统一认证系统集成文档

北京智麟科技有限公司

一、 概述

1. 用途

本文档定义了一组 API 接口，其他系统（如财务系统、教务系统）可以通过调用接口向统一身份认证系统进行集成。

2. 适用版本

本文档定义的接口规范，对全版本统一身份认证系统有效。

3. 通信协议

本文档中，统一身份认证认证服务是通信的服务器端（简称"认证服务器"），调用统一身份认证服务接口的其他系统是第三方应用服务（简称"第三方应用"），使用统一身份认证服务的使用者是师生用户（简称"用户"）。第三方应用和认证服务器通过 HTTP/HTTPS 协议通信，第三方应用使用 HTTP Get/Post 向认证服务器发送请求，认证服务器返回操作结果或用户信息给第三方应用。

4. 调用方式

接口的请求地址是个 URL 地址，每个接口 URL 地址都包括接口位置和请求，例如：

https://sso.edu.cn/serviceValidate?service=abc&ticket=st-123

其中，"https://sso.edu.cn/serviceValidate"是接口位置，本文档中所有"**https://sso.edu.cn**"需替换成实际应用地址。例如："**http://cas.edu.cn/tpass/.....**"

"/serviceValidate"代表服务器接口服务在 Web 上的部署的位置。符号?后面是参数，以 name=value 的形式体现。

当 URL 请求参数值中包含 URL 地址保留字符时，应对参数值进行 URL 编码。URL 地址保留字符包括："**;/?:@&=+$,**"，编码规则是将保留字符替换

