统一认证接入-接口自定义开发手册

一、自定义开发手册

1.1 概述

1.2 自定义接入说明

1.2.1 标准接口示例

1.2.2 方法参数说明

1.2.3 自定义接口实现类

1.2.4 认证接口代码（无需关注）

1.2.5 自定义配置

1.2.6 自定义接入认证问题列表

1.3 提示页面信息

文档版本控制				
序号	日期	服务版本	修改内容	修改人
1	20221103		初稿	杨腾飞
2	20221114		接口方法参数调整	杨腾飞
3	20221216	weaver-intunifyauth-client（1.1.20）	支持实现类自定义配置	杨腾飞
4	20230420	weaver-intunifyauth-client（1.3.5）	支持用户登录语言	杨腾飞
5	20230817	weaver-intunifyauth-client（1.7.0）	支持根据当前域跳转回调地址	杨腾飞

一、自定义开发手册

1.1 概述

自定义实现类接入第三方认证，类似二开的形式，目前E10自定义接入提供一下能力：

1、私有化单租户场景默认开启全资源拦截，系统级控制

2、私有化多租户场景支持团队级控制和系统级控制，支持全资源拦截动态配置（仅主团队）

3、支持页面动态配置自定义实现类

1.2 自定义接入说明

1、需要在微服务weaver-intunifyauth-client-service工程下，创建自定义接口实现类，需要在包com.weaver.intunifyauth.client.custom.service.impl目录下创建，该类必须实现并重写接口com.weaver.intunifyauth.client.custom.service.AuthclientCustomService的所有方法

1.2.1 标准接口示例

接口com.weaver.intunifyauth.client.custom.service.AuthclientCustomService方法如下，自行实现即可

Java 复制代码

```java
public interface AuthclientCustomService {

    /**
     *统一认证接入-登录接口自定义实现
     * @param request 客户端浏览器请求
     * @return Map<String,String> 返回认证登录地址跳转
     */
    @ApiOperation(value = "统一认证接入-登录接口自定义实现",response = String.class)
    @ApiImplicitParam(name = "request",value = "客户端浏览器请求",required = true)
    Map<String,String> login(HttpServletRequest request, HttpServletResponse response);

    /**
     *统一认证接入-用户校验接口自定义实现
     * @param request 客户端浏览器请求
     * @return  Map<String,String> 返回map集合
     */
    @ApiOperation(value = "统一认证接入-用户校验接口自定义实现",response = Map.class)
    @ApiImplicitParam(name = "request",value = "客户端浏览器请求",required = true)
    Map<String,String> getUserInfo(HttpServletRequest request, HttpServletResponse response);

    /**
     * 统一认证接入-登出注销接口自定义实现
     * @param request 客户端浏览器请求
     * @return Map<String,String> 返回map集合 重定向url地址
     */
    @ApiOperation(value = "统一认证接入-登出注销接口自定义实现",response = Map.class)
    @ApiImplicitParam(name = "request",value = "客户端浏览器请求",required = true)
    Map<String,String> logout(HttpServletRequest request, HttpServletResponse response);

    /**
     * 统一认证接入-控制是否跳登录接口
     * @return String 重定向url地址
     */
    @ApiOperation(value = "统一认证接入-控制是否跳登录接口",response = Boolean.class)
```

```
35        Boolean isRedirectLogin();
36    }
```

1.2.2 方法参数说明

描述1：自定义登录login接口

方法名：Map<String,String> login(HttpServletRequest request，HttpServletResponse response);

入参：

参数名	必填	类型	描述
request	是	HttpServletRequest	客户端浏览器请求
response	是	HttpServletResponse	服务端响应

返回值Map

参数名	必填	类型	描述	示例
redirectUrl	是	String	重定向地址	map.put("redirectUrl","https://www.eteams.cn")
code	否	String	异常错误码【900003 自定义登录异常，请检查】	map.put("code","900003")

描述2：获取用户信息接口

方法名：Map<String,String> getUserInfo(HttpServletRequest request，HttpServletResponse response);

