单点接口文档

1.1 CAS 认证标准服务接口

1.1.1 文档概述

目的

本文档详细定义了统一登录平台系统所提供的 CAS 认证服务接口的接口规范。当应用系统在与统一登录平台进行 CAS 认证集成时，需要了解该文档的内容，并根据该文档定义的接口规范进行相应接口的开发或调整。

范围

本文档包括统一登录平台系统与接入到统一登录平台系统的应用在对接过程中所使用的接口与规范的设计。

读者

本文档读者为本项目所有成员及集成至统一登录平台系统的应用方所有项目成员、项目经理以及统一登录平台系统架构师。

定义、术语和缩略语

名词/术语	英文全称	描述
SSO	Single Sign-On	单点登录，一次登录访问所有应用
IAM	Identity and Access Management	身份与访问控制
CAS	Central Authentication Service	CAS 协议是一种专门为 CAS 开发的简单而强大的基于票证的协议

1.1.2 应用集成 SSO 系统准备

周边应用准备

应用系统(需要接入 CAS 认证的应用)准备认证成功 service 地址：

回调地址（接收 code 验证码的地址及认证成功跳转地址）

属性名	说明
service	http://XXX.XXX.XXX.XXX/XX/XX

向 SSO 系统申请注册应用系统并且提供跳转地址给 SSO 系统

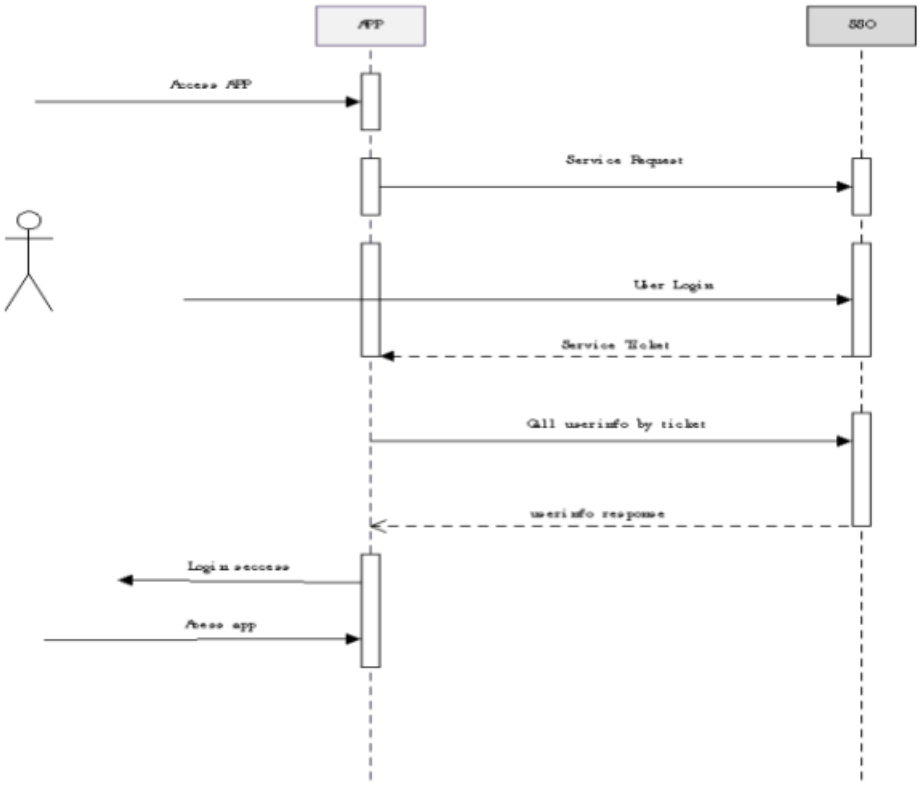
SSO 系统准备

根据应用系统提供的 service 地址，注册应用系统

1.2.3 认证集成过程

CAS 认证过程

下图为 CAS 认证交互过程图：



1,用户访问应用系统地址浏览器重定向请求到 SSO 的认证地址，并且提供相应的参数

2,CAS 完成用户认证，并为 app 提供 ticket。

3,浏览器跳转到 app 并提供 ticket 参数。

4,用户在获取访问 ticket 后，通过调用接口获取认证用户信息。

5,应用获取用户信息，认证完成，用户正常访问 app。

请求 URL 实例及参数说明

组合认证地址浏览器请求认证及参数,重定向请求认证中心：

URL 示例： https://sso.cdutcm.edu.cn/esc-sso/login?service=http://XXX.XXX.XXX.XXX/XX/XX

