|  |
| --- |
| 说明: 金智教育-标志与标准字左右组合规范-小.bmp |
| **CIAPV7.0**  **java应用集成接口说明** |
| **版权声明和保密须知**  本文件中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明，版权均属江苏金智教育信息股份有限公司所有，受到有关产权及版权法保护。任何单位和个人未经江苏金智教育信息股份有限公司的书面授权许可，不得复制或引用本文件的任何片断，无论通过电子形式或非电子形式。  **Copyright © 2015江苏金智教育信息股份有限公司版权所有** |

修改记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序 | 修改人 | 修改内容 | 批准人 | 生效日期 | 版本号 |
| 1 | 许继锋 | 创建文档 |  |  | V0.1 |
| 2 | 衡南南 | 完善文档 |  |  | V1.0 |
| 3 | 刘伟锋 |  |  | 2016-05-19 | V1.1 |
| 4 | 刘伟锋 | 增加白名单 |  | 2017-04-05 | V1.2 |
| 5 | 刘晓峰 | 更新适配CIAP7.0 |  | 2020-06-16 | V2.0 |

文件信息

|  |  |
| --- | --- |
| **起草人：**许继锋 | **审核人：**刘传先 |
| **批准人：**刘传先 | **生效日期：**2015年3月1日 |
| **【文件摘要】**本文件针对CIAP7.0的实施及升级工作进行了纲领性描述，以做为本产品的实施指南。  **【阅读须知】**读者需要熟悉《CIAP7.0术语清单》，并了解项目管理相关要求。  **【发布范围】**产品经理、产品工程部、产品实施人员（含项目经理、合作伙伴等）。  **【发布方式】**小助手平台。 | |

目录

[1 前言 1](#_Toc465237079)

[1.1 编写目的 1](#_Toc465237080)

[1.2 服务对象 1](#_Toc465237081)

[2 集成方式概述 2](#_Toc465237082)

[3 java认证接口集成 3](#_Toc465237083)

[3.1 拷贝提供的jar 3](#_Toc465237084)

[3.2 修改web.xml文件 3](#_Toc465237085)

[3.3 获取用户信息 7](#_Toc465237086)

[3.4 集成应用退出 7](#_Toc465237087)

[3.5 认证接口工作过程 9](#_Toc465237088)

[4 集成应用添加SSL证书信任 10](#_Toc465237089)

[5 demo实例 15](#_Toc465237090)

[5.1 安装jdk及配置环境变量 15](#_Toc465237091)

[5.2 安装tomcat 15](#_Toc465237092)

[5.3 集成demo 16](#_Toc465237093)

# 前言

## 编写目的

本文档作为CIAP7.0版本中应用系统的认证集成手册，旨在指导校内应用系统实现校内统一身份认证和单点登录集成。

## 服务对象

|  |  |
| --- | --- |
| **读者对象** | **阅读范围** |
| 产品开发项目组人员 | 能熟悉掌握整个产品的流程和实施方法，能做到顺利完成产品项目或指导产品项目的实施。 |
| 产品维护人员 | 阅读全部内容 |
| 工程项目经理 | 阅读全部内容 |
| 产品部署实施人员 | 阅读全部内容 |

# 集成方式概述

将高校的各个业务系统与身份管理平台连接，通过身份管理平台实现用户身份认证和单点登录功能，这个过程即应用系统与身份管理平台的集成。CIAP7.0主要包括以下集成方式：（认证接口指CAS接口，代理接口暂时不支持）

* 认证接口：这是各个应用系统与身份管理平台集成的最主要的方式。各应用将认证接口的客户端开发包集成在各应用之中，替换自身原有独立的身份认证功能，通过身份管理平台实现身份认证和单点登录过程。该接口目前涵盖的类型包括JAVA、 PHP和 .NET(2.0+)，适合于不同语言和平台的应用程序。
* LDAP接口：对于选课选这样的高并发应用，我们提供LDAP接口，以满足认证的性能需求。该接口直接通过LDAP向应用系统提供认证服务，但是牺牲了单点登录功能。