入参：

参数名	必填	类型	描述

request	是	HttpServletRequest	客户端浏览器请求
response	是	HttpServletResponse	服务端响应

返回值Map

参数名	必填	类型	描述	示例
accountType	是	String	账号映射规则序号 账号规则对应关系如下： 4：工号 5：电子邮箱 6：手机号码	map.put("accountType","6");
username	是	String	登录账号	map.put("username",13323708622);
code	否	String	异常错误码【900006 用户信息接口异常，请检查】	map.put("code","900006")
langtype	否	String	用户登录语言参数 7 zh_CN 简体中文 9 zh_TW 繁體中文 8 en_US 英语 18 es_ES ...	map.put("langtype","15")

描述3：自定义登出注销

方法名：Map<String,String> logout(HttpServletRequest request，HttpServletResponse response);

入参：

参数名	必填	类型	描述
request	是	HttpServletRequest	客户端浏览器请求
response	是	HttpServletResponse	服务端响应

返回值Map

参数名	必填	类型	描述	示例
logoutUrl	是	String	登出重定向地址	map.put("logoutUrl","https://www.eteams.cn")
isLogout	是	String	是否跳转登出URL 0：不登出 1：登出	map.put("isLogout","1")
code	否	String	异常错误码 【900009 自定义登出异常，请检查】	map.put("code","900009")

描述4：是否跳转第三方认证登录页

方法名：Boolean isRedirectLogin();

返回值Boolean

返回值类型	描述	备注说明

Boolean	接口返回Boolean; true： 重定向跳转 false：不跳转第三方登录页，直接调用getUserInfo接口；	（此种情况适用于请求资源路径带有ticket或授权码参数，已经获取到用户接口所必须参数，这种情况无需跳转登录接口验证，会直接跳转到获取用户信息接口，需要客户端重写接口实现）

注：以上接口中的四个方法必须全部实现，根据接口说明完成客户端自身业务逻辑
针对自定义login接口的实现，返回值url拼接的回调地址需跳转*/papi/iaauthclient/bs/custom/getUserInfo*
示例：比如接入cas认证，认证登录回调地址需返回如下格式：
https://xxxx.com/login?service=https://xxxx/papi/iaauthclient/bs/custom/getUserInfo

