义，拼装自定义Restful消息发送至相应业务服务器即可实现融合通信功能。

通过以上处理方式，开发者可以创建业务操作（如呼叫业务），不需要关心融合通信服务的具体实现技术（如Session Initiation Protocol (SIP)技术）。所有的协议适配和消息路由均由eSDK EC和华为融合通信服务器完成。

我们提供给你的eSDK EC 包内容有以下内容：

- **eSDK CloudEC SDK包**

  eSDK CloudEC接口二次开发使用的SDK包，提供业务编排包和接口参考文档。详情请参见**SDK下载路径**。
- **Sample Codes**

  华为SDK提供一系列Sample codes演示如何调用接口，帮助您完成eSDK EC服务端接口相关业务开发。详情请参见**Sample Codes**。

2 内容导航

本开发指南主要描述eSDK CloudEC二次开放的常用功能，以及指导您如何调用eSDK CloudEC标准化接口使用这些功能。主要内容如下：

1. **相关资源**：二次开发过程中可能涉及到的软件、文档资源链接、远程实验室和技术支持。包括如何通过社区网站获取资料，Sample code下载链接，介绍如何申请远程实验室等。
2. **Hello World**：如果你仅需要尝试把SDK运行起来，您应该首先看本章节内容，它会详细说明如何下载及安装SDK以及如何配置您的开发环境。
3. **初始化配置**：在开发业务功能之前需要完成的初始化配置，这些配置动作仅需执行一次。
4. **典型场景开发**：介绍eSDK EC开放性的典型功能场景，包括开发流程、样例代码以及注意事项等。本开发指南提供以下典型场景：
5. **问题定位**：介绍开发过程中常见问题的定位方法。
6. **开发指南修订记录**：各版本开发指南更新细节。

阅读建议

- 如果想快速入门，可参考 **Hello World**章节。
- 如果想深入了解eSDK CloudEC业务的二次开发，建议您参考**典型场景开发**章节。
- 使用SDK过程中遇到问题可以参考 **问题定位**章节；或者前往华为开发者社区企业云通信版块在线FAQ查询；或联系eSDK技术支持。

3 相关资源

3.1 SDK 和文档下载路径

华为开发者社区

- 访问**华为开发者社区资源中心**，单击“通信&协作 ”，显示的页面选择“企业云通信>CloudEC”，获取SDK和文档。

华为企业产品技术支持网站

- 访问**华为企业产品技术支持网站**，单击“软件下载 > 行业解决方案”，显示的页面选择“eSDK解决方案 > eSDK EC”，获取SDK。
- 访问**华为企业产品技术支持网站**，单击“产品文档 > 行业解决方案”，显示的页面选择“eSDK解决方案 > eSDK EC”，获取文档。

3.2 Sample Codes 下载路径

访问**github**网站， 下载Sample Codes。

这里建议您使用Eclipse4.5及以上版本编译执行Sample Codes。

如果需要学习二次开发，请参考 **Hello World**。

Sample Codes 列表

Sample Code名称	描述
消息推送DEMO	消息推送Demo，通过eSDK API网关提供的RESTful接口，完成服务器级消息的对接。

Sample Code名称	描述
CTD点击回呼DEMO	点击回呼Demo，通过eSDK API网关提供的RESTful接口，完成建立双向回拨通话的过程。
业务管理集成DEMO	业务管理集成Demo，通过eSDK API网关提供的RESTful接口，完成个人及部门信息同步，开销户和绑定号码等操作。
会议管理DEMO	会议管理Demo，通过eSDK服务端提供的RESTful接口，完成对会议的管理能力（预定、查询、修改、删除和添加与会人等操作）。

3.3 免费申请远程实验室

华为 eSDK 远程实验室简介

华为eSDK 远程实验室致力于为合作伙伴提供真实的华为ICT 产品能力的远程对接联调环境，通过在线申请相应ICT 产品的测试账号与权限，您不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发，并实现远程对接测试认证。借助华为远程实验室，您可以“零”出差构建解决方案，“零”设备构建演示环境， 提高对接测试通过率，缩短产品二次开发周期。

华为eSDK远程实验室为开发者提供了7×24小时的免费云化实验室环境，提供真实的华为设备供开发者远程在线开发调试。借助远程实验室自助管理平台，开发者不需要购置华为产品便可基于远程实验室针对相关产品进行二次开发，并实现远程对接测试认证。

目前，华为远程实验室已发布Cloud Computing、Carrier Software、SDN、BYOD、Cloud EC、Agile Network、eLTE、Big Data、IoT、IES等多个生态圈的实验室环境。

相关信息请查询华为开发者社区远程实验室。

远程实验室有哪些优势

- 低门槛：官网注册用户即可申请使用，环境与预约时长、预约次数受限；
- 分级支持：环境按域划分，重点开发者、合作伙伴可访问特定环境并享受额外的预约环境；
- 全球资源高速互联：建设以苏州远程实验室为中心的全球实验室分布格局，依托IT全球100ms高性能骨干网络和IT全球端到端应用保障能力。

如何免费使用远程实验室

请访问远程实验室操作指导，查阅远程实验室预约申请及接入使用方法。

3.4 技术支持渠道

开发过程中，您有任何问题，可以通过如下途径解决：

- 在DevCenter中提单跟踪。

- 在**华为开发者论坛**中发帖求助。
- 华为技术支持热线电话：400-8828-000
- 华为技术支持邮箱：**esdk@huawei.com**

4 Hello World

4.1 Hello World 流程

本示例以Java语言进行eSDK CloudEC的二次集成开发，实现一个简单的CTD呼叫功能。它会详细说明如何下载及安装SDK，以及如何配置你的开发环境。

开发过程中的故障处理请参考**定位指南**。

快速入门流程如下：

4.2 开发环境准备

开发工具

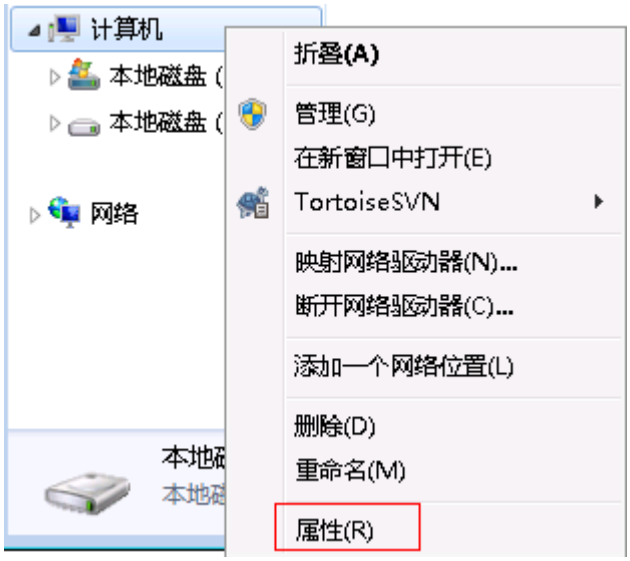
在进行二次开发之前，应该准备好以下环境和资源，如表4-1所示。

表 4-1 资源名称和要求

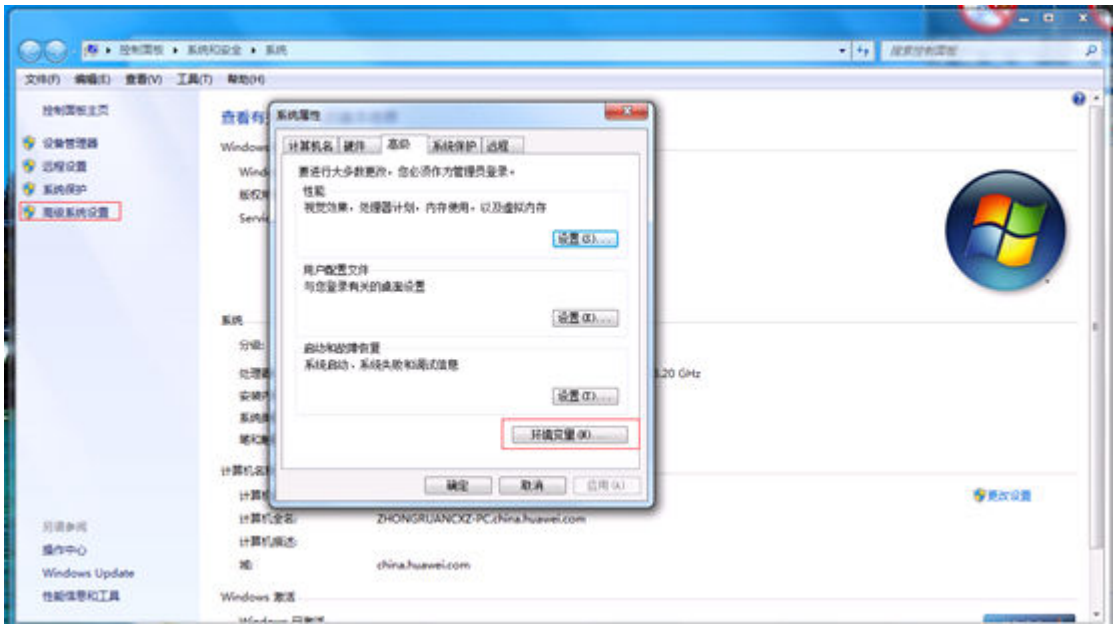
工具名称	版本要求	说明
操作系统	Windows 7专业版	NA
Eclipse	Eclipse 4.5.0及以上版本	NA
JDK	Java Development Kit 1.8.0_111及以上版本	本文档中提供的所有demo运用的均是jdk1.8.0_111版本，建议使用jdk1.8.0_111以上版本，如果需要使用更高版本，请参考如何修改JRE版本？