# java认证接口集成

代理认证配置完后，均需至统一身份管理平台授权访问。先登录身份认证管理平台，在认证管理的认证应用模板，添加需代理认证的应用；添加完应用后，需给相应的组或者帐号授权，允许其访问该应用。

## 拷贝提供的jar

将“java/lib”下的cas-client-core-3.2.1.jar文件拷贝到客户端应用所在服务器中，并将存放这些jar包的路径设置到应用的classpath中。比如WebContent\WEB-INF\lib中

## 修改web.xml文件

在第三方应用的web.xml中添加以下内容：（加色部分需要修改为实际参数）

|  |
| --- |
| <!-- ==================== 单点登录开始 ================ -->  <!-- 用于单点退出，该监听器和过滤器用于实现单点登出功能，可选配置 -->  <listener>  <listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listener-class>  </listener>  <filter>  <filter-name>CAS Single Sign Out Filter</filter-name>  <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>CAS Single Sign Out Filter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- 该过滤器负责用户的认证工作，必须启用它 -->  <filter>  <filter-name>CASFilter</filter-name>  <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>  <init-param>  <param-name>casServerLoginUrl</param-name>  <!--这里的server是CAS服务端的登录地址,login为固定值-->  <param-value>http://ciap7.wisedu.com/authserver/login</param-value>  </init-param>  <init-param>  <param-name>serverName</param-name>  <!--这里是应用地址，注意是域名:端口或者ip:端口-->  <param-value>http://ssodemo.wisedu.com:8080</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CASFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- 该过滤器负责对Ticket的校验工作，必须启用它 -->  <filter>  <filter-name>CAS Validation Filter</filter-name>  <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>  <init-param>  <param-name>casServerUrlPrefix</param-name>  <!--这里的server是CAS服务端的地址,这里不要加login-->  <param-value>http://ciap7.wisedu.com/authserver</param-value>  </init-param>  <init-param>  <param-name>serverName</param-name>  <!--这里是应用地址，注意是域名:端口或者ip:端口-->  <param-value>http://ssodemo.wisedu.com:8080</param-value>  </init-param>  <!--去身份认证的校验的时候也需要加一个编码-->  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CAS Validation Filter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- 该过滤器负责实现HttpServletRequest请求的包裹，比如允许开发者通过HttpServletRequest的getRemoteUser()方法获得SSO登录用户的登录名。 -->  <filter>  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>  <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- =============== 单点登录结束 ================= --> |

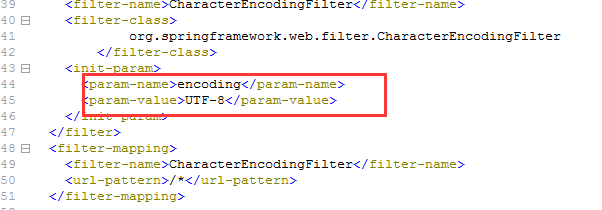
说明：

web项目的编码都encodeFilter的filter-mapping都是放到第一个的，这个执行顺序要注意的。Casclient客户端是不涉及到编码格式的，默认都是来自于项目的。

执行第一个getParameter()的时候，java将会按照编码分析所有的提交内容，而后续的getParameter()不再进行分析，所以setCharacterEncoding()无效。而对于GET方法提交表单是，提交的内容在URL中，一开始就已经按照编码分析提交内容，setCharacterEncoding()自然就无效。

所以请将如上的代码filter-mapping放到项目编码filter-mapping下面，保证最先执行项目编码过滤器。

注意：上面的配置要放在业务系统的web.xml文件中的编码下面，如下图所示部分是业务系统的项目编码。



## 获取用户信息

第三方应用在配置好web.xml之后，需要调整代码用集成之后的方式获取当前登录用户。获取用户信息的示例代码如下：

|  |
| --- |
| //cas-client-3.2.1版本集成  String uid = request.getRemoteUser();  String cn = "";  Principal principal = request.getUserPrincipal();  if(principal!=null && principal instanceof AttributePrincipal){  AttributePrincipal aPrincipal = (AttributePrincipal)principal;  //获取用户信息中公开的Attributes部分  Map<String, Object> map = aPrincipal.getAttributes();  // 获取姓名,可以根据属性名称获取其他属性  cn = (String)map.get("cn");  } |