属性名称	说明	是否必须
service	应用注册服务地址	必须

SSO 系统通过 redirect_uri 浏览器重定向请求反馈 ticket 给应用：

URL 示例 http://XXXX.XXXX.XXXX.XXXX/XXX/XXX?ticket=ST-123456abcdef
获取参数：

属性名称	说明	是否必须
ticket	获取用户信息的凭证	必须

根据 ticket，向认证中心请求获取用户信息
URL 示例：https://sso.cdutcm.edu.cn/esc-sso/p3/serviceValidate?service=http://XXXX.XXXX.XXXX.XXXX/X XX/XXX&ticket=ST-123456abcdef
请求参数：
请求方式：get
请求协议：http 协议

属性名称	说明	是否必须
service	应用注册服务地址	必须
ticket	获取用户信息的凭证	必须
format	返回的格式,默认是 xml 可选值： json	非必须

返回参数：

属性名称	说明	是否必须
user	用户登录的主账号	必须

attributes->phone	用户的手机号	非必须
attributes->email	用户的邮箱	非必须
attributes->username	用户的应用账号	非必须

解析返回数据得到用户信息,应用系统根据这个字段匹配自身平台的账号数据，让用户通过认证。
Attribute 字段里的 phone，email，username 字段是可以配置的
返回样例（xml）

```
<!- ParaView version: 1.1.1_debug  "2018-12-03T09:37:12.732Z" ->
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
    <cas:authenticationSuccess>
        <cas:user>sysadmin</cas:user>
        <cas:attributes>
            <cas:isFromNewLogin>false</cas:isFromNewLogin>
            <cas:authenticationDate>

2019-12-06T23:28:44.443+08:00[Asia/Shanghai]

</cas:authenticationDate>
            <cas:phone>18501612606</cas:phone>

<cas:authenticationMethod>QueryDatabaseAuthenticationHandler</cas:authentication
Method>

<cas:successfulAuthenticationHandlers>QueryDatabaseAuthenticationHandler
            </cas:successfulAuthenticationHandlers>

<cas:longTermAuthenticationRequestTokenUsed>false</cas:longTermAuthenticationReq
uestTokenUsed>
            <cas:email>sysadmin@paraview.cn</cas:email>
            <cas:username>sysadmin</cas:username>
        </cas:attributes>
    </cas:authenticationSuccess>
</cas:serviceResponse>
```

返回样例（json）

```
{
  "serviceResponse": {
    "authenticationSuccess": {
      "user": "sysadmin",
      "attributes": {
        "isFromNewLogin": [
          true
        ],
        "authenticationDate": [
```

```
            1575645600.289000000
          ],
          "phone": [
            "18501612606"
          ],
          "authenticationMethod": "QueryDatabaseAuthenticationHandler",
          "successfulAuthenticationHandlers": [
            "QueryDatabaseAuthenticationHandler"
          ],
          "longTermAuthenticationRequestTokenUsed": [
            false
          ],
          "email": [
            "sysadmin@paraview.cn"
          ],
          "username": [
            "sysadmin"
          ]
        }
      }
    }
  }
}
```


1.2 OAuth2.0 协议标准单点接口

1.2.1 文档概述

目的

本文档详细定义了统一登录平台系统所提供的 OAuth2.0 认证服务接口的接口规范。当应用系统在与统一登录平台进行 OAuth2.0 认证集成时，需要了解该文档的内容，并根据该文档定义的接口规范进行相应接口的开发或调整。

范围

本文档包括统一登录平台系统与接入到统一登录平台系统的应用在对接过程中所使用的接口与规范的设计。

文档中的**${SSO_DOMAIN}**为统一身份认证中心的域名，需要使用者根据实际环境进行调整。

OAuth2.0 协议说明

本项目采用 OAuth2.0 协议中的 **Authorization Code Grant** 模式，相关技术细节参考官网

https://datatracker.ietf.org/doc/html/rfc6749#section-4.1

1.2.2 认证过程

组合认证地址浏览器请求认证及参数,重定向请求认证中心：
请求地址（浏览器 redirect）

${SSO*DOMAIN}/esc-sso/oauth2.0/authorize?client*_id=XXXX&response*_type=code*&redirect_uri=http%3A%2F%2FXXXX.XXXX.XXXX.XXXX/XXX/XXX%3A8080%2FXXXX%2FXXXX&oauth_timestamp=1489739502583&state=http%3A%2F%2Fwww.app1.com%2Ftodu%2F1w2 341123