JDK 环境变量配置

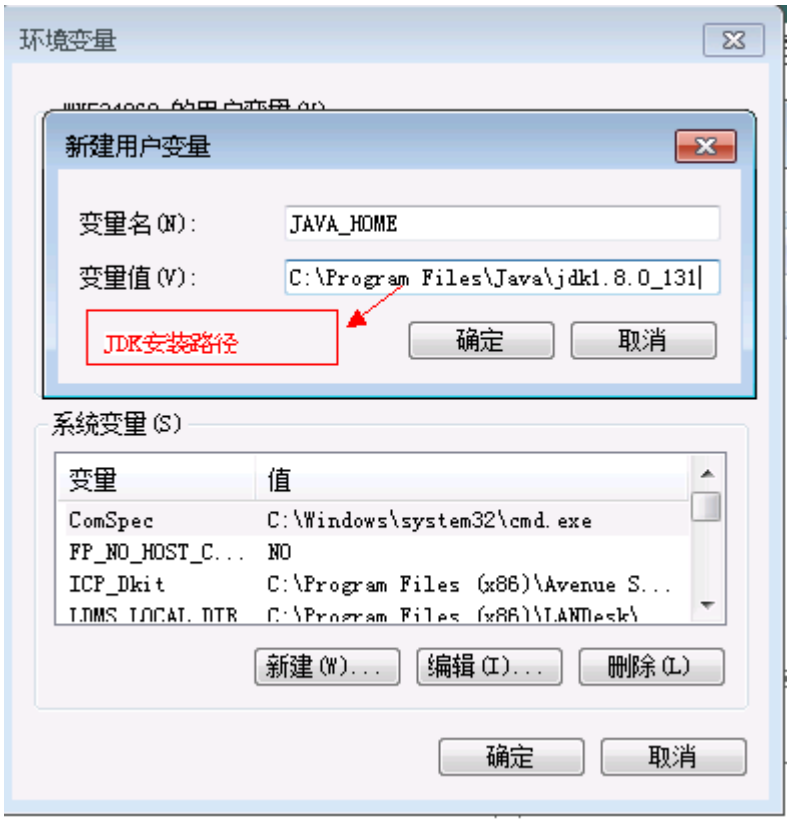
步骤1 计算机右键属性



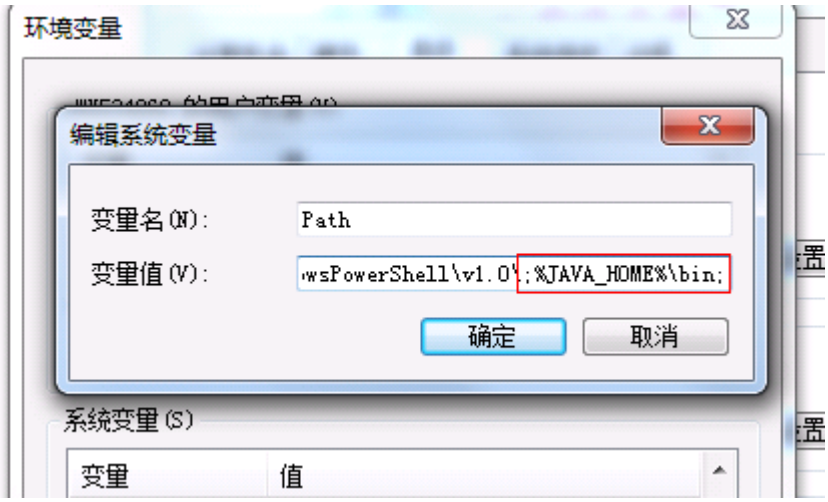
步骤2 弹出如下图，选择高级系统配置，点击环境变量



步骤3 点击确定，然后新建用户变量，变量名为JAVA_HOME，值为JDK安装路径，如下：



步骤4 将JAVA_HOME写入path中



步骤5 打开命令窗口，输入java －version验证是否配置成功

```
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

----结束

编排包和文档

编排包：esdk_apigw_ec_mediator_VX.X.X.zip、esdk_apigw_ec_sp_VX.X.X.zip、esdk_apigw_ec_VX.X.X.zip

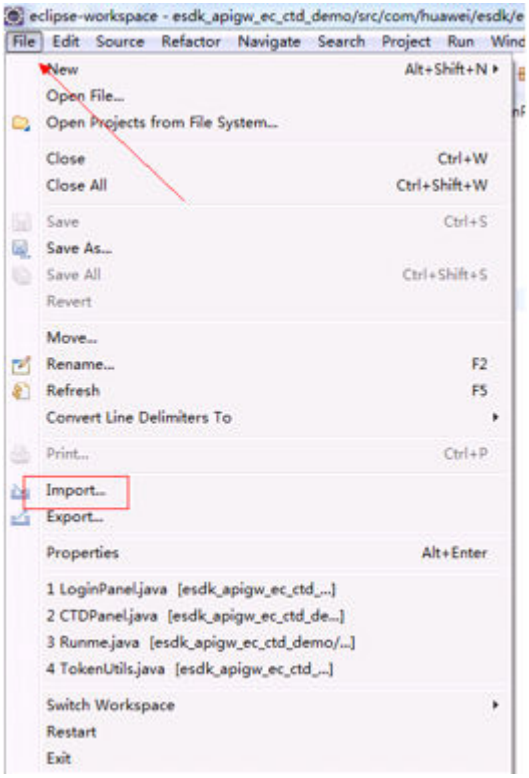
文档名称：《eSDK CloudEC X.X.X 接口参考(REST,USM)》、《eSDK CloudEC X.X.X 开发指南(REST,USM)》

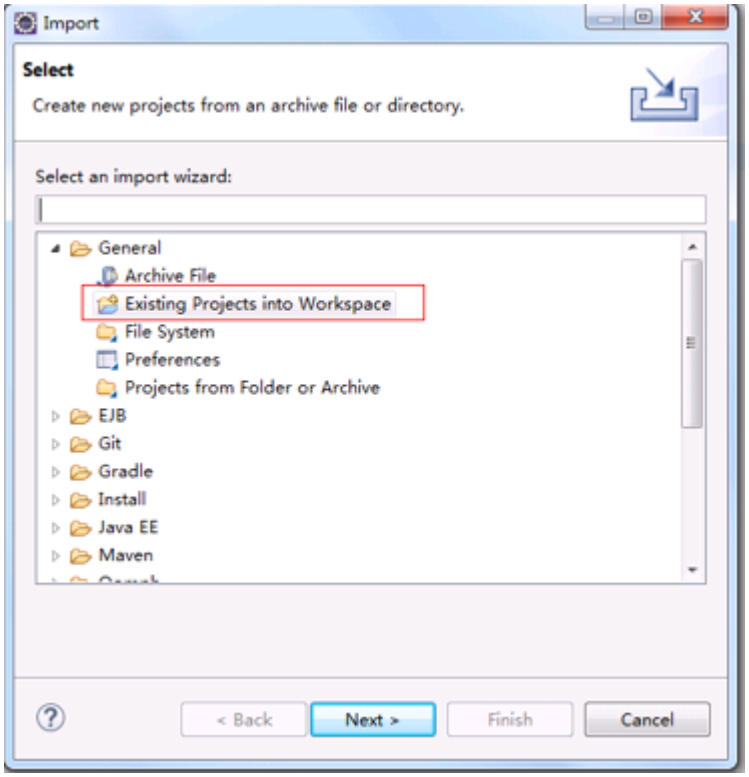
SDK和文档下载路径：参见**SDK和文档下载路径**。

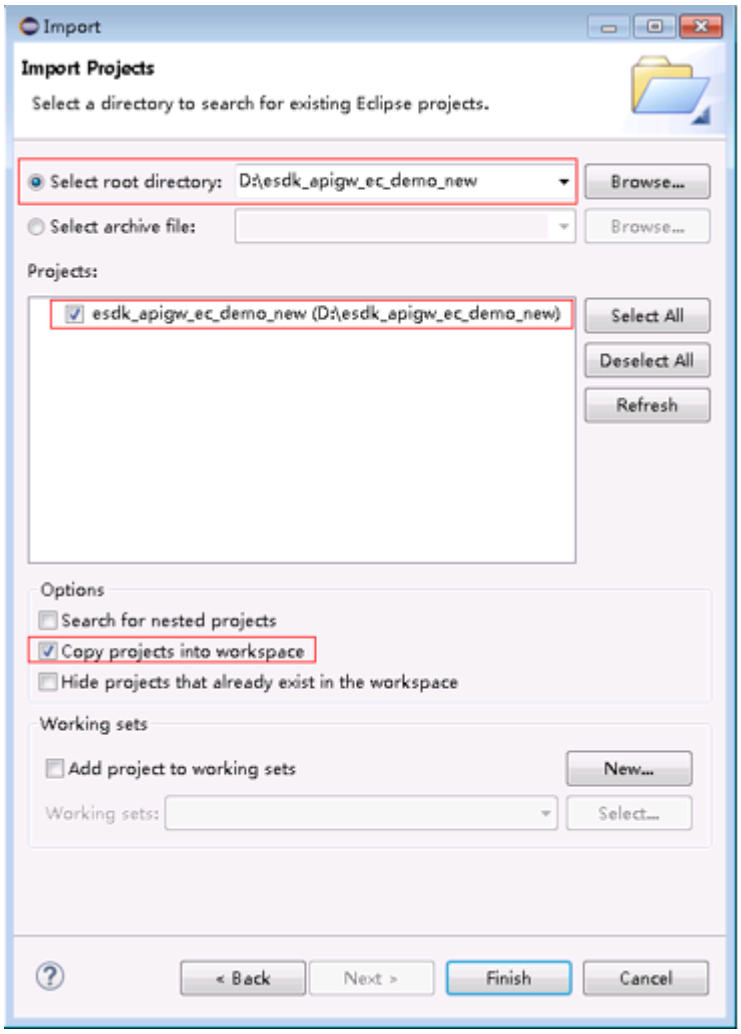
4.3 导入工程项目

步骤1 **打开eclipse，选择File** >Import >General >Existing Projects into Workspace

点击next，点击finish，如图：







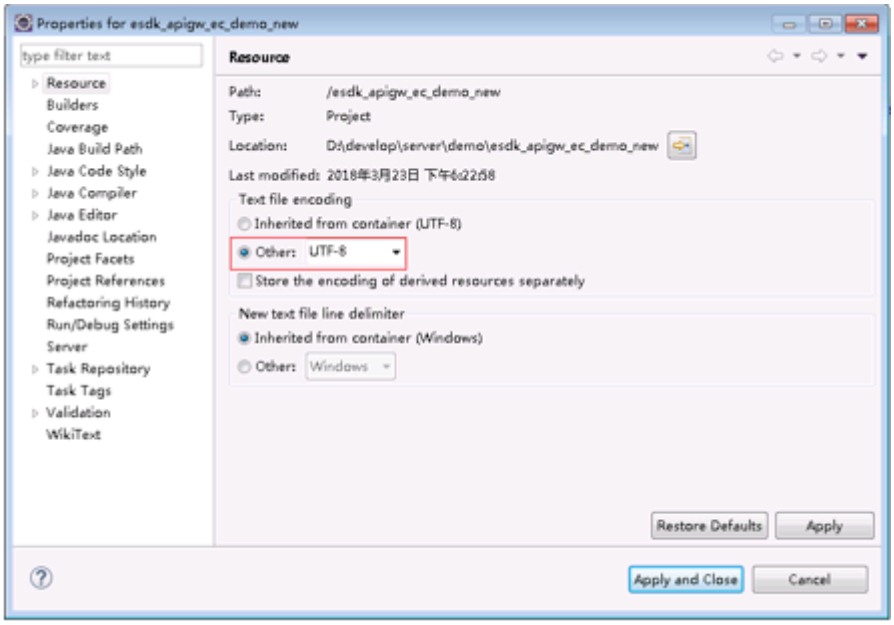
步骤2 点击finish，导入工程完毕。

----结束

4.4 设置编码格式

步骤1 右键单击“esdk_apigw_ec_demo_new”工程（以下简称Demo工程），选择“Properties”，系统打开如Properties for esdk_apigw_ec_demo_new所示窗口。

图 4-1 Properties for esdk_apigw_ec_demo_new

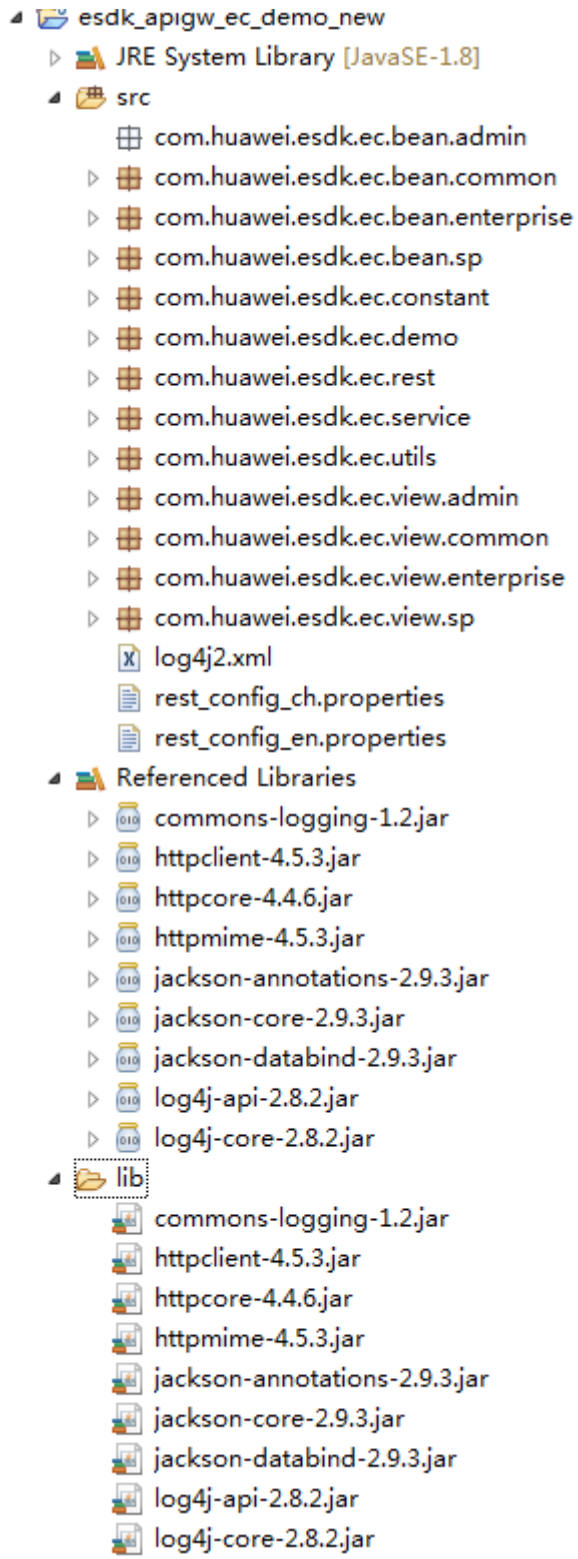


步骤2 选择“Resource”标签页，在“Text file encoding”框中选择“Other”，并在下拉框中将编码格式设置为“UTF-8”，单击“OK”，完成设置。

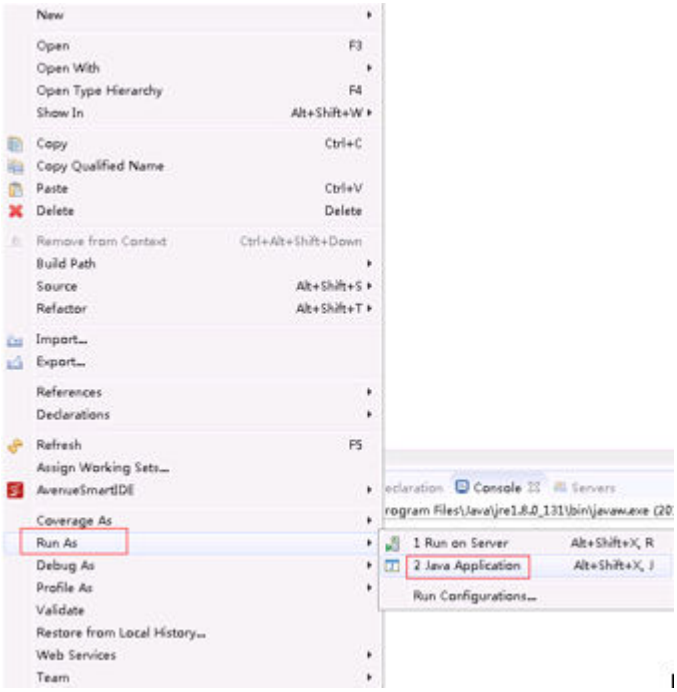
----结束

4.5 调试运行

步骤1 导入成功后，工程结构如下：

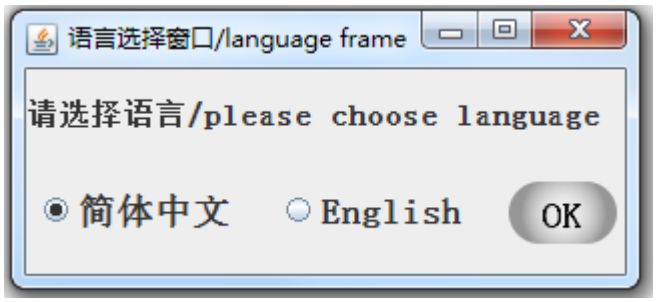


步骤2 在包名为“com.huawei.esdk.ec.demo”中，有个Runme.java,选中右键，Run As >Java Application,如图：

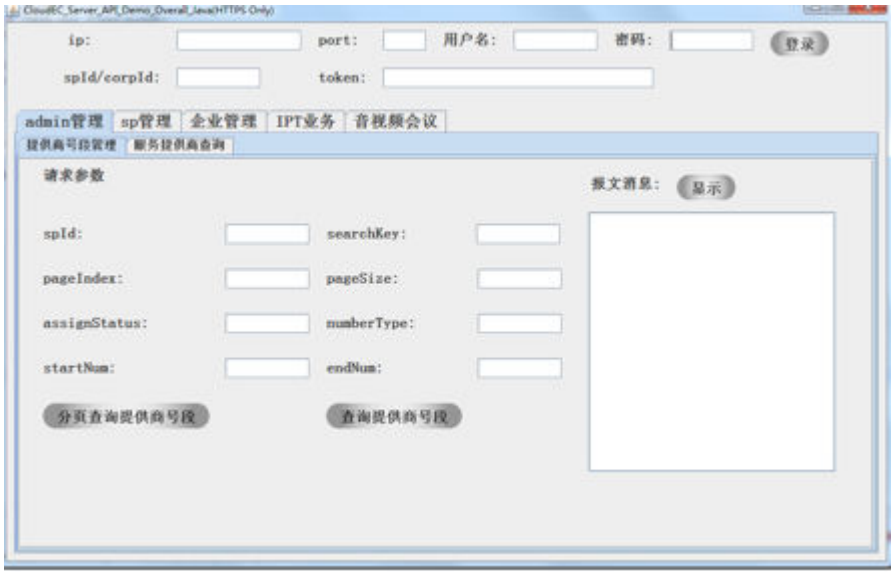


步骤3 运行完效果如下：

首先会弹出一个语言选择框



点击OK进入主界面



----结束

4.6 演示概述

1.在调用各个接口之前，先根据相对应的权限账号进行登录，方可继续执行以下接口按钮

2.音视屏会议中的基本会控接口，需要会议token，在登录之后先执行会议鉴权接口，才可以点击以下按钮调用接口

3.参数的填写可以根据eSDK CloudEC 6.1.0 接口参考 (REST, USM)文档

5 鉴权登录

注意事项：只支持https请求

参数描述：

参数名称	参数含义	取值说明	例子
ip	REST请求地址	API网关提供服务的IP地址。	https：// 127.0.0.1
port	请求端口号	API网关提供服务的端口号。	8000
用户名	用户名	向API网关鉴权的用户名。	
密码	鉴权密码	向API网关鉴权的密码。	
spId/corpId	id信息	如果是sp登录或者企业管理员登录，会返回id信息	无需填写
token	鉴权信息	服务器返回的鉴权值，无需填写	无需填写

6 业务开发

6.1 概述

6.1.1 eSDK EC 的软件架构图

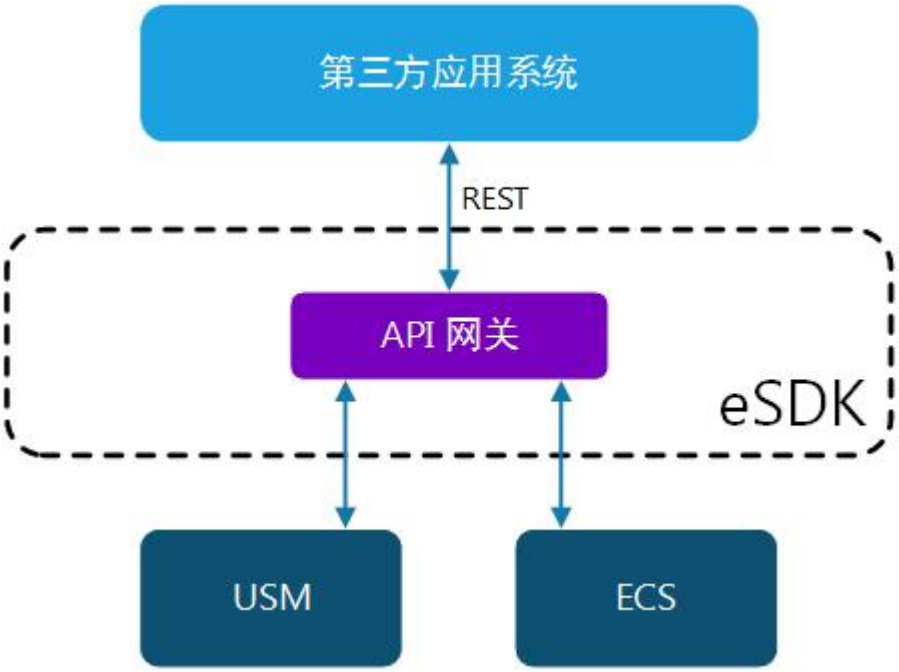


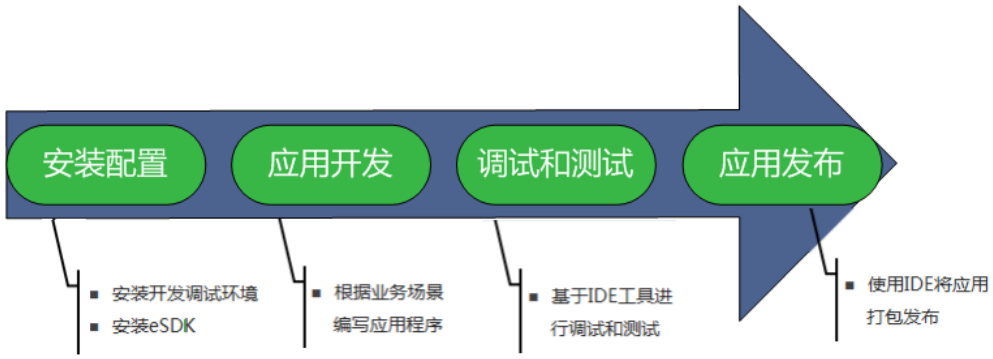
表 6-1 系统核心模块功能结构

系统模块	功能描述
API网关	以RESTful协议的形式开放接口，供第三方应用或Native调用。

6.1.2 eSDK EC 二次开发流程

基于eSDK EC进行二次开发的基本开发步骤如图6-1所示

图 6-1 基本开发步骤



开发主要由四大步骤组成：

- 安装配置
  在这个阶段，开发者需要下载和配置开发调试环境，如Eclipse；同时需要下载和安装配置eSDK EC开发资源。
- 应用开发
  在这个阶段，开发者需要建立自己的项目工程，并根据业务场景需求，参考示例代码和文档，编写应用。
- 调试和测试
  在这个阶段，开发者使用Eclipse自带的调测工具进行应用调试和功能测试。
- 应用发布
  开发者调测完应用后可以通过Eclipse将应用打包发布给最终用户使用。

6.2 消息推送

6.2.1 概述

通过服务器级的消息对接，快速完成企业现有业务系统到信息系统的改造。

传统的对接短信的方式，很难与业务系统提供的数据进行关联，员工无法方便的获取该短信的上下文信息。而使用服务器提供的即时消息接口，借助手机App和EC客户端

的能力，可以更有效沟通（比如建立话题讨论群，查看消息时关联客户的生产系统数据，获取上下文信息），方便快捷的处理业务，定位问题。

比如，生产系统将当前的业务消息（如设备发出告警信息需要通知到对应维护工程师、金融行业办理业务时授权的信息需要发送给对应的授权人等）通过eSDK API网关发送给指定的员工（指定EC账号或者群组）；企业的OA系统发送部门公告通知指定部门的员工。

6.2.2 典型场景开发

本章的开发任务：通过eSDK API网关提供的RESTful接口，完成服务器级消息的对接。同时也包含状态查询。开发者可以事先下载好PC软终端或移动端Android和iOS的App完成消息的接收。

Sample 界面

图 6-2 消息推送主界面

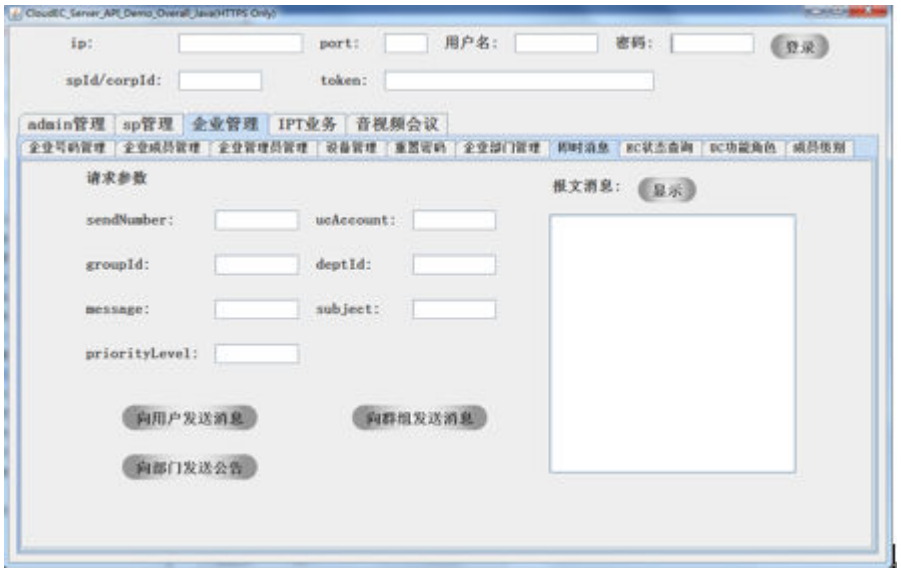
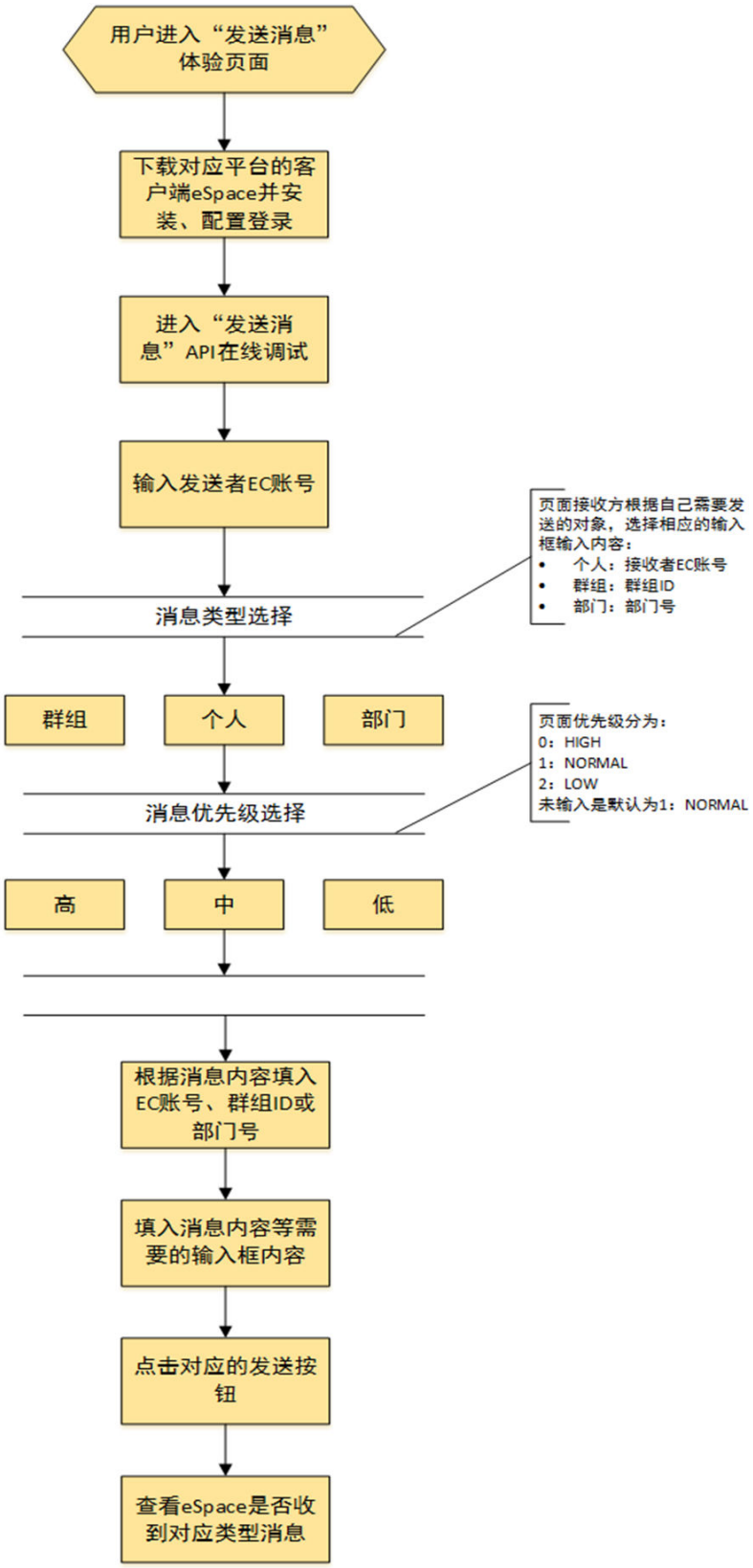


图 6-3 查看 EC 用户状态界面

流程图



前提条件

启动eSDK CloudEC服务。

示例代码

用户通过使用向EC用户发送消息、向群组发送消息、向部门发送系统通知消息实现即时消息发送接收功能。

- 向EC用户发送消息

  根据页面输入的发送者EC账号、接收者账号、消息内容、消息优先级等请求参数，传入后台的appSendMsgToUC方法中进行拼装，通过调用sendMessage方法发送Rest消息，请求eSDK服务的向EC用户发送消息接口，实现向EC用户发送消息功能。

  代码示例：

  http:///{ip}:{port}/im/ecAccount

  调用appSendMsgToUC方法拼装请求参数，发送Rest消息。