## 集成应用退出

第三方应用在退出的时候需要重定向跳转到CIAP的退出地址，具体示例代码如下(红色部分需要根据实际情况调整)，被集成业务系统退出，必须用调用身份认证的退出地址，并且要清除业务系统自身的cookie。

|  |
| --- |
| session.invalidate();  // CIAP的退出地址，ciap7.wisedu.com为CIAP的域名 authserver为CIAP的上下文，logout为固定值  String casLogoutURL = "http://ciap7.wisedu.com/authserver/logout";  // service后面带的参数为应用的访问地址，需要使用URLEncoder进行编码  String redirectURL=casLogoutURL+"?service="+URLEncoder.encode("http://ssodemo.wisedu.com:8080/caslogin.jsp");  response.sendRedirect(redirectURL); |

## 白名单（认证不拦截）

在第三方应用的web.xml中添加以下内容：（加色部分需要修改为实际参数），符合白名单规则的地址，不被身份认证拦截。白名单的配置仅支持正则表达式匹配。

|  |
| --- |
| <filter>  <filter-name>CAS Authentication Filter</filter-name>  <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>  <init-param>  <param-name>casServerLoginUrl</param-name>  <param-value>http://authserver.xxx.edu.cn/authserver/login</param-value>  </init-param>  <!---仅支持正则表达式匹配，支持英文逗号分隔，可以配置多个-->  <init-param>  <param-name>casWhiteUrl</param-name>  <param-value>/demo.\*/index.jsp</param-value>  </init-param>  <init-param>  <param-name>serverName</param-name>  <param-value>http://172.16.7.50:8090</param-value>  </init-param>  </filter> |

## 认证接口工作过程

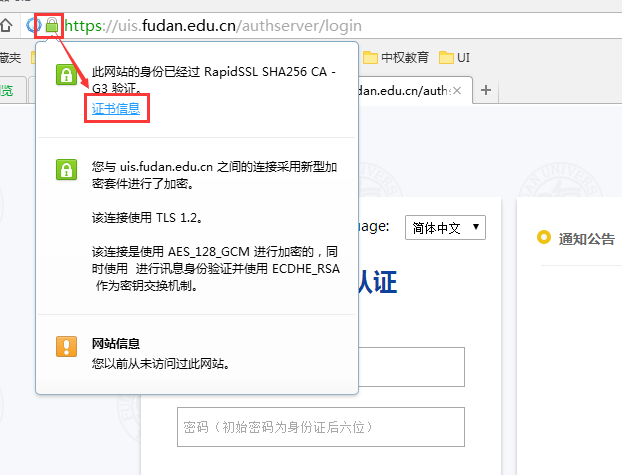


# 集成应用添加SSL证书信任

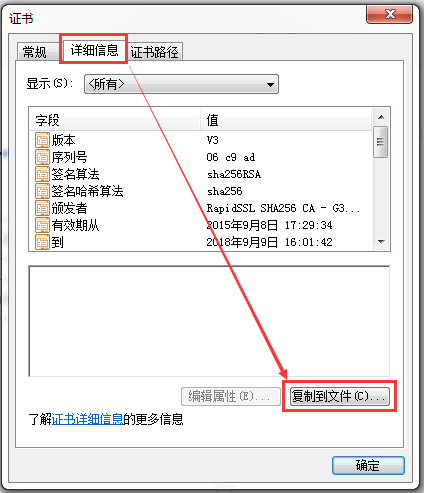
统一身份的集成，待集成的应用系统需要去统一身份平台验证令牌ticket，这个验证如果统一身份平台配置了https，则需要在第三方应用服务器导入SSL证书添加信任，一般情况下，java程序在该第三方应用所使用的jdk中把证书导入security\cacerts即可。