1.2.3 自定义接口实现类

注：以下实现类仅是示例代码，这块方法业务逻辑需自行实现

Java 复制代码

```java
public class AuthclientMobileServiceImpl implements AuthclientCustomService {

    private static final Logger logger = LoggerFactory.getLogger(AuthclientMobileServiceImpl.class);

    @Override
    public Map<String, String> login(HttpServletRequest request, HttpServletResponse response) {
        Map<String,String> maps = new HashMap<>();
        // 获取当前请求域
        String redirectUri = CustomCommonUtil.getRedirectLogin(request);
        logger.info("======redirectUri======"+ redirectUri);
        try {
            maps.put("redirectUrl","http://10.10.10.42/oauth/login?appid=61d08a22-d6ec-4f36-965c-5bb0721a9360&service="+URLEncoder.encode(redirectUri+WeaMultipathConfigUtil.getBackUrl()+"/papi/iaauthclient/bs/custom/getUserInfo","UTF-8"));
        } catch (UnsupportedEncodingException e) {
        }
        return maps;
    }
    @Override
    public Map<String, String> getUserInfo(HttpServletRequest request, HttpServletResponse response) {
        logger.info("-----------支持移动端demo-----------");
        String ticket = request.getParameter("ticket");
        logger.info("-----------ticket-----------"+ticket);
        Map<String, String> retMap = new HashMap<>();

        // 没有ticket直接返回
        if (StrUtil.isEmpty(ticket)) {
            retMap.put("redirectUrl", "");
            retMap.put("code", "10004");
            return retMap;
        }
        // 获取用户信息
        String redirectUri = CustomCommonUtil.getRedirectUserInfo(request);
        logger.info("======redirectUri======"+redirectUri);
        HashMap<String, String> userInfo = getFSUserInfo(redirectUri+ WeaMultipathConfigUtil.getBackUrl()+"/papi/iaauthclient/bs/custom/getUserInfo", ticket);
        if (userInfo.containsKey("username")) {
```

```
35              retMap.put("username", userInfo.get("username"));
36              retMap.put("accountType", "6");
37              retMap.put("code", "");
38          } else {
39              retMap.put("username", "sorry");
40              retMap.put("accountType", "6");
41              retMap.put("code", "400");
42          }
43          try {
44              retMap.put("logouturl", "http://10.10.10.42/oauth/logout?service="+URLEncoder.encode(redirectUri+ WeaMultipathConfigUtil.getBackUrl()+"/papi/iaauthclient/bs/custom/login?tk=6e061bbb052f039e82544935c687fe93","utf-8"));
45          } catch (UnsupportedEncodingException e) {
46          }
47          return retMap;
48      }
49
50      @Override
51      public Map<String, String> logout(HttpServletRequest request, HttpServletResponse response) {
52          logger.info("-----------退出处理----------");
53          // 返回信息
54          String redirectUri = CustomCommonUtil.getRedirectLogout(request);
55          Map<String, String> returnMap = new HashMap<>();
56          returnMap.put("isLogout", "1");
57          try {
58              returnMap.put("logoutUrl", "http://10.10.10.42/oauth/logout?service="+URLEncoder.encode(redirectUri+ WeaMultipathConfigUtil.getBackUrl()+"/papi/iaauthclient/bs/custom/login?tk=6e061bbb052f039e82544935c687fe93","utf-8"));
59              SecurityCasUtils.writeCookie(response, "oaaddress", URLEncoder.encode(redirectUri, "UTF-8"));
60          } catch (UnsupportedEncodingException e) {
61          }
62  //        returnMap.put("logoutUrl", oaUrl);
63          return returnMap;
64      }
65
66  }
```

1.2.4 认证接口代码（无需关注）

controller作为统一认证业务控制层，负责控制业务层接口流转，**开发人员无需关注，知识帮助开发人员梳理整个认证逻辑，只需要注重实现类逻辑即可**；

> 自定义登录接口：控制认证跳转；异常处理；

> 获取用户信息接口：获取用户信息；完成登录认证；创建会EteamsID会话；异常处理；
> 登出注销接口：负责登出注销完成之后页面重定向；注销EteamsID会话；异常处理；