```
//java code
//向EC用户发送消息
private void appSendMsgToUC()
{
    //拼装请求参数，发送请求
    SendMsgToUC payload = new SendMsgToUC();

    //获取发送者EC账号
    payload.setSendNumber(sendNumberField.getText());
    payload.setUcAccount(ucAccountField.getText());
    payload.setMessage(addMessage());
    payload.setDateTime(getDataTime());

    if(StringUtils.isNotEmpty(priorityLevelField.getText().trim()))
    {
        payload.setPriorityLevel(priorityLevelField.getText().trim());
    }
    try
    {
        Token token = LoginUtils.getToken();

        // 利用token发送restful请求
        RestRequest request = new RestRequest(HTTPConstant.HTTP_METHOD_POST);
        request.setPayload(payload);
        ecService.post("/im/ecAccount", request, errInfoLabel, token);
    }
    catch (Exception e)
    {
        LOGGER.error("get Token error",e);
        showErrInfoWithColor("操作失败");
    }
    finally
    {
        ecService.finish();
    }
}
```

- **sendMessage方法**请求eSDK服务的对应接口代码
- **service层处理返回response**对应代码

查询 EC 用户状态

根据页面输入的操作者账号、需要查询的账号，传入后台的queryUCListPresence方法中进行拼装，通过调用sendMessae方法（**代码参考链接**）发送Rest消息，请求eSDK服务的查询EC用户状态信息接口，实现查询EC用户状态信息。

代码示例：

http:///{ip}:{port}/im/presence

调用queryUCListPresence方法拼装请求参数，发送Rest消息

```
//java code
// 查询EC用户状态
//查询EC用户状态
private void queryUCListPresence()
{
    //拼装请求参数，发送请求
    Map<String, String> param = new HashMap<String, String>();
    String[] queryUsersText = queryUsersField.getText().trim().split(",");
    StringBuffer stringBuffer = new StringBuffer();
    stringBuffer.append("[");
    for (String queryUserText : queryUsersText)
    {
        stringBuffer.append("\"").append(queryUserText).append("\"").append(",");
    }
    String substring = stringBuffer.toString().substring(0, stringBuffer.toString().length() -1) +
"]";
    param.put("queryUsers", substring);
    if (StringUtils.isNotEmpty(ECUserField.getText().trim()))
    {
        param.put("user", ECUserField.getText());
    }

    try
    {
        Token token = LoginUtils.getToken();

        // 利用token发送restful请求
        RestRequest request = new RestRequest(HTTPConstant.HTTP_METHOD_GET);
        request.setParameters(param);

