美的集团企业标准

QM－GC08.004－2014

修订

美的集团应用接入规范

2014-06-29 发布　　2014-06-29 实施

美的集团发布

修　订　页

编制/修订原因说明：

为加强集团应用接入美的 LDAP 系统或 4A 单点登录平台的管理，规范应用接入流程，提高应用接入效率，统一帐号管理和认证，特制定本规范。

原章节号	现章节号	修订（内容）说明	修订人/时间
附录二	附录二	用户帐号认证代码修订	蒋明/2016-06-04

运营维护中心审核： 吴自良	其他模块会签人：
各单位会签人： 无	审批人： 谷云松

注：

1、 流程清单内所有文件审批发布必须按照清单规定的审批流程组织审批，并在此处写明审批流程。

编制（流程与 IT 部运营维护中心）→审核（运营维护中心负责人）→会签（流程与 IT 部综合管理中心）→格式审核（战略经营部经营管理）→审批（流程与 IT 部总监）→归档（战略经营部）

2、 流程清单外的文件审批流程由美的集团战略经营部确定。

美的集团企业标准

美的集团应用接入规范

QM-GC08.004-2014

1 总则

1.1 为加强集团应用接入美的 LDAP 系统或 4A 单点登录平台的管理，规范应用接入流程，提高应用接入效率，统一帐号管理和认证，特制定本规范。

1.2 本规范是集团信息化管理制度的重要组成部分，在全集团范围内适用。

2 术语

2.1 美的 4A 安全身份管理平台：简称 4A 平台，指实现统一帐号（ACCOUNT）、统一认证（AUTHENTICATION）、统一授权（AUTHORIZATION）和统一审计（AUDIT）的信息系统集合。

2.2 LDAP：英文全称是 Lightweight Directory Access Protocol，即轻量级目录访问协议，主要用于帐号信息存储和帐号认证，具体在文档 RFC2251~RFC2256 和 RFC2829~RFC2831 中有详细说明。