统一认证自定义接口业务控制层 Java 复制代码

```java

**
 * @author yangtengfei
 * @since 20221026
 */
@RestController
@RequestMapping("papi/iaauthclient/bs/custom")
@Api(tags = "统一认证自定义接口业务逻辑层")
public class IaAuthCustomController {
    Logger logger =  LoggerFactory.getLogger(IaAuthCustomController.class);
    @Autowired
    private WeaverCasProperties weaverCasProperties;
    @Autowired
    private WeaverCasCommonService weaverCasCommonService;
    @Autowired
    private IaAuthclientCustomService iaAuthclientCustomService;
    @Autowired
    private IaAuthclientService authclientService;
    @Autowired
    @Qualifier("intunifyauthclient_UserClientService")
    private UserClientService userClientService;
    @RpcReference
    private RemoteBasicCommonService remoteBasicCommonService;
    @RpcReference
    private RemoteCusAppMenuService cusAppMenuService;
    @Autowired
    private WeaverCasUserInfoService weaverCasUserInfoService;
    @Value("${oa_domain}")
    public String oaDomain;
    @Value("${oa_domain_address}")
    public String oaDomainAddress;
    /**
     * 统一认证接入-登录接口自定义实现
     * @param servletRequest 服务器请求
     * @param servletResponse 服务器响应
     */
    @RequestMapping(value = "login")
    @ApiOperation("统一认证接入-登录接口自定义实现")
    @WeaPermission(publicPermission = true)
    public void  customLogin(ServletRequest servletRequest, ServletResponse servletResponse){
        HttpServletResponse response_ = (HttpServletResponse) servletResponse;
```

```
        HttpServletRequest request_ = (HttpServletRequest) servletReques
t;
        // 映射规则类转换
        try {
            String oaAddress =  request_.getParameter("oaAddress");
            String redirect_uri =  request_.getParameter("redirect_uri");
            String requestUrl = URLDecoder.decode(request_.getRequestURL
().toString()+"?"+request_.getQueryString(),"UTF-8");
            String tk = CustomCommonUtil.getFieldValue(requestUrl,"tk");
            if (null!=tk && !tk.equals("")){
                tk = StandardSymmeric.decode(tk);
            }
            Map map = userClientService.getAuthInfo(tk);
            String redirectUri = CustomCommonUtil.getRedirectUriPath(weav
erCasProperties,request_,response_,oaAddress,redirect_uri,map);
            AuthCustomConfigBean authCustomConfigBean = iaAuthclientCusto
mService.getAuthCustomIsOpen(tk);
            SecurityCasUtils.writeCookie(response_,"redirect_uri",redirec
tUri+"/papi/iaauthclient/bs/custom/login?tk="+CustomCommonUtil.getFieldVa
lue(requestUrl,"tk"));
            // 添加oa原始地址参数到cookie中
            SecurityCasUtils.writeCookie(response_,"tk",CustomCommonUtil.
getFieldValue(requestUrl,"tk"));
            // 添加校验
            String path = URLEncoder.encode(redirectUri+WeaMultipathConfi
gUtil.getFrontUrl()+"?intunifyauth_client=freepass","UTF-8");
            // 异常信息校验
            CustomCommonUtil.validateConfig(map,request_,response_,path);
            if(map.get("auth_type")!=null && !CustomCommonUtil.getCustomA
uthType().contains(map.get("auth_type").toString().toLowerCase())){
                ErrorPageUtil.redirectPages("80028",path,response_);
            }
            AuthclientCustomService authclientCustomService = CustomCommo
nUtil.getCustomService(map,authCustomConfigBean,iaAuthclientCustomServic
e,authclientService);
            if (authclientCustomService == null){
                CustomCommonUtil.redirectPagesLogin("70007",path,request
_,response_);
                return;
            }
            // 校验是否跳转登录接口，否则直接重定向用户接口
            if(!authclientCustomService.isRedirectLogin()){
                getUserInfo(servletRequest,servletResponse);
                return;
            }
            logger.info("=========================================");
            // 调用实现类login接口获取重定向地址
```

```
            Map<String, String> loginMap = authclientCustomService.login(request_,response_);
            // 重定向地址空校验
            CustomCommonUtil.validateLogin(loginMap,request_,response_,path);
            logger.info("======login redirectUrl======"+loginMap.get("redirectUrl"));
            response_.sendRedirect(loginMap.get("redirectUrl"));
        } catch (IOException e) {
            logger.info("",e);
        }