        ecService.get("/im/presence", request, errInfoLabel, token);
    }
    catch (Exception e)
    {
        LOGGER.error("get Token error",e);
        showErrInfoWithColor("操作失败");
    }
    finally
    {
        ecService.finish();
    }
}
```

- **sendMessage方法**请求eSDK服务的对应接口代码
- **service层处理返回response**对应代码

6.3 业务发放管理

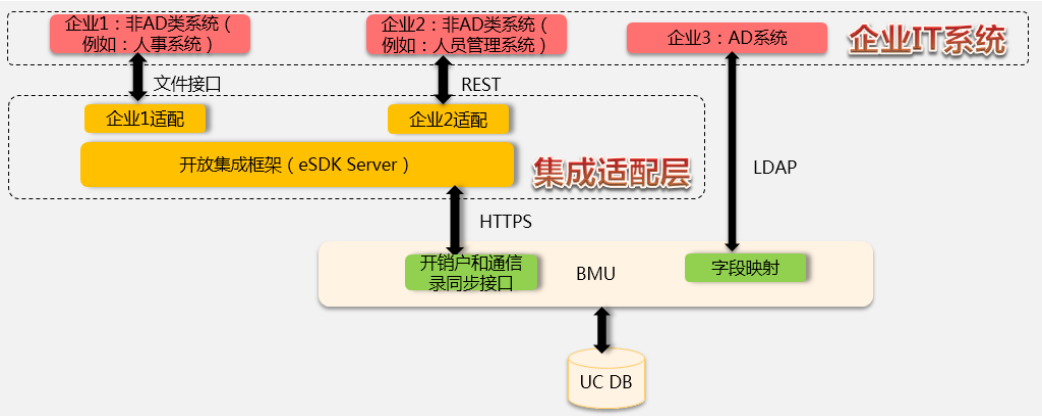
6.3.1 概述

随着企业的发展，业务系统的数量在不断的增加，老的系统却不能轻易的替换，这会带来很多的开销，其一是管理上的开销。需要维护的系统越来越多，很多系统的数据是相互冗余和重复的，数据的不一致性会给管理工作带来很大的压力。

为了保证各个业务系统的同源同根，就必须从一个数据系统往各个业务系统完成数据同步。特别地，针对eSapce EC系统来说，从企业的信息系统（如自建的人员管理系统）同步用户信息才能保证与各个企业业务系统的数据一致性。

本场景主要介绍EC的业务发放（企业号码跟企业成员的开销户等）和通讯录查询及部门管理的功能。

开发框架如下：



主要功能：

- 支持与微软AD直接对接
- 支持通过Excel模板导入
- 提供开销户和通讯录同步接口，支持第三方调用实现用户信息导入

本文仅介绍通过提供REST接口方式（上图中间的路径），完成通讯录同步的开发过程。

6.3.2 典型场景开发

本章的开发任务：提供标准RESTful接口调用，完成部门管理，企业号码管理，企业成员管理，所涉及到的接口如下所示：

- 企业部门管理接口
- 企业账号管理接口
- 企业号码管理接口

Sample 界面

图 6-4 企业部门管理界面

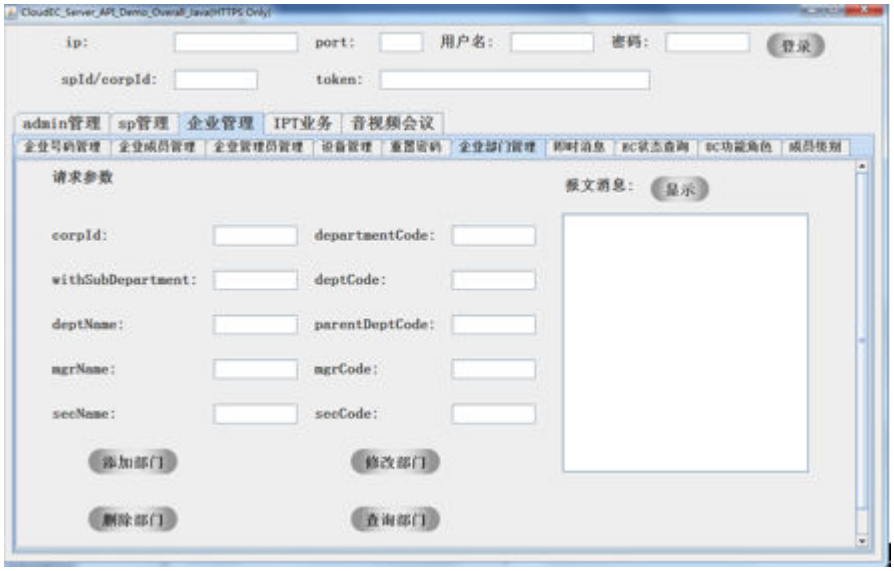


图 6-5 企业成员管理界面

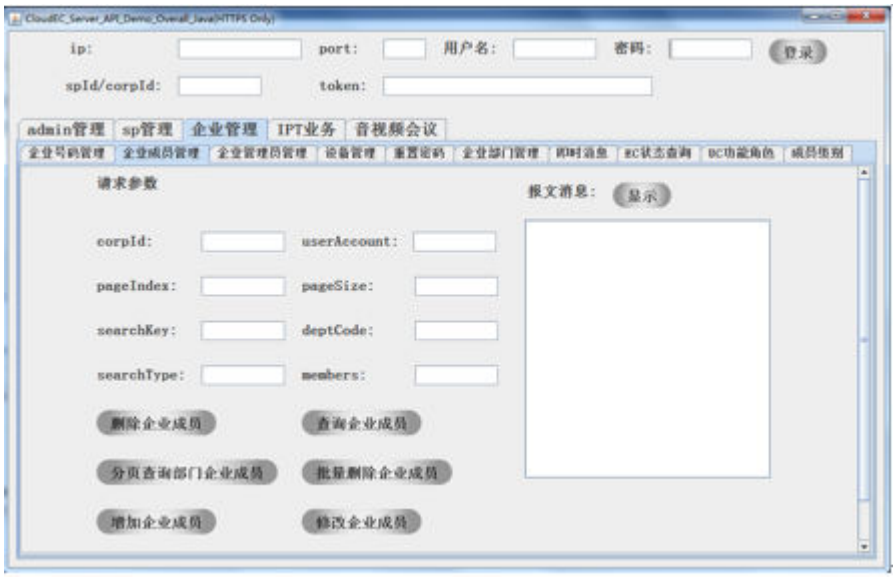


图 6-6 企业成员详情界面

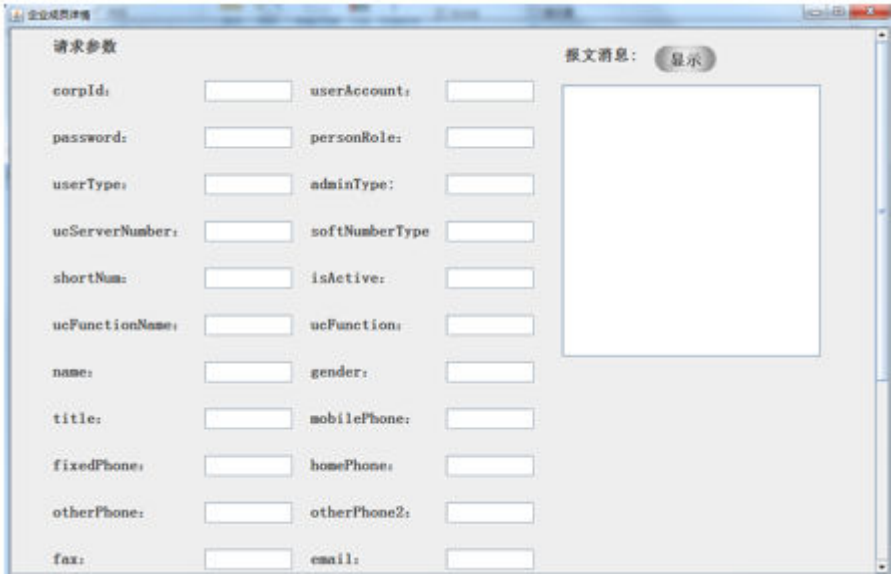


图 6-7 企业号码管理界面

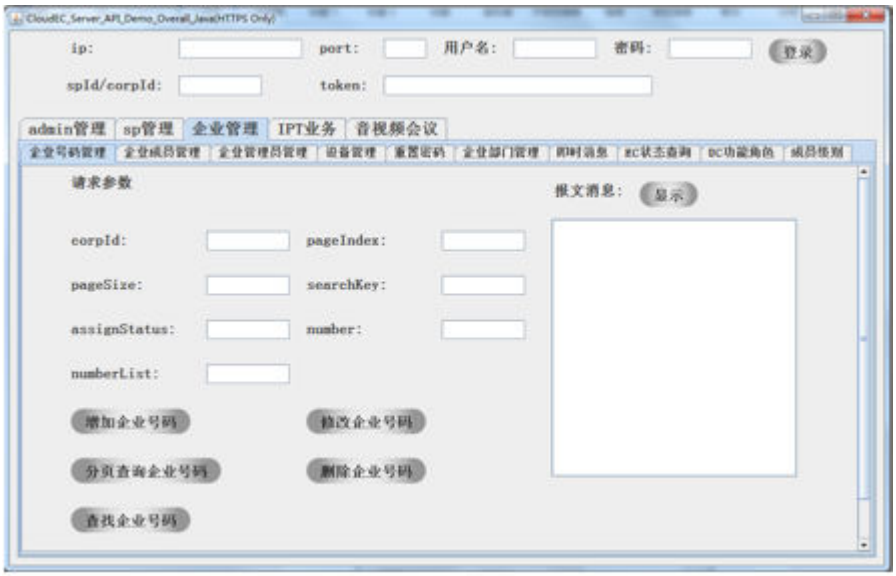
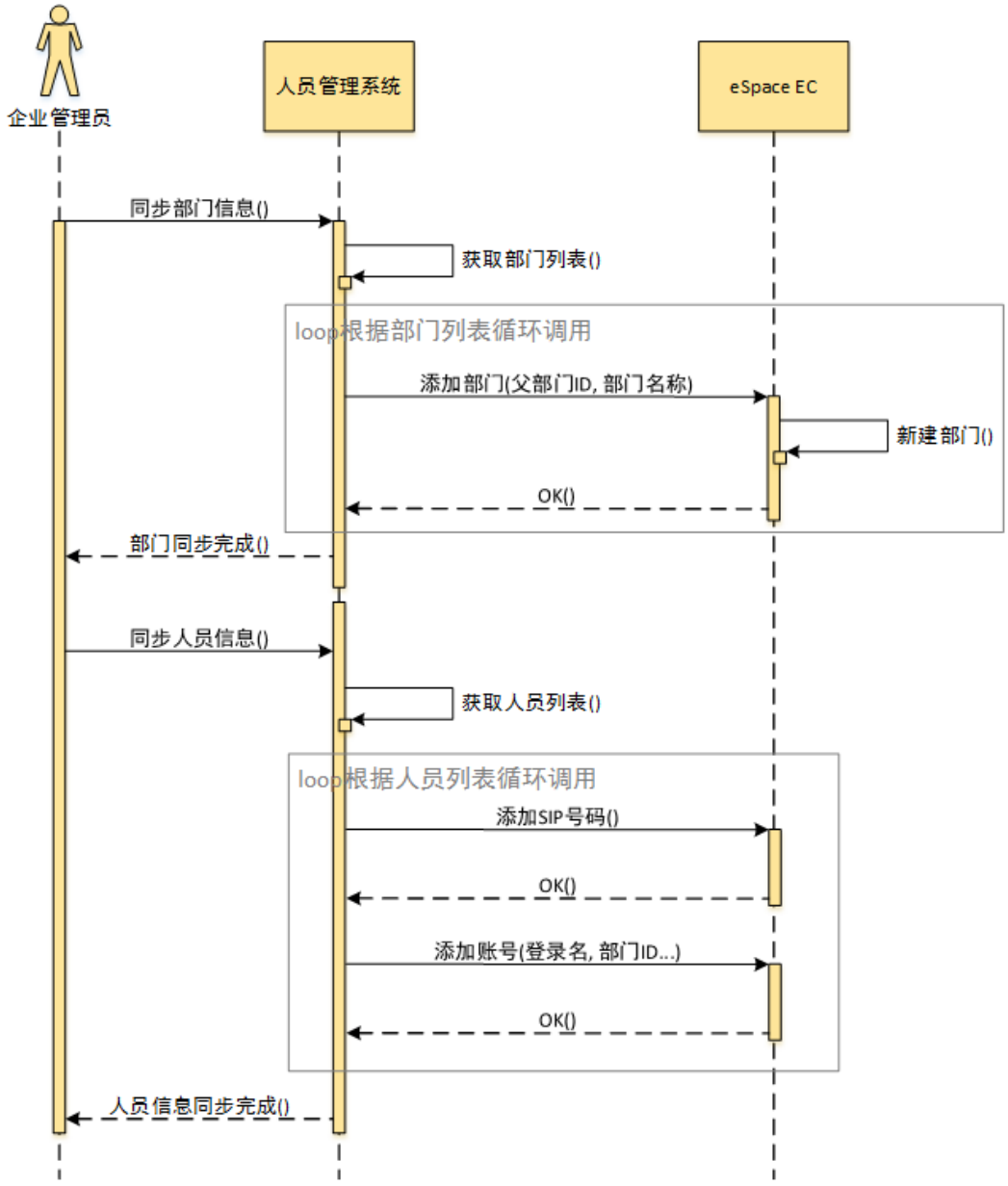


图 6-8 企业号码详情界面

时序图



流程说明：

步骤1 企业管理员向人员管理系统发起同步部门信息请求；

步骤2 人员管理系统获取部门列表；

步骤3 循环调用添加部门接口（传入父部门ID和本部门名称，请注意根部门的ID为“1”或者“-1”，请与系统管理员获取）；

步骤4 eSpace EC根据企业部门信息同步请求，创建完成部门树结构；

步骤5 企业管理员向人员管理系统发起同步人员信息请求；

步骤6 人员管理系统获取企业通讯录列表；

步骤7 调用添加企业号码接口；

步骤8 调用添加企业成员接口（传入员工的登录名、部门ID等属性）；

步骤9 根据企业通讯录成员列表，循环步骤**步骤7~步骤8**，完成所有人员信息的同步。

----结束

前提条件

启动eSDK CloudEC服务。

示例代码

- 添加企业部门

  根据页面中输入的部门编码、部门名称、父部门编码等请求参数，传入后台的addDept方法中进行拼装，通过调用sendMessage方法发送Rest消息，请求eSDK服务的添加部门接口，实现添加部门功能。

  - 代码示例：

    (POST)http://{ip}:{port}/corp/{corpId}/dept

  - 调用addDept方法拼装请求参数，发送Rest消息

```
//java code
// 添加部门
private void addDept()
{
    // 拼装请求参数，发送请求
    DeptInfo payload = new DeptInfo();
    payload.setDeptCode(deptCodeField.getText());
    payload.setDeptName(deptNameField.getText());
    payload.setParentDeptCode(parentDeptCodeField.getText());

    if (StringUtils.isNotEmpty(mgrNameField.getText()))
    {
        payload.setMgrCode(mgrNameField.getText());
    }
    if (StringUtils.isNotEmpty(mgrCodeField.getText()))
    {
        payload.setMgrName(mgrCodeField.getText());
    }
    if (StringUtils.isNotEmpty(secNameField.getText()))
    {
        payload.setSecName(secNameField.getText());
    }
    if (StringUtils.isNotEmpty(secCodeField.getText()))
    {
        payload.setSecCode(secCodeField.getText());
    }

    RestRequest request = new RestRequest(HTTPConstant.HTTP_METHOD_POST);
    request.setPayload(payload)
    try
    {
        Token token = LoginUtils.getToken();
        ecService.post("/corp/" + corpIdField.getText() + "/department",
        request, errInfoLabel, token);

    }
    catch (Exception e)
    {
        showErrInfoWithColor("操作失败");
        LOGGER.error("get Token error:" , e);
    }
    finally
    {
        ecService.finish();
```