成%hexhex 形式。具体参见"RFC2396: Uniform Resource Identifiers (URI):Generic Syntax"。

有关 URL 地址的编码规范，请参考 HTTP 1.1 规范。

二、 页面跳转集成接口

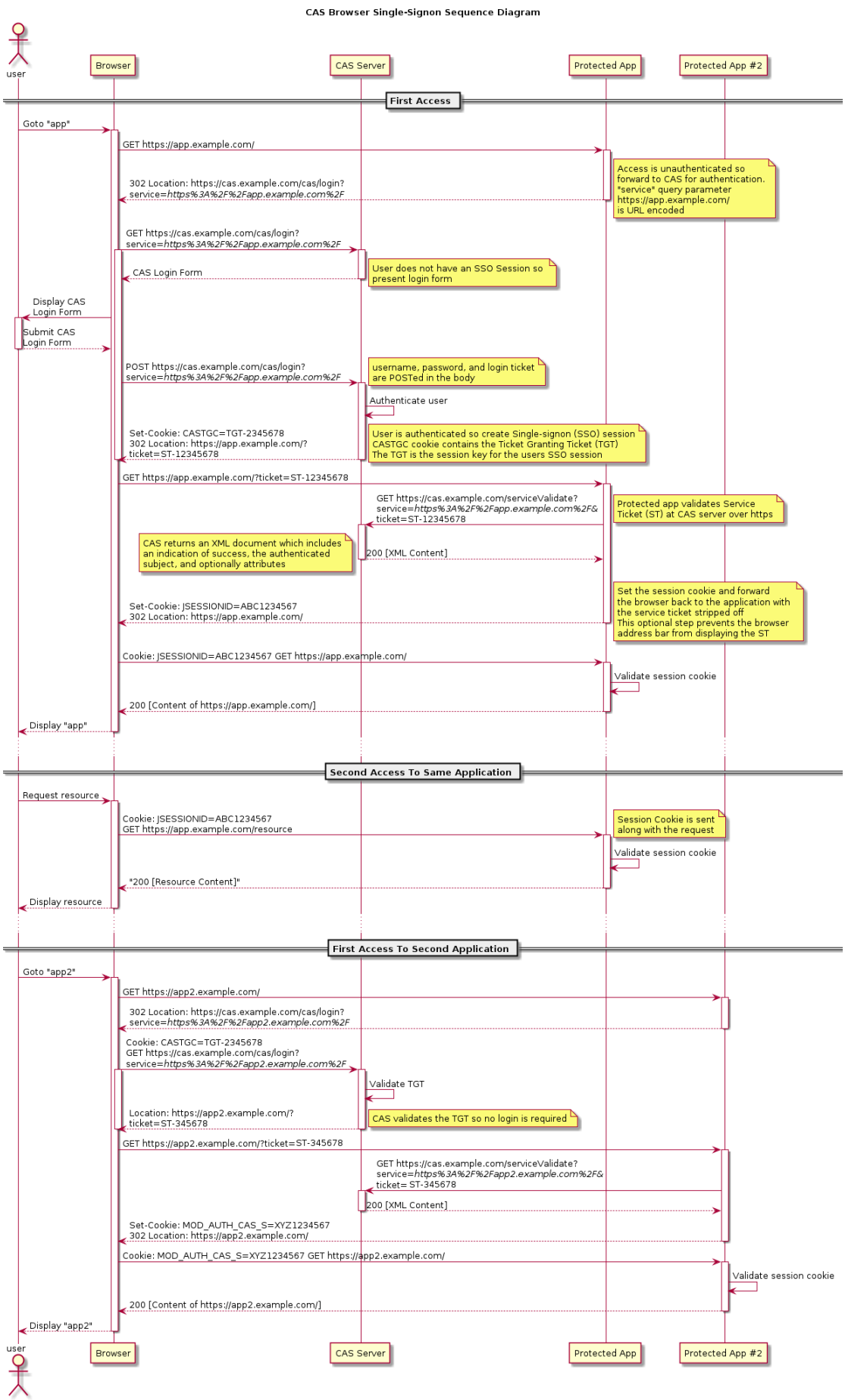
页面跳转登录方式为推荐方式，同等条件下建议使用该方式进行集成。

1. 登录流程

页面跳转集成兼容 CAS 2.0、CAS 3.0 协议标准，流程如下图。图中包含三种场景下的登录流程：首次登陆、登录后第二次访问同一个第三方应用、登录后首次访问第二个第三方应用。

首次登陆流程为：

1. 用户通过浏览器打开第三方应用，第三方应用验证用户登录状态（如通过 session cookie 判断），判断用户未登陆后，将用户重定向到认证服务器；
2. 认证服务器验证用户登录状态，判断用户未登录后，向用户展示登录页面；
3. 用户完成登录操作；
4. 认证服务器设置用户浏览器 cookie 的 CASTGC 内容，并重定向用户到第三方应用；
5. 第三方应用从用户请求中获取 ticket，并使用 ticket 调用认证服务器验证接口，获取用户信息；


CAS Browser Single-Signon Sequence Diagram
user
Browser
CAS Server
Protected App
Protected App #2
First Access
Goto "app"
GET https://app.example.com/
Access is unauthenticated so forward to CAS for authentication. "service" query parameter https://app.example.com/ is URL encoded
302 Location: https://cas.example.com/cas/login?service=https%3A%2F%2Fapp.example.com%2F
GET https://cas.example.com/cas/login?service=https%3A%2F%2Fapp.example.com%2F
User does not have an SSO Session so present login form
CAS Login Form
Display CAS Login Form
Submit CAS Login Form
POST https://cas.example.com/cas/login?service=https%3A%2F%2Fapp.example.com%2F
username, password, and login ticket are POSTed in the body
Authenticate user
User is authenticated so create Single-signon (SSO) session
CASTGC cookie contains the Ticket Granting Ticket (TGT)
The TGT is the session key for the users SSO session
Set-Cookie: CASTGC=TGT-2345678
302 Location: https://app.example.com/?ticket=ST-12345678
GET https://app.example.com/?ticket=ST-12345678
GET https://cas.example.com/serviceValidate?service=https%3A%2F%2Fapp.example.com%2F&ticket=ST-12345678
Protected app validates Service Ticket (ST) at CAS server over https
CAS returns an XML document which includes an indication of success, the authenticated subject, and optionally attributes
200 [XML Content]
Set the session cookie and forward the browser back to the application with the service ticket stripped off
This optional step prevents the browser address bar from displaying the ST
Set-Cookie: JSESSIONID=ABC1234567
302 Location: https://app.example.com/
Cookie: JSESSIONID=ABC1234567 GET https://app.example.com/
Validate session cookie
200 [Content of https://app.example.com/]
Display "app"
Second Access To Same Application
Request resource
Cookie: JSESSIONID=ABC1234567
GET https://app.example.com/resource
Session Cookie is sent along with the request
Validate session cookie
"200 [Resource Content]"
Display resource
First Access To Second Application
Goto "app2"
GET https://app2.example.com/
302 Location: https://cas.example.com/cas/login?service=https%3A%2F%2Fapp2.example.com%2F
Cookie: CASTGC=TGT-2345678
GET https://cas.example.com/cas/login?service=https%3A%2F%2Fapp2.example.com%2F
Validate TGT
Location: https://app2.example.com/?ticket=ST-345678
CAS validates the TGT so no login is required
GET https://app2.example.com/?ticket=ST-345678
GET https://cas.example.com/serviceValidate?service=https%3A%2F%2Fapp2.example.com%2F&ticket= ST-345678
200 [XML Content]
Set-Cookie: MOD_AUTH_CAS_S=XYZ1234567
302 Location: https://app2.example.com/
Cookie: MOD_AUTH_CAS_S=XYZ1234567 GET https://app2.example.com/
Validate session cookie
200 [Content of https://app2.example.com/]
Display "app2"