2.3 OAM：英文全称是 Oracle Access Manager，是 Oracle 统一访问控制平台，是实现统一单点登录的主要组件。

2.4 OIM：英文全称是 Oracle Identity Manager，是 Oracle 统一帐号管理平台，是实现帐号推送和同步的主要组件。

2.5 下游应用系统：指需要接入美的 LDAP 或 4A 单点登录平台的应用系统，如 MIP、Mail、E-HR 系统等。

2.6 统一认证管理：是指通过 OAM，实现各个应用系统之间的单点登录访问。

2.7 统一帐号管理：是指由下游应用通过标准的 LDAP 协议同步用户帐号或通过 OIM 向下游应用推送用户帐号，实现多套系统使用同一帐号。

3 参考文献

3.1 RFC2251~RFC2256 和 RFC2829~RFC2831

3.2 美的信息用户统一身份管理平台 LDAP Schema 设计

4 服务目录

4.1 用户帐号供给：通过标准的 LDAP 协议或 OIM 向下游应用系统提供用户帐号信息。

4.2 组织架构供给：通过标准的 LDAP 协议向下游应用系统提供组织架构信息。

4.3 帐号统一认证：向下游应用系统提供用户统一帐号认证服务，统一控制密码强度和密码策略，并实现应用系统间的单点登录。

4.4 帐号分类管理：对不同特性的用户的帐号进行分类管理。

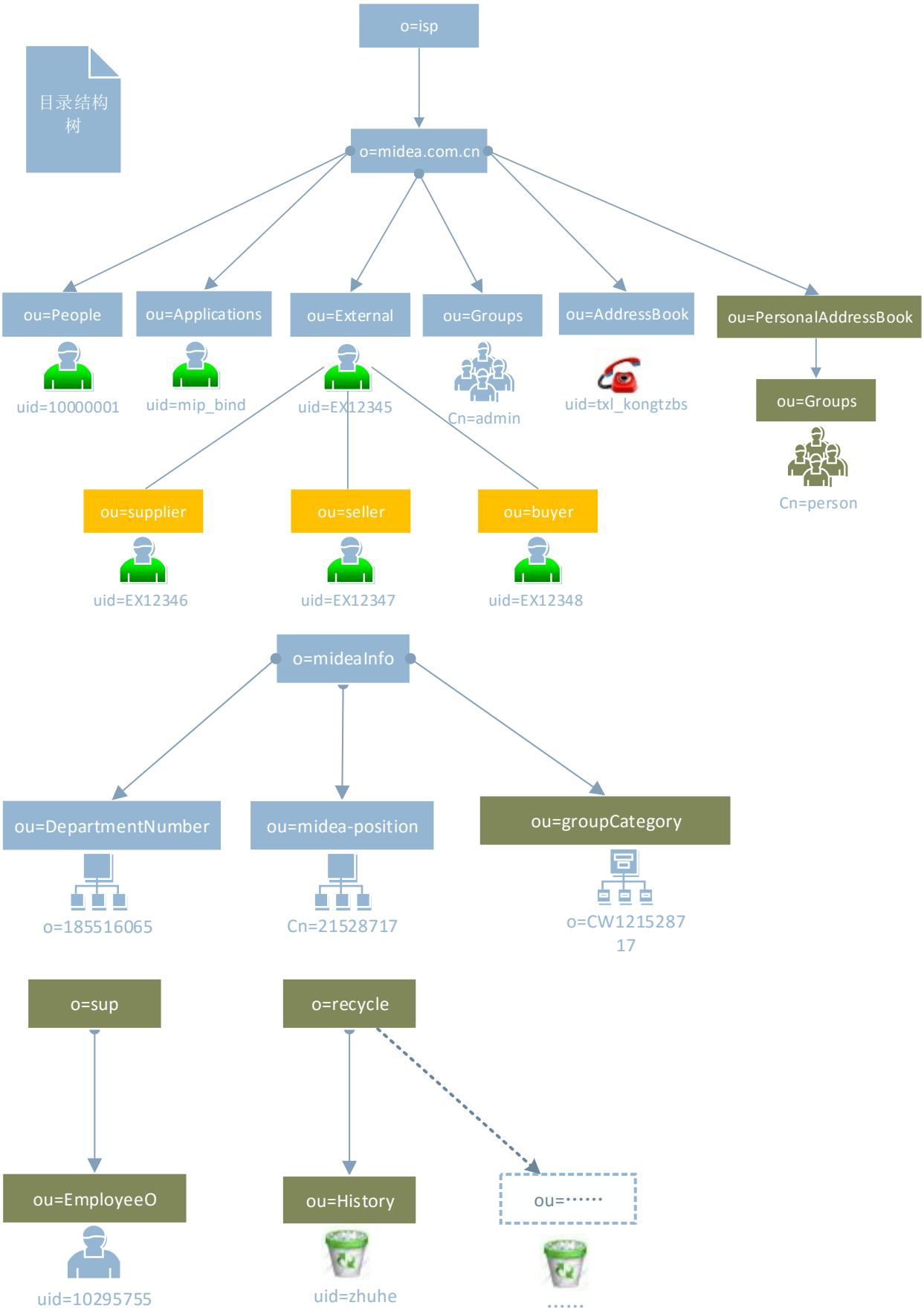
4.5 帐号权限控制：通过静态组或动态组的方式控制用户帐号在下游应用系统的 IT 权限。