参数说明

参数名	说明	必填
client_id	客户端 ID，由 IAM 系统管理员分配	是
response_type	其值固定为 code	是
redirect_uri	回调地址，由被集成的系统提供，并在 IAM 系统中注册。需要转义。	是
oauth_timestamp	时间戳，格式为当前毫秒数	否
state	额外参数，如果传递 http 需要转义	否

SSO 系统通过 redirect_uri 浏览器重定向请求反馈 code 给应用：

http://XXXX.XXXX.XXXX.XXXX/XXX/XXX?code=123456abcdef&state=http%3A%2F

%2Fwww.app1.com%2Ftodu%2F1w2341123

参数说明

参数名	说明	必填
code	由 IAM 生成的 code，一次有效，使用后即作废	是
state	如果在请求时传递了此参数，IAM 也会原样带回	否

二：应用获取的 code，跟认证中心交互获取 token，使用 http 方式请求 token 接口

请求地址（POST）

${SSO*DOMAIN*}*/esc-sso/oauth2.0/accessToken*

参数说明

参数名	说明	必填
grant_type	固定为 authorization_code	是
oauth_timestamp	时间戳，格式为当前毫秒数	是
client_secret	客户端密钥，IAM 系统管理员分配	是
client_id	应用注册 ID ，IAM 系统管理员分配	是
code	上一步接收到的 code 值	是
redirect_uri	回调地址，由被集成的系统提供，并在 IAM 系统中注册。需要转义。	是

返回值

```
{
    "access_token": "AT-7-3KbHW3iqrQWBLnCAEn0xgVuOIkPgyCdSjf7",
    "token_type": "bearer",
    "expires_in": 86400
}
```

返回值参数说明

参数名	说明
access_token	access_token
token_type	固定为 bearer
expires_in	token 过期时间，单位秒

代码示例

```
/**
     * 获取 token
     **/
    private String getToken(String code) {
        String url = tokenUrl + "?grant_type=" + grantType
                + "&oauth_timestamp=" + System.currentTimeMillis()
                + "&client_id=" + clientId
                + "&client_secret=" + clientSecret
                + "&code=" + code
                + "&redirect_uri=" + redirectUrl;
        String result = HttpUtil.post(url, (String) null);
        return result;
    }
```

注：1，tokenUrl 指的是上面获取 token 的请求地址

2，HttpUtil 用的是第三方 jar 中提供的工具方法：

```
<dependency>
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-all</artifactId>
            <version>5.7.3</version>
        </dependency>
```

3，其余参数见接口参数说明

三：根据 token，向认证中请求获取用户信息

请求地址（GET）

${SSO*DOMAIN}/esc-sso/oauth2.0/profile?*access_token=b1d508f3-32e7-4d6d-ac62-87836406704c

参数说明

参数名	说明
access_token	上一步获取到的 access_token

返回结果


```
{
attributes: {
token_expired: "7200",
token_gtime: "1622859245756",
account_no: "test"
},
id: "test"
}
```