6. 第三方应用使用获取到的信息（如学工号、姓名）完成系统内登陆逻辑（如设置 session）。

登录后第二次访问同一个第三方应用流程为：

1. 用户通过浏览器打开第三方应用，第三方应用验证用户登录状态（如通过 session cookie 判断）；
2. 第三方应用判断用户已登陆，向用户展示内容页面。

登录后首次访问第二个第三方应用流程为：

1. 用户通过浏览器打开第三方应用，第三方应用验证用户登录状态（如通过 session cookie 判断），判断用户未登陆后，将用户重定向到认证服务器；
2. 认证服务器通过 CASTGC 验证用户登录状态，判断内容有效后，重定向用户到第三方应用；
3. 第三方应用从用户请求中获取 ticket，并使用 ticket 调用认证服务器验证接口，获取用户信息；
4. 第三方应用使用获取到的信息（如学工号、姓名）完成系统内登陆逻辑（如设置 session）。

2. 登录接口/login

地址	https://sso.edu.cn/login		
调用方式	GET		
参数	service	必填	URL 编码的第三方应用接口地址，用户在/login 接口完成登录后，系统会将其重定向到的该地址，此接口由第三方应用提供，要在服务管理的白名单内，接口需要完成的操

			作： 1. 从请求中获取 ticket 参数； 2. 使用 ticket 调用票据验证接口/serviceValidate 获取用户信息（ticket 一次生效）； 3. 完成用户登录（如写 session）。
调用场景	第三方应用在判断用户未登录后，将用户重定向到该地址		

示例：

https://sso.edu.cn/login?service=http%3a%2f%2fcaiwu.edu.cn

用户完成登录后，将重定向到：

http://caiwu.edu.cn/sso?ticket=ST-123456

3. 登出接口/logout

地址	https://sso.edu.cn/logout
调用方式	GET
调用场景	用户在第三方应用点击注销/登出按钮后，第三方应用： 1. 调用后台接口，使用户 session 失效； 2. 重定向用户到该地址，使统一认证登录状态失效。

示例：

https://sso.edu.cn/logout

4. 票据验证接口/serviceValidate

地址	https://sso.edu.cn/serviceValidate （CAS 2.0 协议接口） https://sso.edu.cn/p3/serviceValidate （CAS 3.0 协议接口）
调用方式	GET

参数	service	必填	同登录接口/login 的 service 参数
	ticket	必填	第三方应用获取到的 ticket，如 ST-123456
返回值	XML 格式编码的调用结果，如果 ticket 验证通过，将返回用户信息，否则返回错误信息		
调用场景	第三方应用获取到 ticket 后，调用该接口获取用户信息		

错误码包含：

1. **INVALID_REQUEST** - 请求参数错误
2. **INVALID_TICKET** - ticket 验证失败
3. **INVALID_SERVICE** - ticket 验证成功，但是 service 不在白名单中
4. **INTERNAL_ERROR** - 认证服务内部错误

示例：

https://sso.edu.cn/serviceValidate?service=http%3a%2f%2fcaiwu.edu.cn&ticket=ST-123456