```
    }
}
```

- **sendMessage方法请求**的添加部门接口代码参考
- **service层处理返回response**对应代码

6.4 点击呼叫

6.4.1 概述

点击呼叫（CTD，Click To Call）又称双向回拨，是通过传统电信网外呼双方手机或固话号码，将双方建立通话，电商类、培训资讯类（如在线医疗咨询，售后客户关怀）网站嵌入双向回拨按钮，让网站客户能够免费、便捷的与网站客服的沟通，提高服务满意度，助力销售转化；基于华为通讯平台的双向回拨接口，信息平台类的网站能够实现用户号码保护，注重客户隐私、提升平台价值。

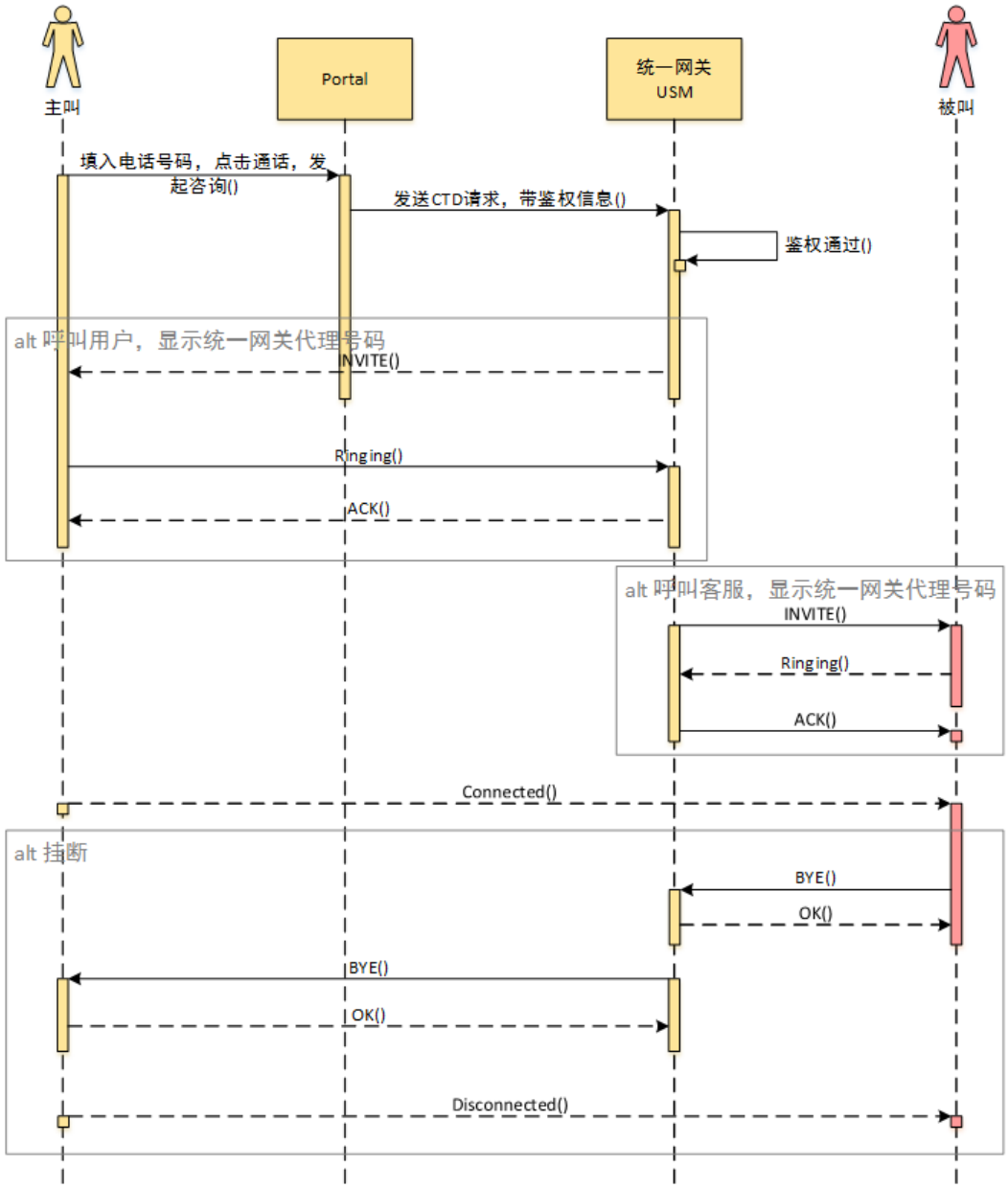
6.4.2 典型开发场景

本章的开发任务：通过eSDK服务端提供的RESTful接口，完成CTD呼叫的功能。

Sample 界面

图 6-9 CTD 呼叫界面

时序图



流程说明：

步骤1 用户填入操作者企业成员账号、主叫号码、被叫号码，发起CTD呼叫；

步骤2 Portal发送CTD请求给统一网关，携带参数为用户的主叫号码及被叫号码；

步骤3 统一网关收到请求，先向用户的主叫号码发起呼叫；

步骤4 用户收到来电邀请，显示号码为统一网关代理号码，接听来电；

步骤5 统一网关向被叫号码发起呼叫；

步骤6 被叫收到来电邀请，显示号码为统一网关代理号码，接听来电；

步骤7 用户与被叫之间建立通话，开始沟通；

步骤8 被叫挂断电话，向统一网关发送挂断请求；

步骤9 统一网关响应挂断请求，并通知主叫挂断；

步骤10 主叫挂断，通话结束。

----结束

前提条件

启动eSDK CloudEC服务，添加企业成员。

示例代码

根据页面中输入的用户账号、主叫号码，被叫号码请求参数，传入后台的call方法中进行拼装，通过调用sendMessage方法发送Rest消息，请求API网关发起CTD呼叫接口，实现CTD呼叫功能。

代码示例：

(POST)http://{ip}:{port}/opengw/number/{number}/ctd

调用call方法拼装请求参数，发送Rest消息

```
//java code
// 点击呼叫
private void call()
{
    RestRequest request = new RestRequest(HTTPConstant.HTTP_METHOD_POST);
    //拼接参数发送请求
    CallBean payload = new CallBean();
    payload.setCallee(calleeField.getText().trim());
    if (StringUtils.isNotEmpty(callerField.getText().trim()))
    {
        payload.setCaller(callerField.getText().trim());
    }
    request.setPayload(payload);
    try
    {
        Token token = LoginUtils.getToken();
        ecService.post(requestBtn, responseBtn, "/number/" + numberField.getText().trim() +
        "/ctd", retDescArea, request, errInfoLabel, token);

    }
    catch (Exception e)
    {
        LOGGER.error("getToken error", e);
    }
    finally
    {
        ecService.finish();
    }
}
```

- **sendMessage方法**调用eSDK服务的CTD呼叫接口代码
- **service层处理返回response**对应代码

6.4.3 扩展开发场景

将接口封装成移动端本地API，集成到客户App中，实现与其生产系统的集成。

上图中描叙的是某汽车销售4S店，在客户进店保养后，处理该工单时，如遇到问题需要跟客户沟通，可以发起一键呼叫（可隐藏客户电话号码，控制竞争风险；快捷发起呼叫，提高工作效率）。在处理完后，也可以进行售后跟踪，发起客户关怀，提升客户满意度。

6.5 会议管理

6.5.1 概述

企业的日常会议数量多且密度高，有对企业内部的也有对外的，因为使用频繁对会议质量有一定的要求。传统常规会议受时间、地域限制，通常无法解决跨地域开会问题，或者需要与会人员出差，会议效率较低，且成本居高不下。华为的多媒体会议系统能加快信息传递速度，强化管理效率。

6.5.2 典型开发场景

本章的开发任务：通过eSDK服务端提供的RESTful接口，完成对会议的管理能力（预定、查询、修改、删除和添加与会人等操作）。开发者可以通过浏览器下载一个会议客户端，加入到已申请开通的会议中去。

Sample 界面

图 6-10 预约会议界面

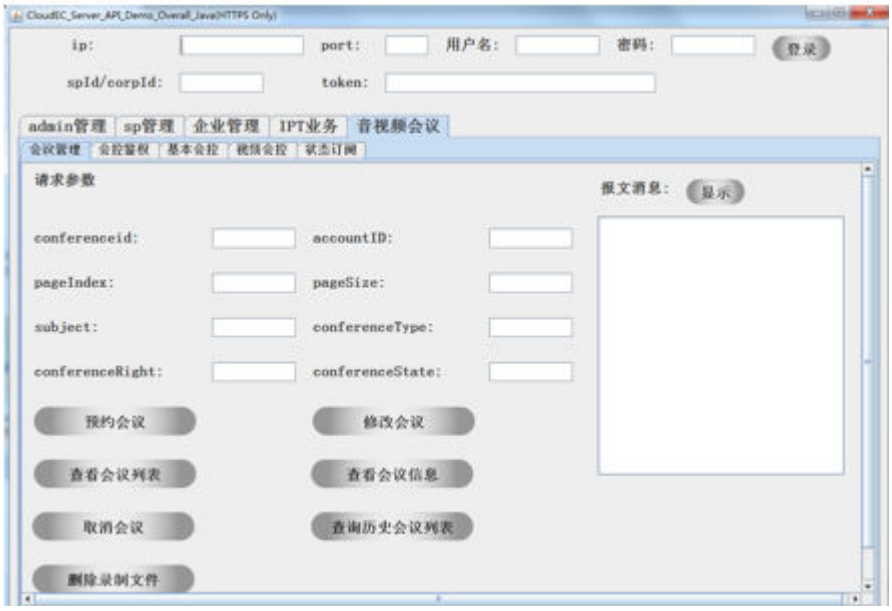
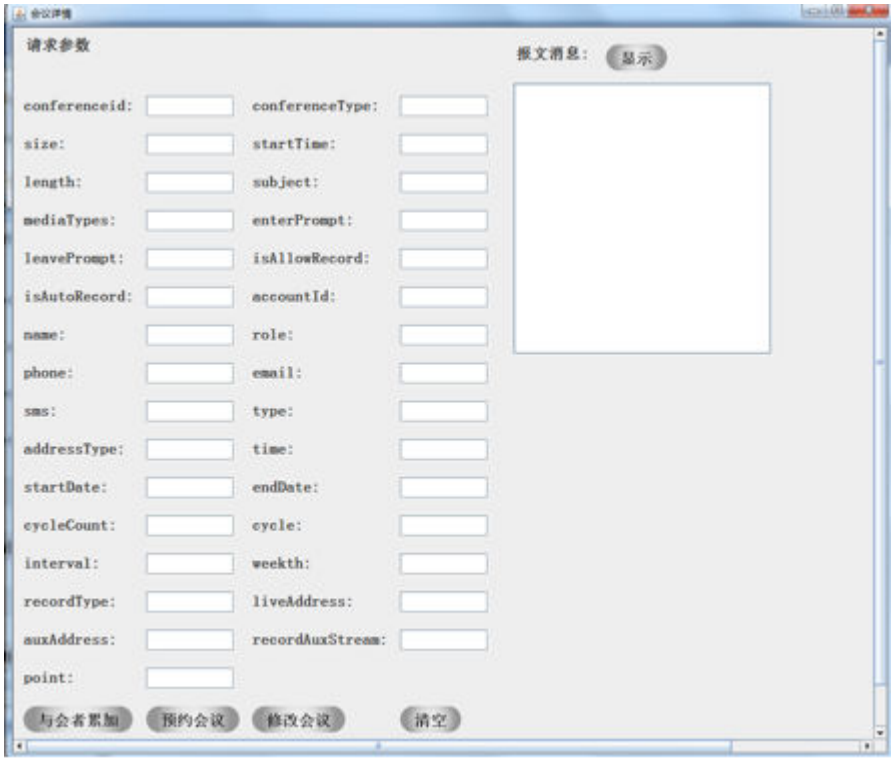


图 6-11 点击预约会议，弹出会议参数详情



流程说明

步骤1 企业工作人员在会议管理系统中预约一个会议，传入预约会议所需的参数如：会议时间，与会人数、会议密码、会议主题等等；

步骤2 会议管理系统调用预约会议接口（传入会议时间，与会人数、会议密码、会议主题等参数）预约一个会议成功；

步骤3 会议预约人员根据需求添加需要参与会议的成员，调用邀请与会者接口完成与会人的添加；

步骤4 如果会议已经预约成功，调用修改会议接口，修改会议的基本信息如：会议时间、人数等；

步骤5 企业工作人员可以根据传入的会议开始、结束时间、主题等参数，调用查询会议列表接口，查询用户所有预约的会议信息；

步骤6 如果要取消预约的会议，用户可以传入会议ID、网关IP调用取消预约会议接口，删除已预约的会议。

----结束

示例代码

- 预定会议

  根据eSDK CloudEC接口文档中对应的预定会议接口的输入参数，在Sample界面上输入会议类型、与会人数、会议主题、媒体类型等参数值，传入后台的scheduledMeeting方法中进行拼装，通过调用sendMessage方法发送Rest消息，请求eSDK服务的预定会议接口，实现预定一个预约会议功能，根据返回参数获得预约的会议ID。

  - 代码示例：

    (POST)http://{ip}:{port}/conferences

```
//java code
// 预约会议
private void scheduledMeeting()
{
    try
    {
        Token token = LoginUtils.getToken();

        // 利用token发送restful请求
        RestRequest request = new RestRequest(HTTPConstant.HTTP_METHOD_POST);

        //参数的设定
        request.setPayload(addScheduleMeetingBean());

        ecService.post("/conferences", request, errInfoLabel, token);
    }
    catch (Exception e)
    {
        LOGGER.error("get Token error:" + e);
        showErrInfoWithColor("操作失败");
    }
    finally
    {
        ecService.finish();
    }
}
```

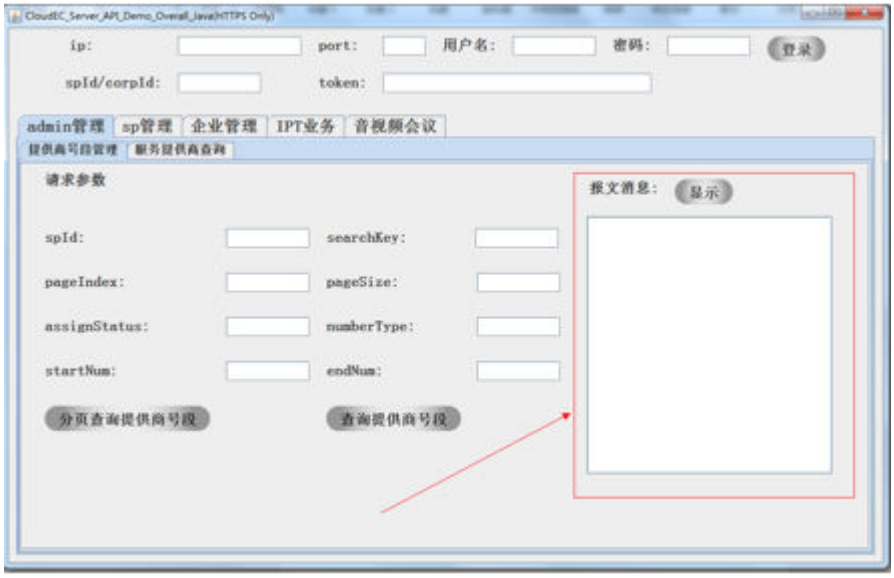
  - **sendMessage方法**请求eSDK服务的对应接口代码

- **service层处理返回response**对应代码

7 消息打印

Sample 界面

图 7-1



7.1 请求&响应消息

点击“显示”按钮，显示请求&响应报文信息

7.2 示例代码

```
private void init()
{

    this.showBtn.addMouseListener(new MouseAdapter()
```

```
    {
        public void mouseClicked(MouseEvent e)
        {
            retDescArea.setText("");
            if (StringUtils.isNotEmpty(ResultUtils.getReq()))
            {
                retDescArea.setText(ResultUtils.getReq() +"\r\n" +"----------------------------"
                + "\r\n" + ResultUtils.getResponse());
            }
        }
    });
}
```

8 定位指南

错误码获取方法

每个接口调用后，都会有一个返回值。如果返回值为0，表示成功；否则表示失败，该返回值即为错误码。

错误信息查询方法

在《eSDK CloudEC 1.1.0 接口参考 (REST,USM)》中，列出了所有错误码信息，如下截图。

根据接口返回的错误码，查询相对应的错误描述。

9 附录

9.1 sendMessage 方法请求