安装证书举例，360浏览器为例

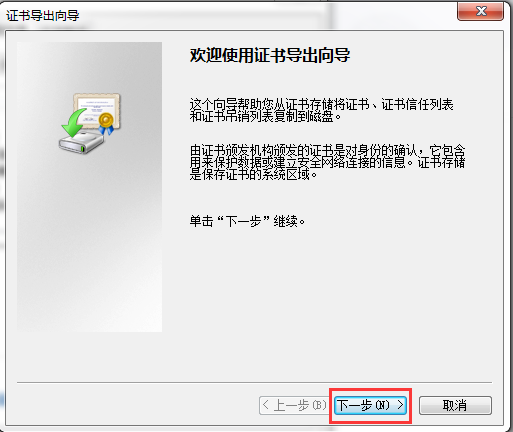
第一步：首先,需要用浏览器下载https证书，访问https://uis.fudan.edu.cn

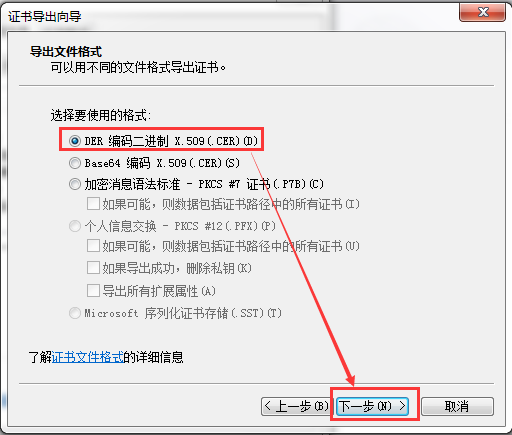


第二步：切换到证书的详细信息，并选择复制到文件

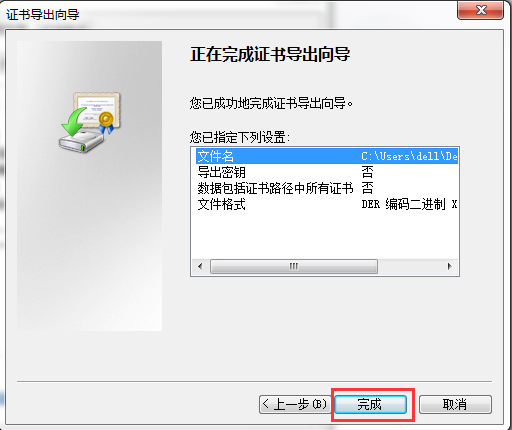


第三步：按证书导出向导，下一步选择证书格式，一般选择缺省的证书格式









第六步：复制证书到jdk环境下，例如：/opt/soft/fudan.cer

第七步：使用命令行 cd $JAVA\_HOME/security/cacerts

$JAVA\_HOME替换为你自己的JDK目录

|  |
| --- |
| /usr/java/jdk1.7.0\_80/jre/lib/security |

第八步：输入命令

|  |
| --- |
| keytool -import -alias cacerts -keystore cacerts -file /opt/soft/fudan.cer |

第九步:按提示输入cacerts证书库的密码，缺省为changeit

第十步:信任证书，导入完成。

第十一步:查看导入的证书，输入keytool -list -keystore cacerts

注: 更新证书前需输入keytool -delete -alias cacerts -keystore cacerts 先删除证书

# demo实例

## 安装jdk及配置环境变量

1、下载jdk，下载地址：

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

jdk的版本1.5,1.6,1.7都可以的，我选的是：



2、将下载的文件放到/opt目录下，之后解压到当前目录：

|  |
| --- |
| cd /opt  tar -zxf jdk-7u71-linux-x64.tar.gz |

修改文件夹名为jdk1.7。

3、找到/etc目录下的profile文件，在文件最后加上jdk的环境变量：

|  |
| --- |
| export JAVA\_HOME=/opt/jdk1.7  export PATH=$JAVA\_HOME/bin:$PATH  export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar |

4、为了使得配置快速生效，执行以下命令：

source /etc/profile

5、至此，jdk的安装全部结束，执行：

java -version

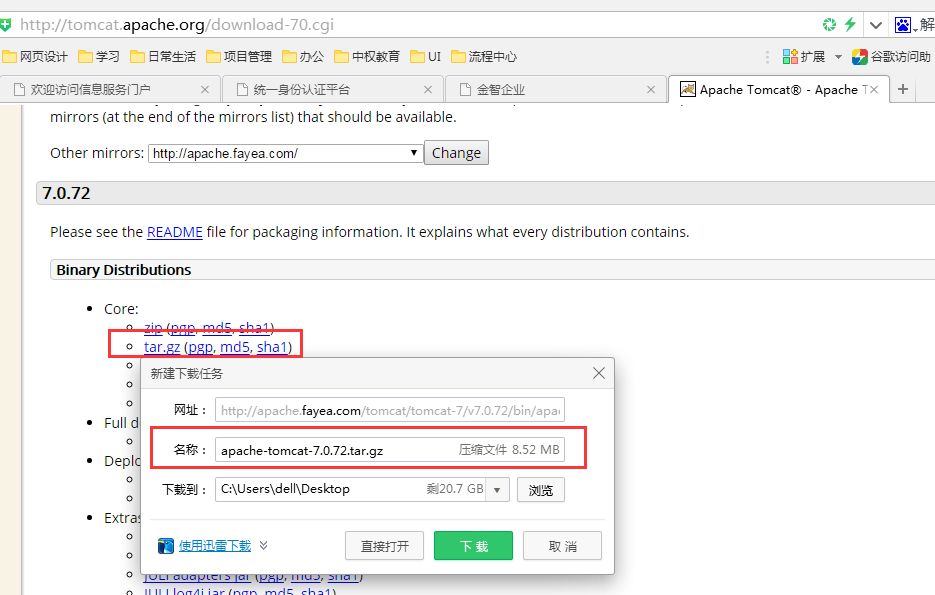
出现下图，则安装成功：



## 安装tomcat

1、下载tomcat，下载地址：http://tomcat.apache.org/download-70.cgi

【注意】:本手册中tomcat下载的版本7是，也可以用其他版本



2、将下载的.tar.gz格式的tomcat放在/opt下，解压。

cd /opt

tar -zxvf apache-tomcat-7.0.72.tar.gz

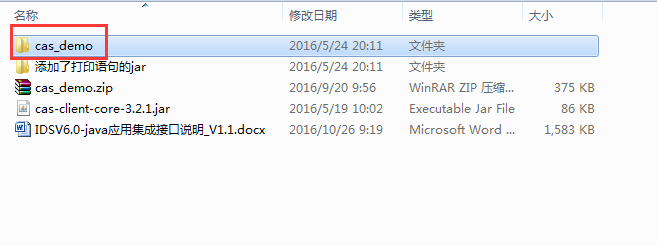
修改解压出来的文件夹名为tomcat7，改不改名字无所谓。

3、启动tomcat、停止tomcat

|  |
| --- |
| /opt/tomcat7/bin  ./startup.sh 启动tomcat  ./shutdown.sh 停止tomcat |

## 集成demo

1、将提供的cas\_demo放在/opt/tomcat7/webapps下。



2、修改/opt/tomcat7/webapps/cas\_demo/WEB-INF下的web.xml文件。

web.xml配置文件：

只需要修改以下几处：

第一处：

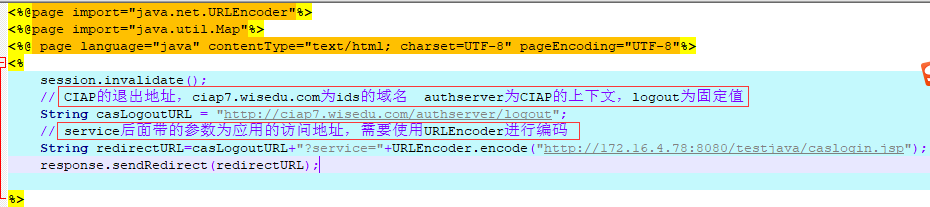


第二处：



3、修改demo的退出地址

修改cas\_demo目录中caslogout.jsp文件。



4、重新启动tomcat

|  |
| --- |
| /opt/tomcat7/bin  ./shutdown.sh  ./startup.sh |

5、在Mions Console管控台注册应用，并授权用户。

1)添加应用：





2）CAS协议配置





3）启用CAS协议



4）授权应用给用户：





6、访问集成的demo

<http://172.16.6.138:7006/cas_demo/caslogin.jsp>

并输入授权的用户，我这里是用户testlxf。