验证成功后，将返回：

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
    <cas:authenticationSuccess>
        <cas:user>username</cas:user>
        <cas:name>张三</cas:name>
        <cas:employeeNumber>12345</cas:employeeNumber>
    </cas:authenticationSuccess>
</cas:serviceResponse>
```

验证失败后，将返回：

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
    <cas:authenticationFailure code="INVALID_TICKET">
        Ticket ST-123456 not recognized
    </cas:authenticationFailure>
</cas:serviceResponse>
```

三、 RESTful 集成接口

RESTful 接口用于无法使用页面跳转登录的场景。

1. 登录流程

RESTful 登录流程包含两种主要场景：首次登陆、再次登录。

首次登录流程如下：

1. 用户在第三方应用输入用户名、密码；
2. 第三方应用使用用户名、密码调用登录接口获取 TGT（TGT 长期有效，第三方应用可缓存该内容，以实现后续的免密码登录）；
3. 第三方应用使用 TGT 调用票据接口，获取 ticket；
4. 第三方应用使用 ticket 调用认证服务器验证接口，获取用户信息，完成登录。

再次登录流程如下：

1. 用户打开第三方应用，应用取出缓存的 TGT；
2. 第三方应用使用 TGT 调用票据接口，获取 ticket；
3. 第三方应用使用 ticket 调用认证服务器票据验证接口，获取用户信息，完成登录。

2. 登录接口/v1/tickets

地址	https://sso.edu.cn/v1/tickets		
调用方式	POST		
	'Content-type': 'Application/x-www-form-urlencoded'		
参数	username	必填	用户的用户名
	password	必填	用户的明文密码
返回值	XML 格式编码的调用结果，如果用户名、密码验证通过，将返回 TGT，否则返回错误信息		
调用场景	第三方应用使用用户输入的用户名、密码调用该接口获取 TGT		

注意：调用该接口的 IP 需要在服务的 IP 白名单中，否则无法调用。

示例：

POST https://sso.edu.cn/v1/tickets HTTP/1.0
'Content-type': 'Application/x-www-form-urlencoded'
username=12345&password=password

验证成功后，将返回：

201 Created
Location: http://www.whatever.com/cas/v1/tickets/{TGT}

3. 票据接口/v1/tickets/{TGT}

地址	https://sso.edu.cn/v1/tickets/{TGT}		
调用方式	POST		
	'Content-type': 'Application/x-www-form-urlencoded'		
	service	必填	URL 编码的第三方应用接口地址，此地址要在服务管理的白名单内
返回值	XML 格式编码的调用结果，如果 TGT、service 验证通过，将返回 ticket，否则返回错误信息		
调用场景	第三方应用使用 TGT 调用该接口获取 ticket		

示例：

POST https://sso.edu.cn/v1/tickets/TGT-123456 HTTP/1.0
'Content-type': 'Application/x-www-form-urlencoded'
service=http%3a%2f%2fcaiwu.edu.cn