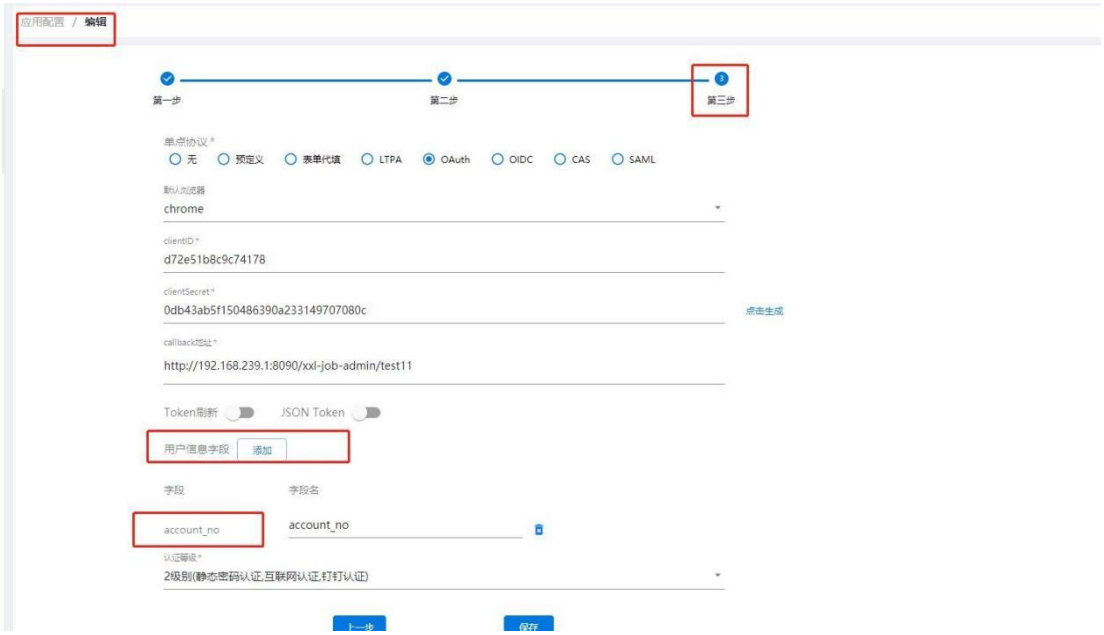

返回结果参数说明

参数名	说明
id	当前登录用户的 UID
account_no	下游系统的账号(用于账号委托,公用账号时使用)

代码实例

统一身份认证系统中

应用配置(配置中可以生成 client_id,client_secret,以及配置 account_no



第三方应用系统

拦截器拦截所有请求

未登录时,重定向到统一身份认证系统

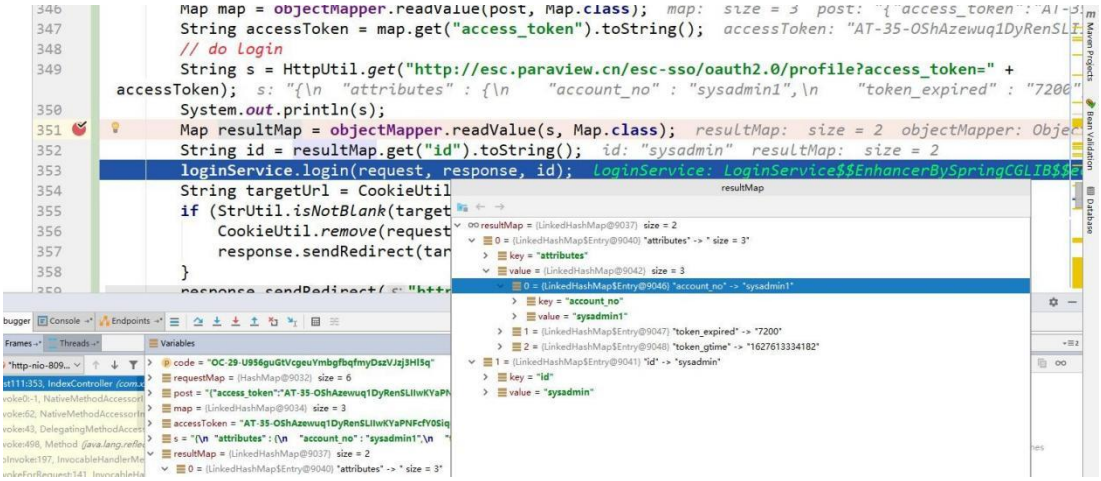
```
            } else {
                response.sendRedirect( s: "http://esc.paraview.cn/esc-sso/oauth2
.0/authorize?client_id=d72e51b8c9c74178&response_type=code&oauth_timestamp="+System
.currentTimeMillis()+"&redirect_uri=" + URLEncoder.createDefault().encode( path: "http://192.168.239
.1:8090/xxl-job-admin/test11", StandardCharsets.UTF_8));
            }
```