    }


    /**
     *  统一认证接入-用户信息接口自定义实现
     * @param servletRequest 服务器请求
     * @param servletResponse 服务器响应
     */
    @RequestMapping(value = "getUserInfo",method = RequestMethod.GET)
    @ApiOperation("统一认证接入-用户信息接口自定义实现")
    @WeaPermission(publicPermission = true)
    public void  getUserInfo(ServletRequest servletRequest, ServletResponse servletResponse) throws IOException {
        HttpServletResponse response_ = (HttpServletResponse) servletResponse;
        HttpServletRequest request_ = (HttpServletRequest) servletRequest;
        // 第三方不跳转登录页，可以指请求中获取参数oaAddress
        String oaAddress  = request_.getParameter("oaAddress");
        String redirect_uri  = request_.getParameter("redirect_uri");
        String requestUrl = URLDecoder.decode(request_.getRequestURL().toString()+"?"+request_.getQueryString(),"UTF-8");
        // 测试认证接口跳转，打印输出信息
        String tk  = CustomCommonUtil.getTenantKey(request_,requestUrl);
        Map map = userClientService.getAuthInfo(tk);
        String domain = CommonUtils.getDomainBymap(weaverCasProperties,oaAddress,redirect_uri,request_,map);
        AuthCustomConfigBean authCustomConfigBean = iaAuthclientCustomService.getAuthCustomIsOpen(tk);
        String frontdomain = domain+WeaMultipathConfigUtil.getFrontUrl();
        String backdomain = domain+WeaMultipathConfigUtil.getBackUrl();
        String path = URLEncoder.encode(frontdomain+"?intunifyauth_client=freepass","UTF-8");
        AuthclientCustomService authclientCustomService = CustomCommonUtil.getCustomService(map,authCustomConfigBean,iaAuthclientCustomService,aut
```

```
hclientService);
113             // 映射规则类转换
114             Map<String,String> userInfoMap = authclientCustomService.getUserI
nfo(request_,response_);
115             // TODO 网关认证-code异常提示页面
116             // 如果请求获取不到oa原始地址，从cookie获取
117             if(null== oaAddress || "".equals(oaAddress)){
118                 oaAddress = SecurityCasUtils.getCookie(request_, "oaaddres
s");
119                 oaAddress = (null==oaAddress || "".equals(oaAddress)) ? oaAdd
ress: URLDecoder.decode(oaAddress,"UTF-8");
120             }
121             if(null!=userInfoMap.get("status") && "400".equals(userInfoMap.ge
t("status"))){
122                 ErrorPageUtil.redirectPage(response_,request_,path, userInfoM
ap, frontdomain);
123             }
124             // 校验实现类用户信息
125             CustomCommonUtil.validateUserInfo(oaDomainAddress,userInfoMap,req
uest_,response_,path);
126             /*创建json对象存储请求参数*/
127             JSONObject jsonParam = CustomCommonUtil.getJsonParam(userInfoMap,
request_,tk);
128             logger.info("请求参数:"+jsonParam.toString());
129             JSONObject jsonResult = null;
130             try {
131                 jsonResult = CommenJsonUtil.getJsonResult(backdomain, jsonPar
am.toString());
132             } catch (Exception e) {
133             }
134             // 登录接口返回值异常校验
135             CustomCommonUtil.validateJsonResult(oaDomainAddress,response_,jso
nResult,path,userInfoMap);
136             JSONObject jsonData = jsonResult.getJSONObject("data");
137             // 添加CsrfToken到cookie,安全模块要求
138             StringUtil.addCsrfCookie(response_, StringUtil.createCsrfToken
();
139             logger.info("WeaverCasUserInfoService getEmployeeIdByEteamsId:
{}"+ weaverCasUserInfoService.getEmployeeIdByEteamsId(jsonData.getString
("eteamsId")));
140             // 屏蔽菜单【申请创建加入团队】
141             cusAppMenuService.configMenuShowType(Long.parseLong("511588266444
1018158"), 0,tk, weaverCasUserInfoService.getEmployeeIdByEteamsId(jsonDat
a.getString("eteamsId")));
142             String gopage = request_.getParameter("gopage");
143             logger.info("gopage重定向URL: " + gopage);
144             boolean isFalse1 = false;
```

```
145             // gopage跳转
146             if (gopage != null && !gopage.equals("") && null==userInfoMap.get
    ("is_xf")) {
147                 // 构造客户端会话信息eteamsid
148                 SecurityCasMethodUtils.writeLogout(response_,URLEncoder.encod
    e(backdomain+"/papi/iaauthclient/bs/custom/logout","UTF-8"));
149                 SecurityCasUtils.writeEteamsId(response_,jsonData.getString
```