```
//java code
public RestResponse sendMessage(Token token, String resourceUri, RestRequest message)
throws ClientProtocolException, URISyntaxException, IOException
{
    //write the token to the request
    Map<String, String> mapheaders = message.getHttpHeaders();
    String authorization = token.getToken();
    mapheaders.put("Authorization", authorization);

    //getting the HTTP host information
    HttpHost target = token.getHttpHost();

    return sendMessage(target, resourceUri, message, null, null);
}

    public RestResponse sendMessage(HttpHost target, String resourceUri, RestRequest message,
String userName,
            String password)throws ClientProtocolException, URISyntaxException, IOException
    {

        if (null == target)
        {
            throw new NullPointerException("HttpHost target can not be null.");
        }
        else
        {
            this.target = target;
        }

        RestResponse restResponse = new RestResponse();

        adapterScheme();

        HttpRequestBase request = buildRequestMessage(message, resourceUri);


        HttpResponse response = null;
```

```
        try
        {
            response = client.execute(target, request, localContext);

            if (response != null)
            {
                restResponse.setHttpCode(response.getStatusLine().getStatusCode());

                //获取响应体
                HttpEntity entity = response.getEntity();
                if (null != entity)
                {
                    restResponse.setEntity(EntityUtils.toString(entity));
                }

                Header[] headers = response.getAllHeaders();
                if (null != headers)
                {
                    Map<String, List<String>> headerMap = restResponse.getHeaders();

                    //获取响应头消息
                    for (Header header : headers)
                    {
                        if (headerMap.containsKey(header.getName()))
                        {
                            headerMap.get(header.getName()).add(header.getValue());
                        }
                        else
                        {
                            List<String> headerVals = new ArrayList<String>();
                            headerVals.add(header.getValue());
                            headerMap.put(header.getName(), headerVals);
                        }
                    }
                }

                return restResponse;
            }

            return null;

        }
        catch (Exception e)
        {
            LOGGER.error("httpclient error", e);
            return null;
        }
        finally
        {
            if (null != request)
            {
                request.releaseConnection();
            }
        }
    }
```

9.2 service 层处理返回 response

```
//java code
private String analysisResult(RestResponse response, JLabel errInfoLabel)
{
    if (STATUSCODE_OK != response.getHttpCode())
    {
        showErrInfoWithColor("操作失败", errInfoLabel);

        String result = "HTTP/1.1 " + response.getHttpCode() + "\n" +
StringUtils.formatHttpHeads(response);
        return result;
```

```
    }
    else
    {
        String entity = response.getEntity();

        //把json转换为javaBean
        QueryResponse<?> responseBean = new Gson().fromJson(StringUtils.formatJson(entity),
QueryResponse.class);

        //处理data，获取会议token的,其他接口返回不包含token，只有会控鉴权有返回token值
        Object data = responseBean.getData();
        beanToJson = StringUtils.beanToJson(data);
        if (beanToJson.startsWith("{"))
        {
            HashMap<String, Object> jsonToMap = StringUtils.jsonToMap(beanToJson);
            if (jsonToMap.containsKey("token"))
            {
                String meetingToken = "Basic " + (String)jsonToMap.get("token");
                //将token存储起来
                MeetingTokenUtils.setMeetingToken(meetingToken);
            }

        }

        if (RETURNCODE_OK == responseBean.getReturnCode())
        {
            showErrInfoWithColor("操作成功", errInfoLabel);
        }
        else
        {
            showErrInfoWithColor("操作失败", errInfoLabel);
        }