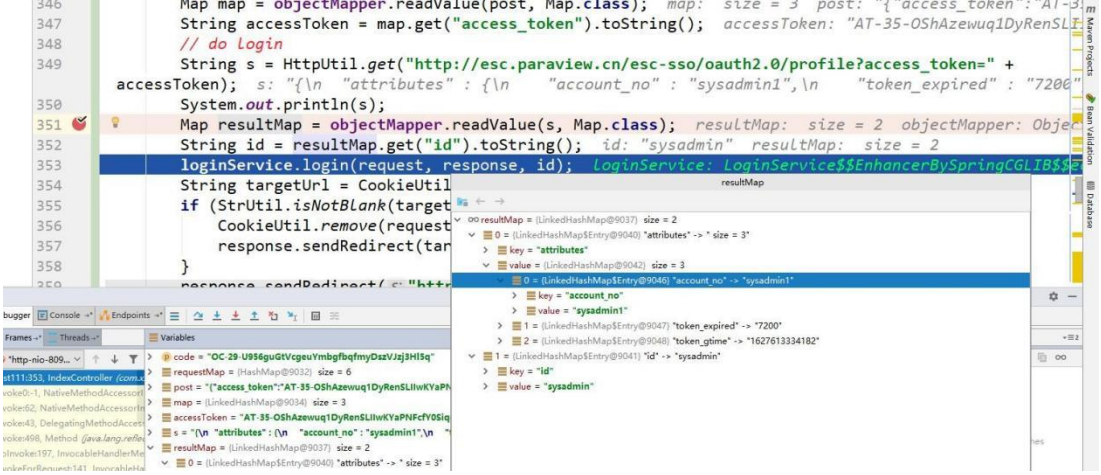
统一身份认证系统会自动回调配置的回调接口并带上 code

通过 code 去统一身份认证系统换取 token

```java
    @RequestMapping("/test11")
    @PermissionLimit(limit = false)
    public void test111(HttpServletRequest request, HttpServletResponse response, ModelAndView
modelAndView, String code) throws IOException {
        System.out.println(code);
        Map<String, Object> requestMap = new HashMap<>();
        requestMap.put("grant_type", "authorization_code");
        requestMap.put("oauth_timestamp", System.currentTimeMillis());
        requestMap.put("client_id", "d72e51b8c9c74178");
        requestMap.put("client_secret", "0db43ab5f150486390a233149707080c");
        requestMap.put("code", code);
        requestMap.put("redirect_uri", URLEncoder.createDefault().encode( path: "http://192.168.239
.1:8090/xxl-job-admin/test11", StandardCharsets.UTF_8));
        String post = HttpUtil.post( urlString: "http://esc.paraview.cn/esc-sso/oauth2.0/accessToken",
requestMap);
        System.out.println(post);
        Map map = objectMapper.readValue(post, Map.class);
        String accessToken = map.get("access_token").toString();
```

通过 token 去统一身份认证系统换取登录的用户 取 account_no




Map map = objectMapper.readValue(post, Map.class);
String accessToken = map.get("access_token").toString();
// do login
String s = HttpUtil.get("http://esc.paraview.cn/esc-sso/oauth2.0/profile?access_token=" +
accessToken);
System.out.println(s);
Map resultMap = objectMapper.readValue(s, Map.class);
String id = resultMap.get("id").toString();
loginService.login(request, response, id);
String targetUrl = CookieUtil
if (StrUtil.isNotBlank(target
CookieUtil.remove(request
response.sendRedirect(tar
}
resultMap
resultMap = (LinkedHashMap@9037) size = 2
0 = (LinkedHashMap$Entry@9040) "attributes" -> " size = 3"
key = "attributes"
value = (LinkedHashMap@9042) size = 3
0 = (LinkedHashMap$Entry@9046) "account_no" -> "sysadmin1"
key = "account_no"
value = "sysadmin1"
1 = (LinkedHashMap$Entry@9047) "token_expired" -> "7200"
2 = (LinkedHashMap$Entry@9048) "token_gtime" -> "1627613334182"
1 = (LinkedHashMap$Entry@9041) "id" -> "sysadmin"
key = "id"
value = "sysadmin"
Console
Endpoints
Frames
Threads
Variables
code = "OC-29-U956guGtVcgeuYmbgfbqfmyDszVJzj3HI5q"
requestMap = (HashMap@9032) size = 6
post = "{"access_token":"AT-35-OShAzewuq1DyRenSLIlwKYaPN
map = (LinkedHashMap@9034) size = 3
accessToken = "AT-35-OShAzewuq1DyRenSLIlwKYaPNFcfY0Siq
s = "{\n "attributes" : {\n "account_no" : "sysadmin1",\n
resultMap = (LinkedHashMap@9037) size = 2
0 = (LinkedHashMap$Entry@9040) "attributes" -> " size = 3"
Maven Projects
Bean Validation
Database