1.2.5 自定义配置

自定义配置入口：登录E10【后台管理系统】-【集成中心】-【统一认证接入管理】认证方式选择自定义；

自定义开发手册，点击配置页下方小提示【接口自定义开发文档】



1.2.6 自定义接入认证问题列表

1.如何获取原始请求地址

// 添加oa资源地址到cookie中，比如请求地址http://www.baidu.com/xxxx/xxxx

Cookie oaCookie = new Cookie("oaaddress","http://www.baidu.com/xxxx/xxxx");

response_.addCookie(oaCookie);

2.如何获取租户代码?

//租户代码保存在cookie中

Cookie tkCookie = new Cookie("tk",StandardSymmeric.encode(tk));

response_.addCookie(tkCookie);

// 租户值加解密方法，实现类需要cookie自取解密即可

加密：StandardSymmeric.encode(tk)

解密：StandardSymmeric.decode(tk)

3.访问OA跳转认证失败，无法登录如何处理?

注意！！！全资源拦截一旦开启，认证过程中出现异常，跳转提示页面碰到登不出去的情况，临时两种方式解决：

方式1：可以通过执行papi/iaauthclient/bs/custom/config/delCache

方式2：访问当前域名加上?intunifyauth_client=freepass

4.login方法如何根据请求地址获取当前域?

CustomCommonUtil.getRedirectLogin(request);

5.getUserInfo方法如何根据请求地址获取当前域?

CustomCommonUtil.getRedirectUserInfo(request);

6.logout方法如何根据请求地址获取当前域?

CustomCommonUtil.getRedirectLogout(request);

7.自定义实现类开发完成文件放在哪个服务下的包?

需要放在weaver-intunifyauth-client服务下包com.weaver.intunifyauth.client.custom.service.impl路径下，然后重启服务

8.私有化单租户如何开启全资源拦截?

私有化单租户默认开启全拦截配置，页面配置不显示

9.私有化多租户如何开启全资源拦截?

私有化多租户只有主管理租户才可以显示配置全资源拦截开关

10.weaver-intunifyauth-client 1.7.0版本之前支持多地址吗?

不支持，1.7.0（包含）之后支持，但是前提条件认证服务也支持多地址配置

1.7.0版本之前支持系统默认地址跳转

11.如何查询系统默认地址?

nacos平台搜索*weaver-common-cas-core*配置文件，此文件配置项地址就是系统默认地址

1.3 提示页面信息

自定义接口接入认证过程中，如果自定义接口返回值异常或其他异常情况，将会跳转到提示页面。

以下列举可能出现的提示：

code	提示信息	描述
500	请求服务器失败，请联系管理员	系统错误

10001	登陆失败，请联系管理员	passport登录失败
10003	账号信息不存在，请联系管理员	user账号信息不存在
10004	人员信息不存在，请联系管理员	employee人员信息不存在
70001	请求参数有误，请联系管理员	集成登录请求参数缺失[tenantKey]
70002	请求参数有误，请联系管理员	集成登录请求参数缺失[type]
70003	请求参数有误，请联系管理员	集成登录请求参数缺失[value]
70004	集成登录请求code校验失败，请联系管理员	集成登录请求code校验失败
80038	鉴权参数不完整，请求不合法	鉴权参数不完整，请求不合法
80039	鉴权失败，鉴权信息有误	鉴权失败，鉴权信息有误
900001	自定义登录返回值【Map】为空	自定义登录返回值【Map】为空
900002	自定义登录返回参数【redirectUrl】为空	自定义登录返回参数【redirectUrl】为空
900003	自定义登录异常，请检查	自定义登录异常，请检查
900004	用户信息接口返回值【Map】为空	用户信息接口返回值【Map】为空
900005	用户信息接口返回参数【username】为空	用户信息接口返回参数【username】为空
900006	用户信息接口异常，请检查	用户信息接口异常，请检查
900007	自定义登出返回值【Map】为空	自定义登出返回值【Map】为空
900008	自定义登出返回参数【logoutUrl】为空	自定义登出返回参数【logoutUrl】为空
900009	自定义登出异常，请检查	自定义登出异常，请检查