        String result = "HTTP/1.1 " + response.getHttpCode() + "\n" +
        StringUtils.formatHttpHeads(response) + "\n" +
        StringUtils.formatJson(entity);
        return result;
    }
}
```

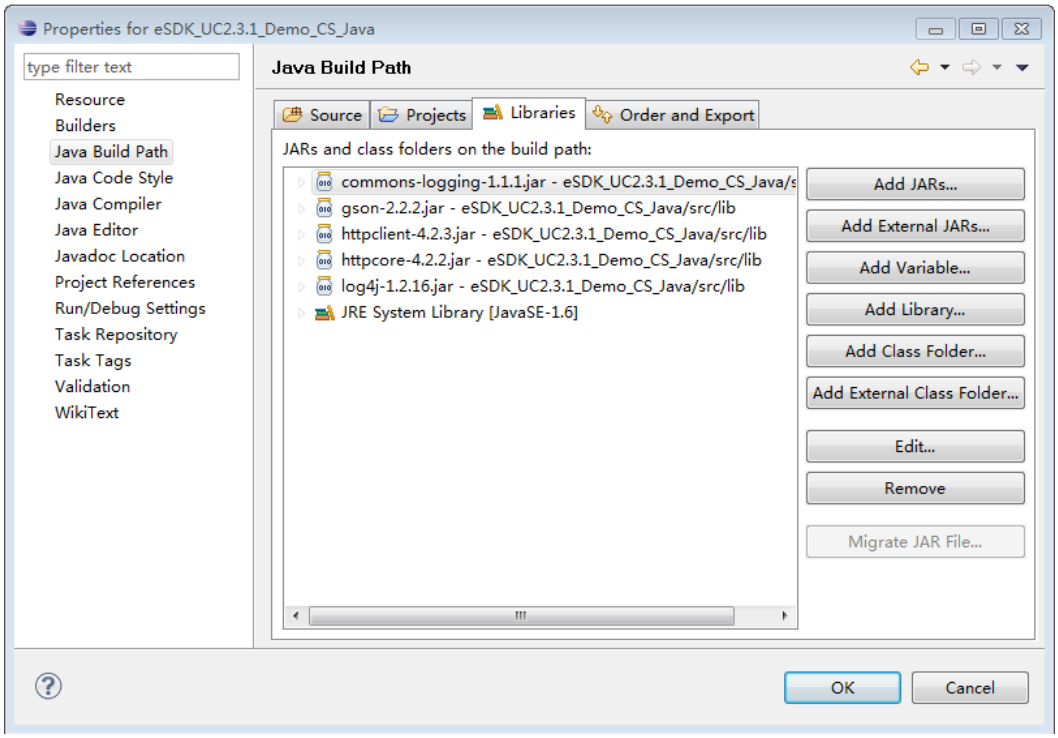
9.3 FAQ

9.3.1 如何修改 JRE 版本？

此步骤根据需要选择，如果不需要修改JRE版本，可以跳过此步骤，本节主要以从JRE1.6换到新安装的JRE1.8为例。

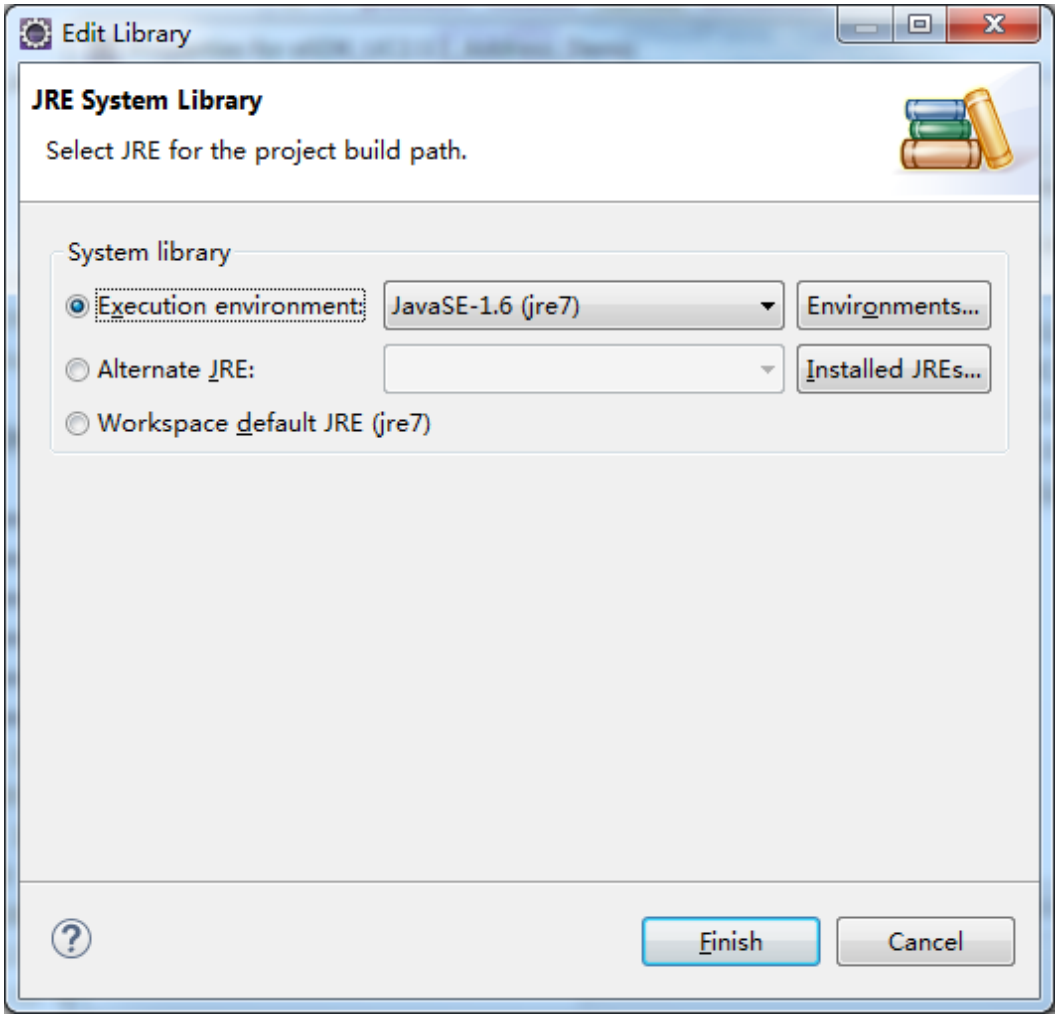
步骤1 正确安装完成jdk1.8.0_92后，右键单击demo工程名，选择“Build Path >Configure Build Path”，显示工程属性窗口，如图9-1所示。

图 9-1 Libraries



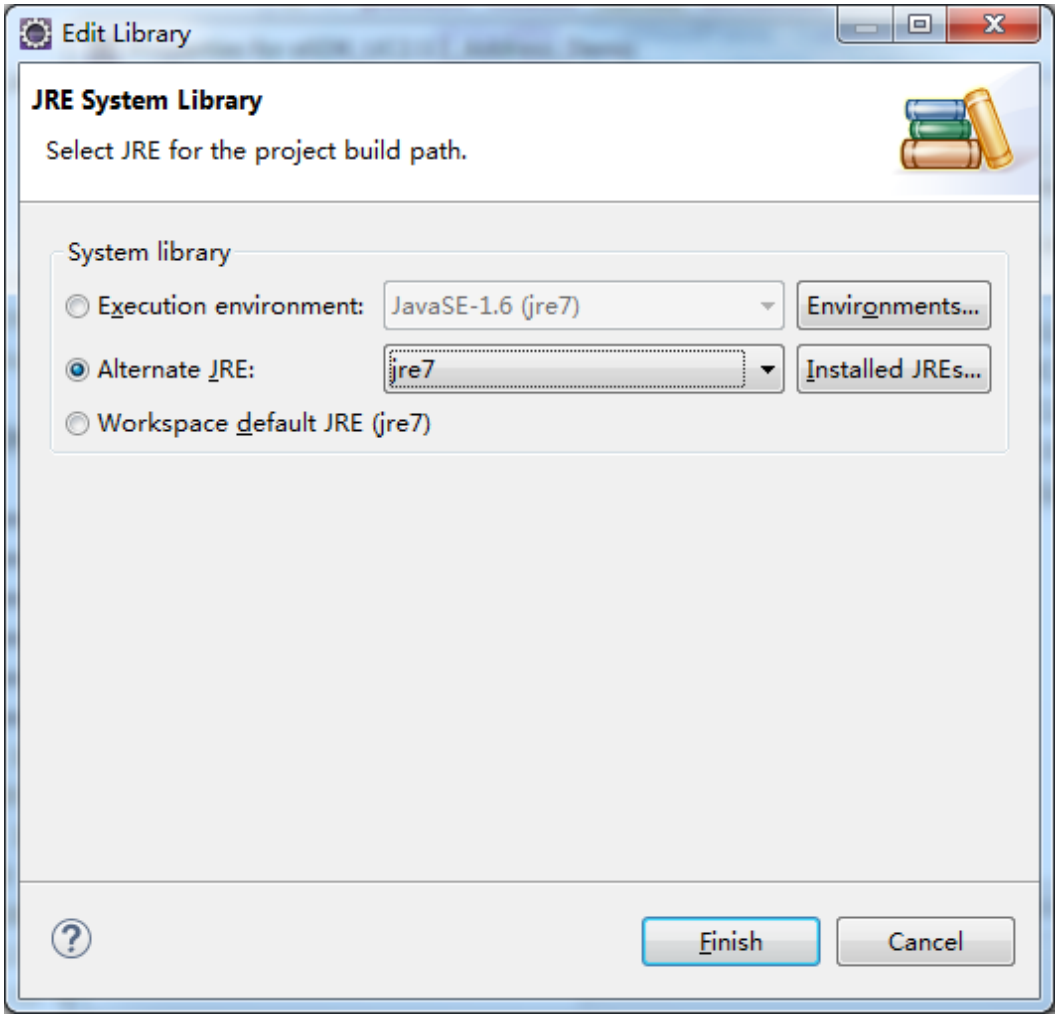
步骤2 选中“JRE System Library[JavaSE-1.6]”，单击“Edit”按钮，显示编辑库窗口，如图9-2所示。

图 9-2 Edit Library



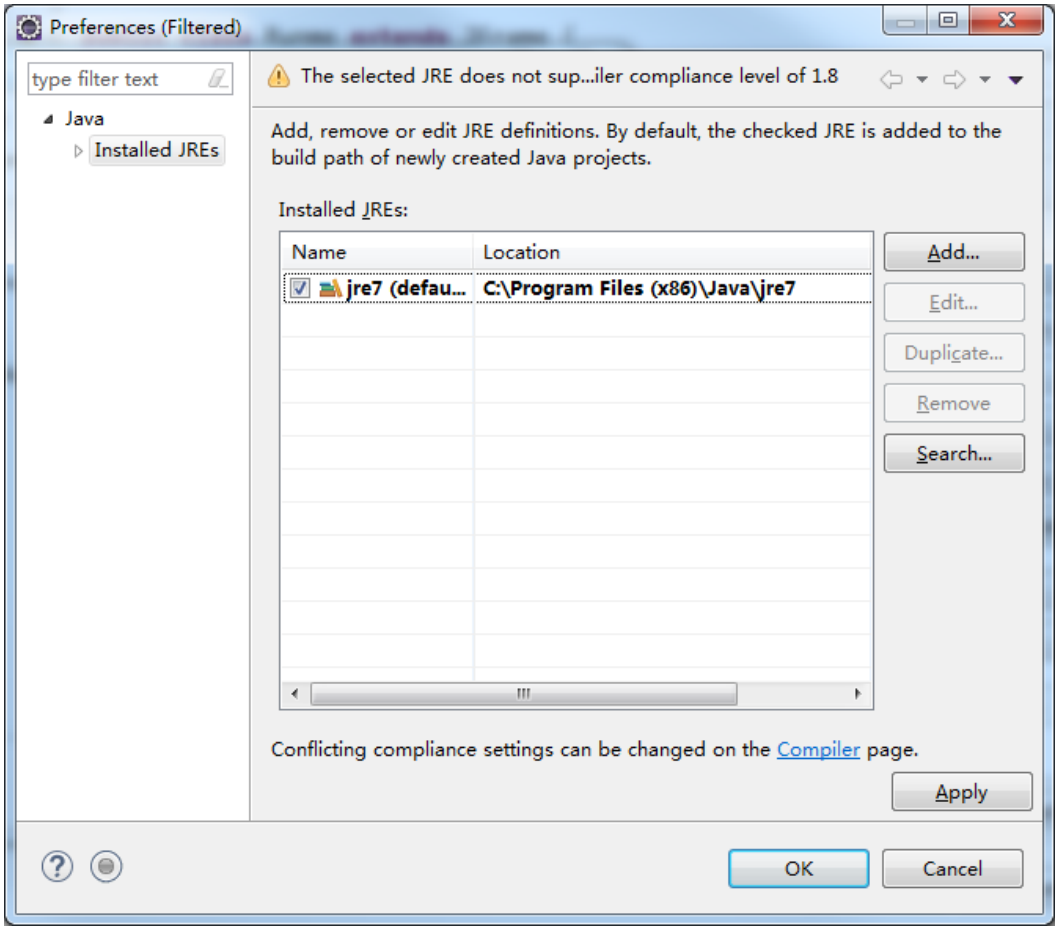
步骤3 选择“Alternate JRE”，图中出现的jre7是已导入的JRE1.7版本，可忽略内容显示，如图9-3所示。

图 9-3 Alternate JRE



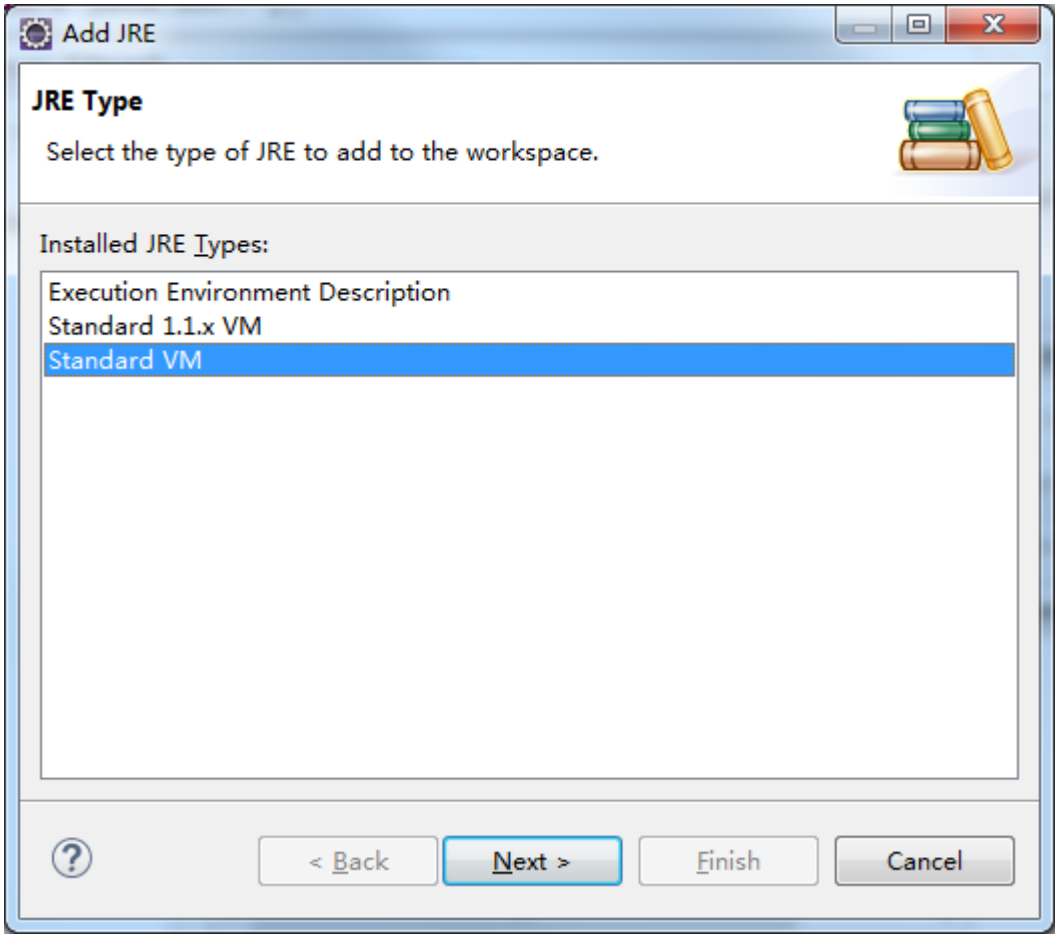
步骤4 单击“Installed JREs”，如图9-4所示。

图 9-4 Preferences



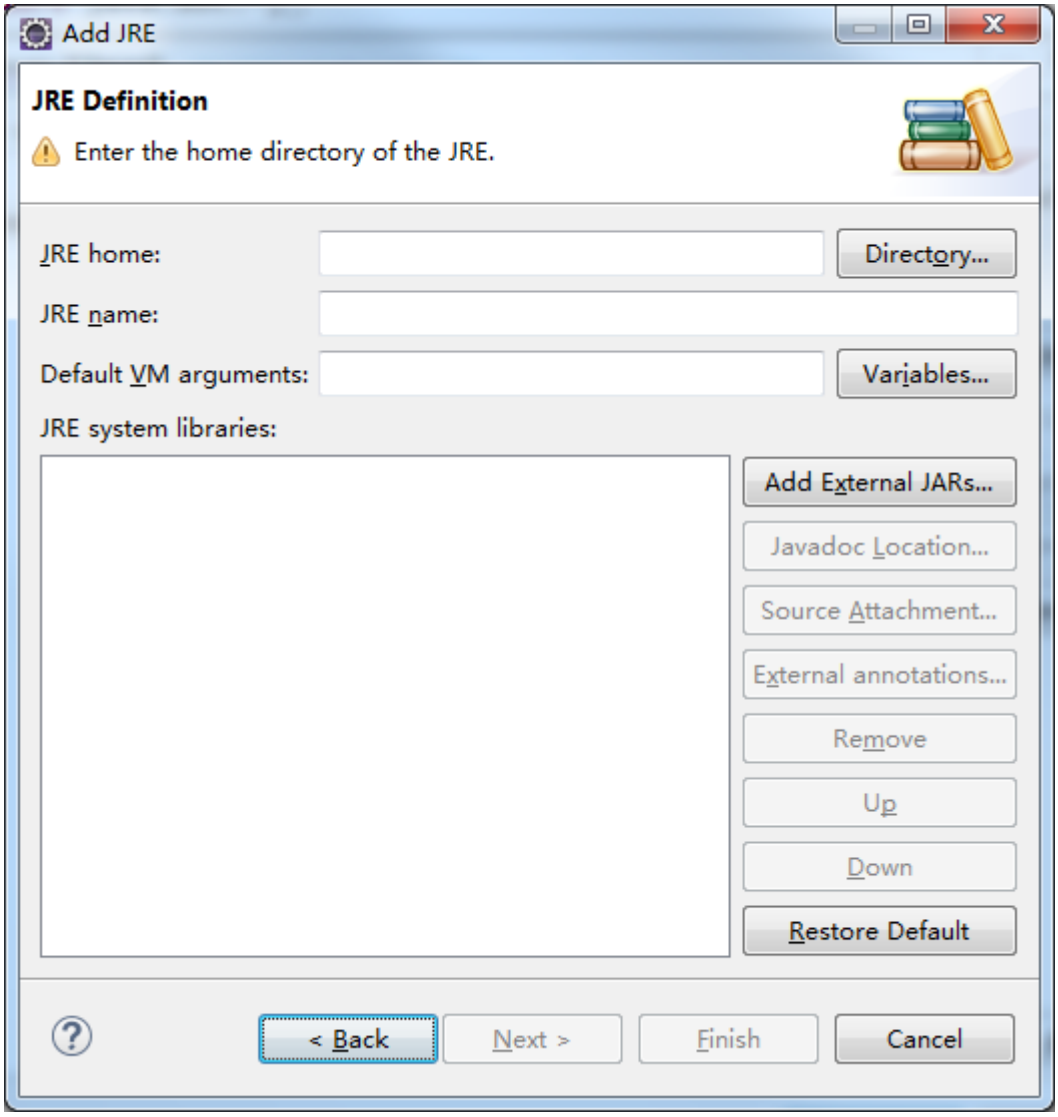
步骤5 单击“Add”按钮，显示添加JRE界面，如图9-5所示。

图 9-5 Add JRE



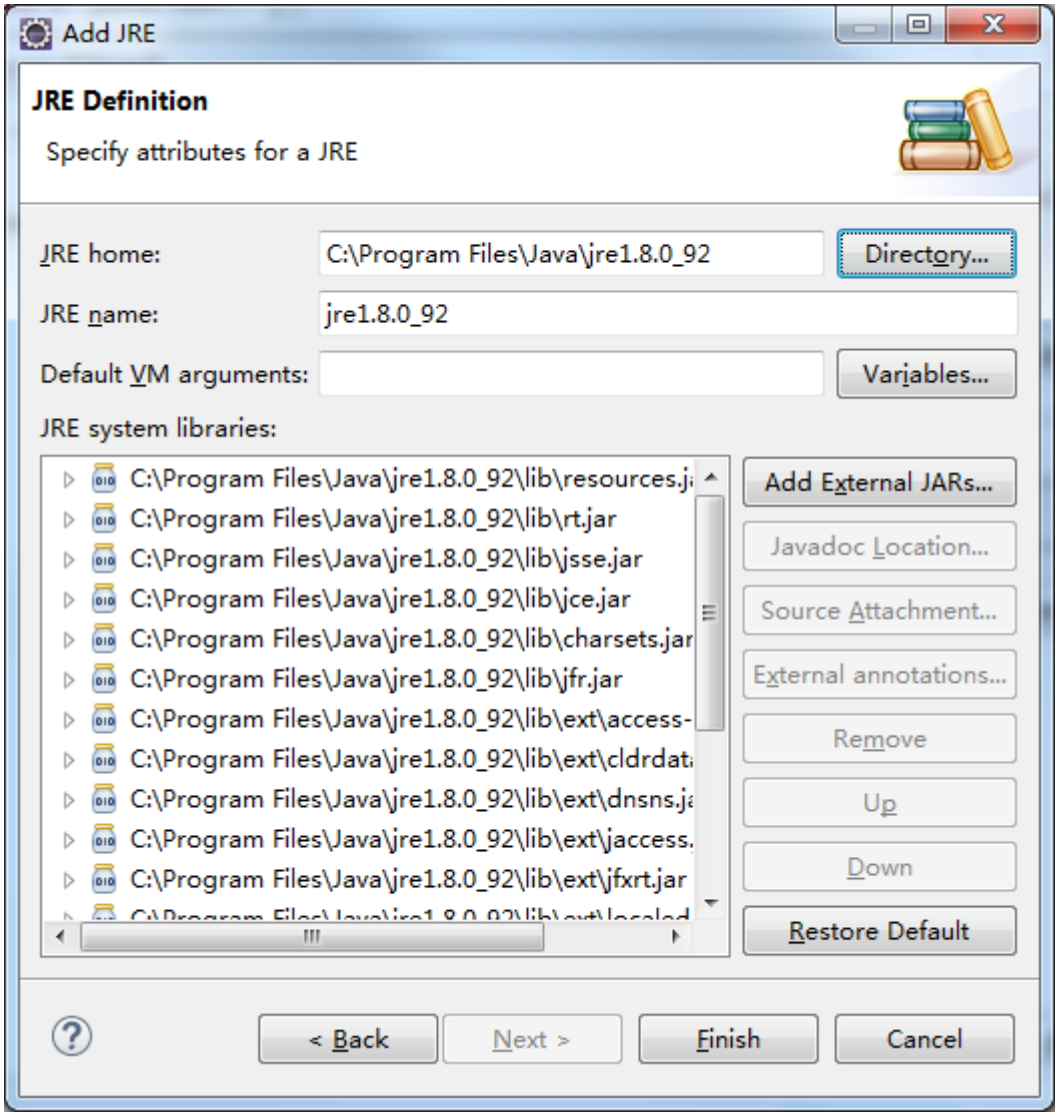