验证成功后，将返回：

200 OK
ST-123456

4. 票据验证接口

同页面跳转登录的票据验证接口。

附件 1：页面跳转登录 Tomcat 项目示例

1. 导入依赖文件

以 Maven 的 pom.xml 为例，添加依赖如下：

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
</dependency>
<dependency>
    <groupId>org.jasig.cas.client</groupId>
    <artifactId>cas-client-core</artifactId>
    <version>3.4.1</version>
</dependency>
```

2. web.xml filter 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_4_0.xsd"
          version="4.0">
    <filter>
        <filter-name>CASFilter</filter-name>
        <filter-class>org.jasig.cas.client.authentication. AuthenticationFilter</filter-class>
        <init-param>
            <param-name>casServerLoginUrl</param-name>
            <!--这是认证服务端登录地址，需要根据学校现状做调整-->
            <param-value> https://sso.edu.cn/login</param-value>
        </init-param>
        <init-param>
            <param-name>serverName</param-name>
            <!--这是第三方应用地址，即当前系统的访问地址，需要修改为你的系统-->
            <param-value>http://caiwu.edu.cn</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CASFilter</filter-name>
        <url-pattern>/protectedUrls/*</url-pattern>
    </filter-mapping>

    <!-- 该过滤器负责对 Ticket 的校验工作，必须启用 -->
    <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>
                <!--这是认证服务端地址前缀，需要根据学校现状做调整-->
                <param-value> https://sso.edu.cn</param-value>
            </init-param>
            <init-param>
                <param-name>serverName</param-name>
                <!--这是第三方应用地址，即当前系统的访问地址，需要修改为你的系统-->
                <param-value> http://caiwu.edu.cn</param-value>
            </init-param>
        </filter>
        <filter-mapping>
            <filter-name>CAS Validation Filter</filter-name>
            <url-pattern>/protectedUrls/*</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 getUserPrincipal()方法获得 SSO 登录用户的信息。
        -->
        <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>/protectedUrls/*</url-pattern>
        </filter-mapping>

    <!--
    示例 1：通过 filter 根据单点登录的结果设置本系统的登录状态信息
    若不通过 filter 进行登录，可注释掉以下内容
    -->
        <filter>
            <display-name>DemoFilter</display-name>
            <filter-name>DemoFilter</filter-name>
            <filter-class>cn.edu.sso.demo.filter.DemoFilter</filter-class>
        </filter>
        <filter-mapping>
            <filter-name>DemoFilter</filter-name>
            <url-pattern>/protectedUrls/filter_login.jsp</url-pattern>
```

```
        <!--<url-pattern>/protectedUrls/*</url-pattern>-->
    </filter-mapping>
</web-app>
```

3. 通过 filter 完成登录

这里展示了如何通过自定义的 filter 完成登录，filter 需要在 web.xml 中进行注册（示例 1 部分）。

```
package cn.edu.sso.demo.filter;



import org.jasig.cas.client.authentication.AttributePrincipal;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.Map;

public class DemoFilter implements Filter {

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
FilterChain filterChain) throws IOException, ServletException {

        // 获取用户拓展属性
        AttributePrincipal principal = (AttributePrincipal)
((HttpServletRequest)servletRequest).getUserPrincipal();
        Map attributes = principal.getAttributes();

        HttpSession session = ((HttpServletRequest) servletRequest).getSession();
        if (session.getAttribute("user") == null) {
            // 完成登录行为
            session.setAttribute("user", attributes);
            System.out.println("login in success from filter");
        }

        filterChain.doFilter(servletRequest, servletResponse);

    }

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {}
```

```
    @Override
    public void destroy() {}
}
```

4. 通过 jsp 完成登录

```
<%@ page import="org.jasig.cas.client.authentication.AttributePrincipal" %>
<%@ page import="java.util.Map" %>
<% String basePath = request.getScheme() + "://" + request.getServerName() + ":" + request.getServerPort() + request.getContextPath(); %>
<% AttributePrincipal principal = (AttributePrincipal)request.getUserPrincipal(); %>
<% Map attributes = principal.getAttributes(); %>
<%-- 完成登录 --%>
<% if (session.getAttribute("user") == null) { session.setAttribute("user", attributes); } %>
<% System.out.println("login in success from page"); %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page trimDirectiveWhitespaces="true" %>
<html>
  <head>
    <title>SSO JSP Login</title>
  </head>
  <body>
      <h1>遍历所有开放的属性：</h1>
      <table border="1">
        <% for (Object key : attributes.keySet()) { %>
        <tr>
          <td><%=key%></td><td><%=attributes.get(key)%></td>
        </tr>
        <% } %>
      </table>

      <h1>注销：</h1>
      <button onclick="logout()">登出</button>
  </body>

  <script>
      function logout() {
          // 使本系统登录状态失效
          var httpRequest = new XMLHttpRequest();
          // 先调用本系统注销接口
          httpRequest.open("GET", "<%=basePath%>/logout.jsp");
          httpRequest.send();
          httpRequest.onreadystatechange = function (ev) {
              if (httpRequest.readyState === 4 && httpRequest.status === 200) {
```

```
                    console.log(httpRequest.responseText);
                    if (httpRequest.responseText === "success") {
                        // 本系统注销成功后，调用 sso 的注销
                        window.open("https://sso.edu.cn/logout", "_self");
                    }
                }
            };
        }
    </script>
</html>
```