步骤6 单击“Next”，如图9-6所示。

图 9-6 JRE Definition1



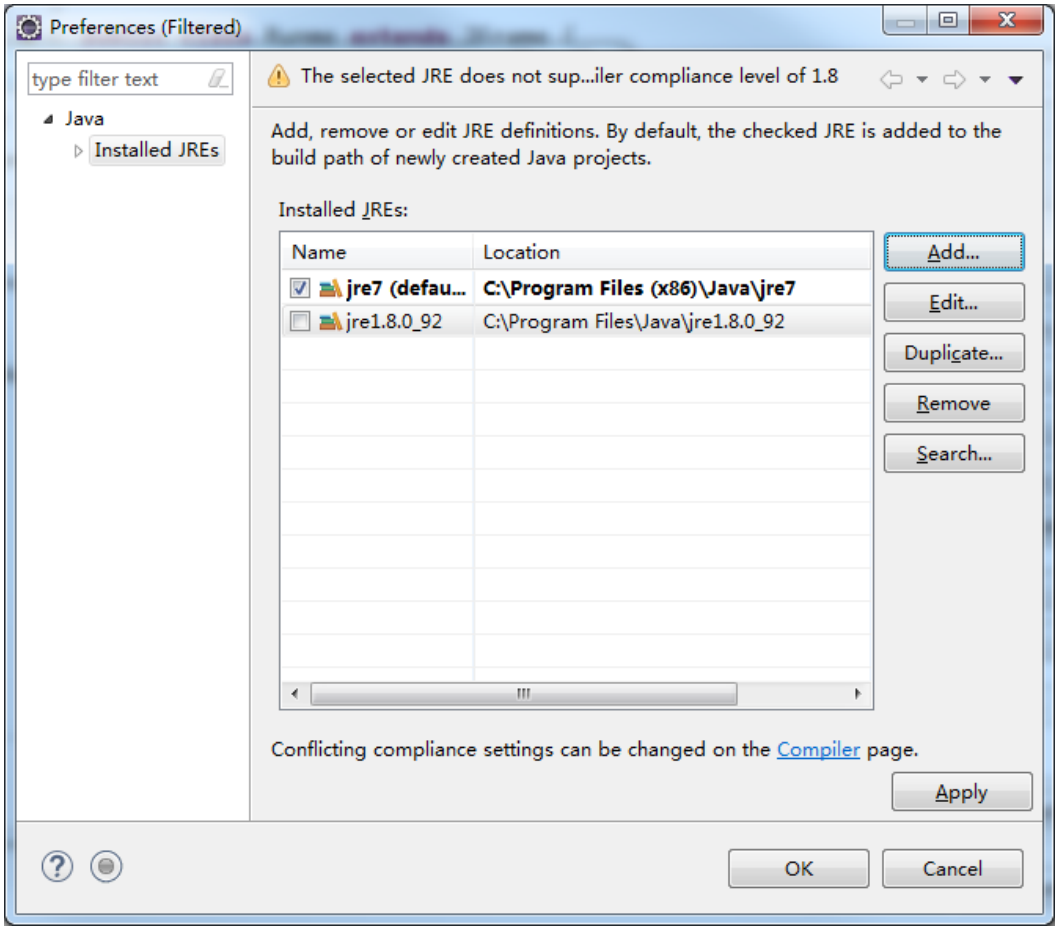
步骤7 单击“Directory”，添加已安装的JRE路径，如图9-7所示。

图 9-7 JRE Definition2



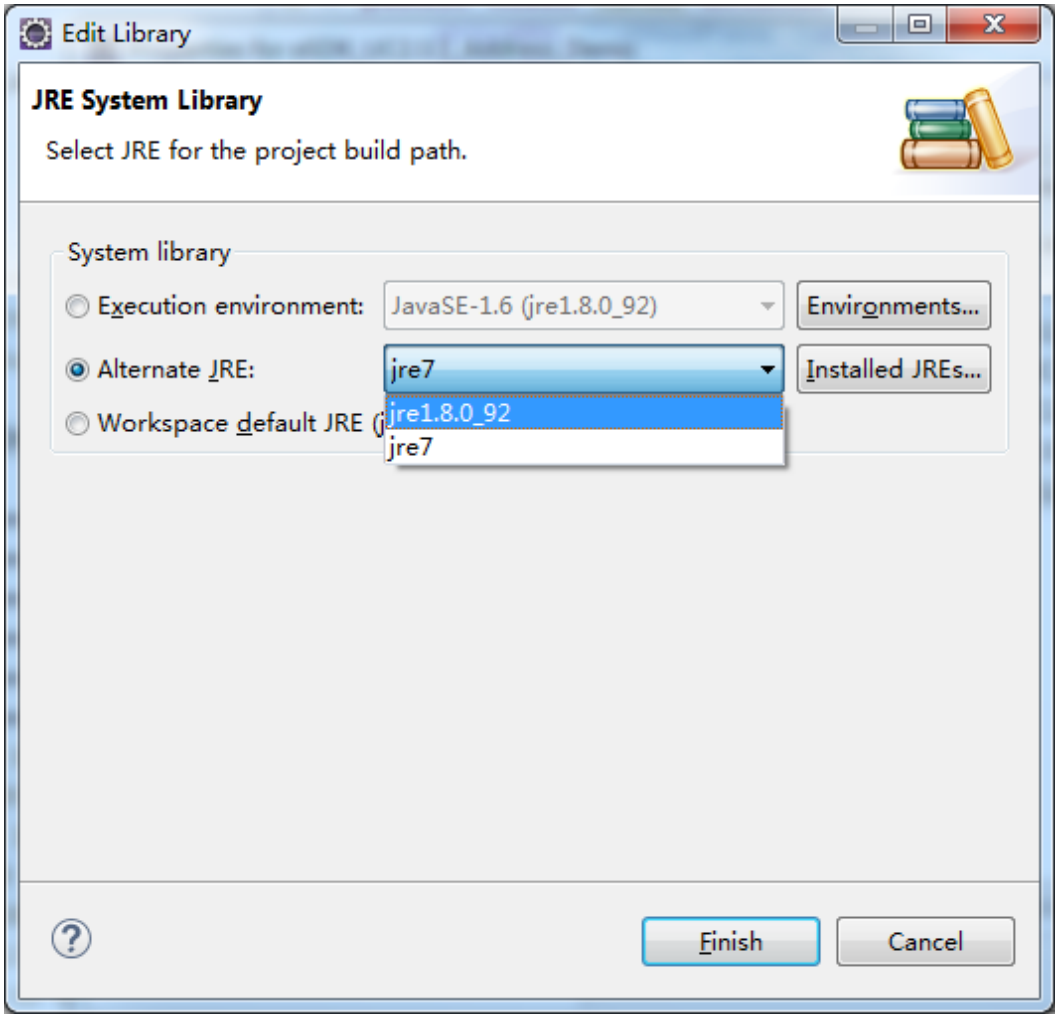
步骤8 单击“Finish”，完成JRE添加，如图9-8所示。

图 9-8 Preferences1



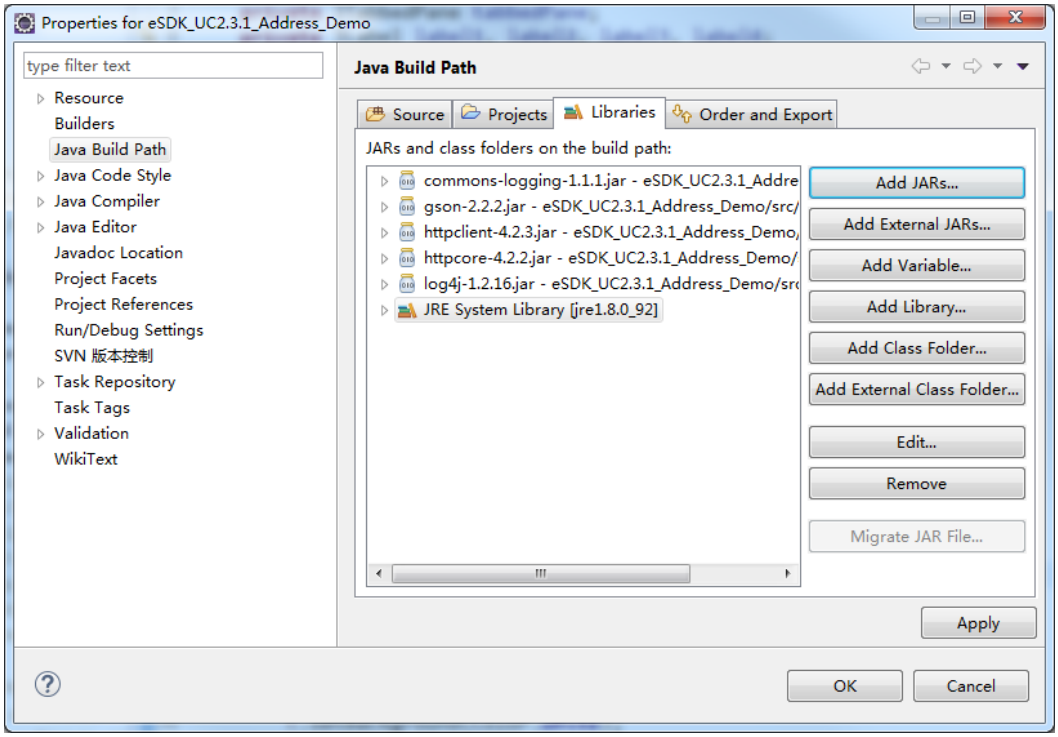
步骤9 勾选中“jre1.8.0_92”，单击“Apply”应用设置，然后单击“OK”完成添加。完成后，在Edit Library窗口的Alternate JRE下拉框中显示刚才添加的JRE，如图9-9所示。

图 9-9 Alternate JRE1



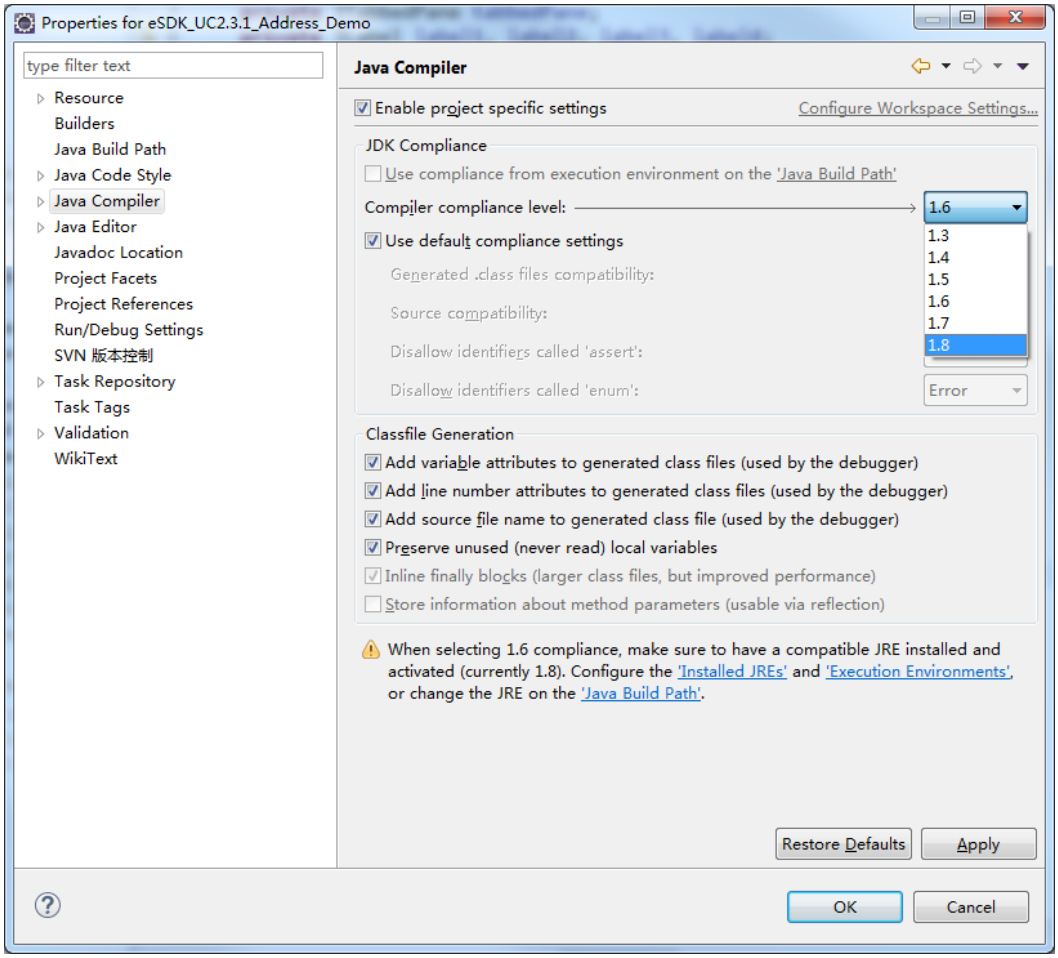
步骤10 选中“Alternate JRE”下拉框“jre1.8.0_92”，单击“Finish”，完成demo工程切换jre版本，如图9-10所示。

图 9-10 Libraries1



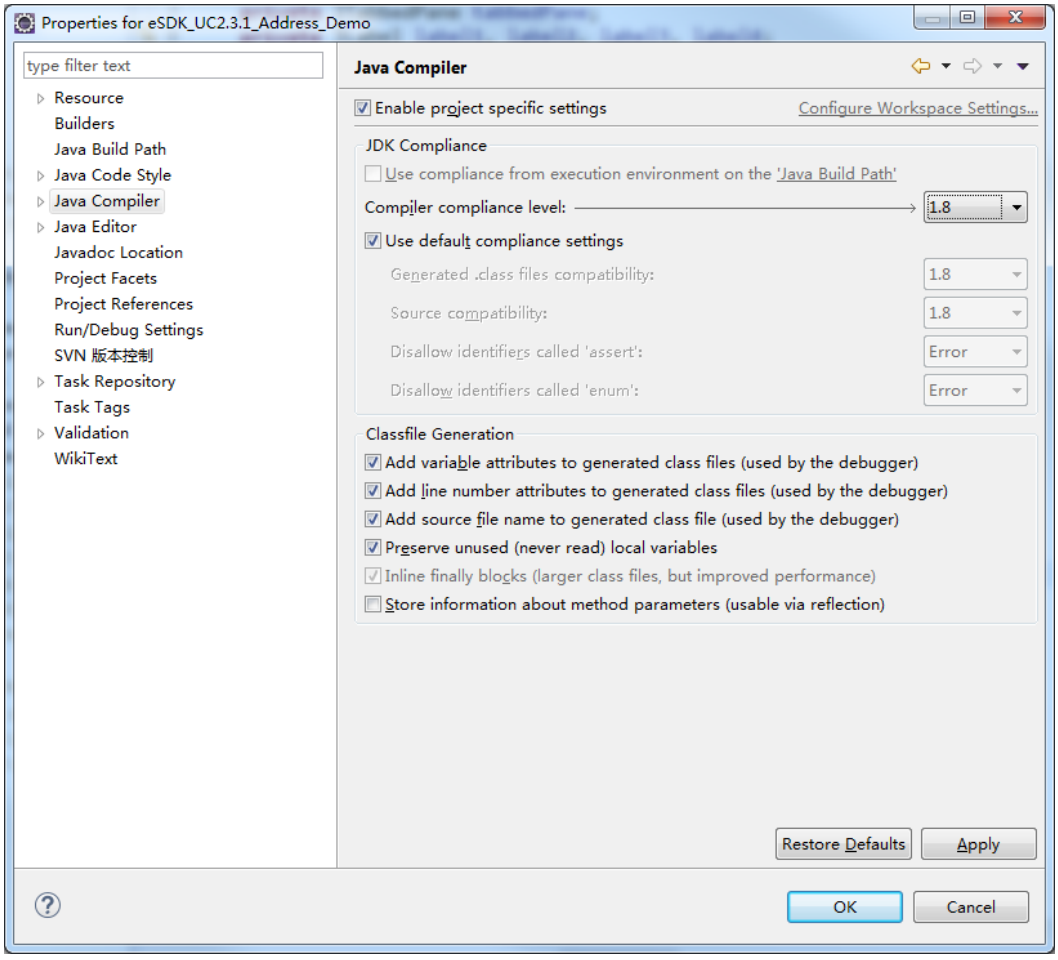
步骤11 上图单击“Apply”后，在左边选中“Java Compiler”标签页，如图9-11所示。

图 9-11 Java Compiler



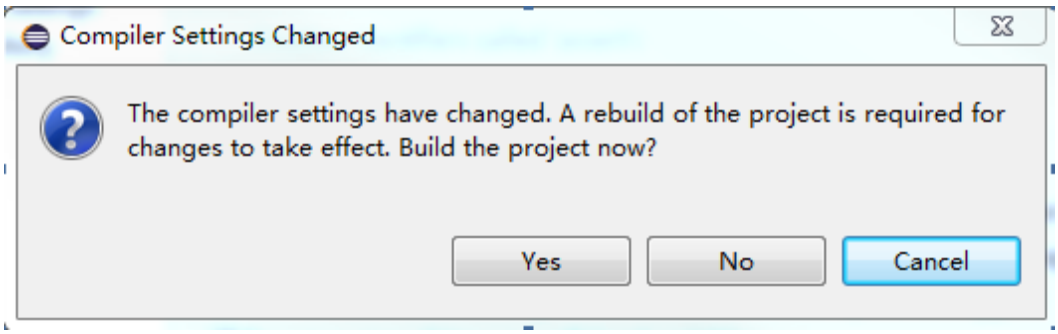
步骤12 在“Compiler compliance level”下拉菜单中选择jre相应“1.8”版本，如图9-12所示。

图 9-12 Java Compiler1



步骤13 单击“Apply”应用设置，单击“OK”，弹出如图9-13确认框。单击“Yes”，完成全部设置。

图 9-13 Compiler Settings Changed



----结束

10 历史修订记录

日期	修订版本	描述
2017-10-31	1.1.0	1.1.0版本文档首次发布。