附件 2：RESTful 登录 Java 示例

```
package restfulAPI;

import java.io.IOException;
import java.net.URLEncoder;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.NameValuePair;

public class TstRestfulAPI {

    public static String getTicket(final String server, final String username, final String password,
            final String service) {
        notNull(server, "server must not be null");
        notNull(username, "username must not be null");
        notNull(password, "password must not be null");
        notNull(service, "service must not be null");

        return getServiceTicket(server, getTicketGrantingTicket(server, username, password), service);
    }

    /**
     * 取得 ST
     * @param server
     * @param ticketGrantingTicket
     * @param service
```

```
     */
    private static String getServiceTicket(final String server, final String ticketGrantingTicket,
final String service) {
         if (ticketGrantingTicket == null)
              return null;

         final HttpClient client = new HttpClient();

         final PostMethod post = new PostMethod(server + "/" + ticketGrantingTicket);

         post.setRequestBody(new NameValuePair[] { new NameValuePair("service", service) });

         try {
              client.executeMethod(post);

              final String response = post.getResponseBodyAsString();

              switch (post.getStatusCode()) {
              case 200:
                   return response;

              default:
                   warning("Invalid response code (" + post.getStatusCode() + ") from CAS
server!");
                   info("Response (1k): " + response.substring(0, Math.min(1024,
response.length())));
                   break;
              }
         }

         catch (final IOException e) {
              warning(e.getMessage());
         }

         finally {
              post.releaseConnection();
         }

         return null;
    }

    /**
     * @param server
     * @param username
```

```
      * @param password
      */
     private static String getTicketGrantingTicket(final String server, final String username, final
String password) {
          final HttpClient client = new HttpClient();

          final PostMethod post = new PostMethod(server);

          post.setRequestBody(new NameValuePair[] { new NameValuePair("username", username),
                    new NameValuePair("password", password) });

          try {
               client.executeMethod(post);

               final String response = post.getResponseBodyAsString();
               info("TGT="+response);
               switch (post.getStatusCode()) {
                    case 201: {
                         final                 Matcher                 matcher                 =
Pattern.compile(".*action=\".*/(.*?)\".*").matcher(response);

                         if (matcher.matches())
                              return matcher.group(1);

                         warning("Successful ticket granting request, but no ticket found!");
                         info("Response    (1k):    "    +    response.substring(0,    Math.min(1024,
response.length())));
                         break;
                    }
                    default:
                         warning("Invalid response code (" + post.getStatusCode() + ") from CAS
server!");
                         info("Response    (1k):    "    +    response.substring(0,    Math.min(1024,
response.length())));
                         break;
                    }
          }
          catch (final IOException e) {
               warning(e.getMessage());
          }
          finally {
               post.releaseConnection();
          }
```

```
          return null;
     }

     private static void ticketValidate(String serverValidate, String serviceTicket, String
service) {
          notNull(serviceTicket, "paramter 'serviceTicket' is not null");
          notNull(service, "paramter 'service' is not null");

          final HttpClient client = new HttpClient();
          GetMethod post = null;

          try {
               post                                        =                                        new
GetMethod(serverValidate+"?"+"ticket="+serviceTicket+"&service="+URLEncoder.encode(service, "UTF-
8"));
               client.executeMethod(post);

               final String response = post.getResponseBodyAsString();
               info(response);
               switch (post.getStatusCode()) {
                    case 200: {
                         info("成功取得用户数据");
                    }
                    default: {
                    }
               }

          } catch (Exception e) {
               warning(e.getMessage());
          } finally {
               //释放资源
               post.releaseConnection();
          }

     }

     private static void notNull(final Object object, final String message) {
          if (object == null)
               throw new IllegalArgumentException(message);
     }

     public static void main(final String[] args) throws Exception {
          //server 不可修改
          final String server = "https://sso.edu.cn/v1/tickets";
```

```
        //用户名
        final String username = "username";
        //密码
        final String password = "password";
        //此地址要在服务管理的白名单内
        final String service = "http://caiwu.edu.cn/";
        //serviceValidate 接口地址
        final String proxyValidate = "https://sso.edu.cn/serviceValidate";

        ticketValidate(proxyValidate, getTicket(server, username, password, service), service);

    }

    private static void warning(String msg) {
        System.out.println(msg);
    }

    private static void info(String msg) {
        System.out.println(msg);
    }

}
```