4.6 用户行为审计：审计用户帐号的登录和访问行为信息。

5 统一帐号管理接入规范

5.1 帐号管理系统架构

5.1.1 LDAP 目录树架构

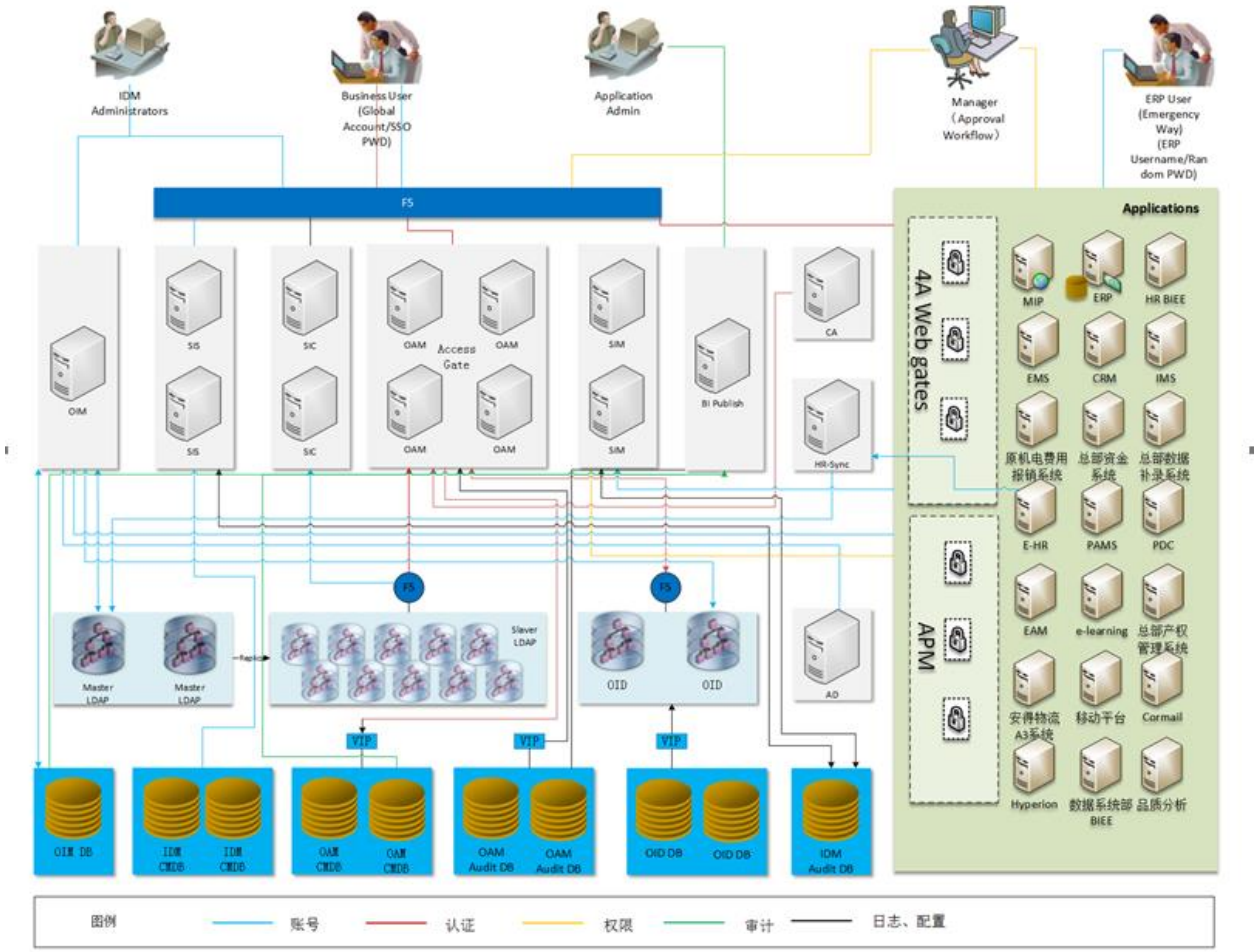


美的 LDAP 共有四个根节点：o=isp、o=sup、o=recycle 和 o=mideainfo，并按美的业务需要继续往下细分。

现在，共分成以下 11 个大类，详细的用户、部门、职位的属性定义请参见“附录一”。

节点	保存的内容
ou=People,o=midea.com.cn,o=isp	内部帐号、职能帐号和系统帐号
ou=Applications,o=midea.com.cn,o=isp	应用集成帐号
ou=External,o=midea.com.cn,o=isp	外部帐号
ou=Groups,o=midea.com.cn,o=isp	静态组、动态组
ou=AddressBook,o=midea.com.cn,o=isp	公共通讯录
ou=PersonalAddressBook,o=midea.com.cn,o=isp	个人通讯录
ou=DepartmentNumber,o=mideainfo	部门信息
ou=Midea-position,o=mideainfo	岗位信息
ou=GroupCategory,o=mideainfo	群组分类
ou=People,o=History,o=recycle	已离职删除的内部帐号
ou=Employee0,o=sup	0 类员工信息

5.1.2 帐号管理总体架构



目前正式环境的 LDAP 版本是 Oracle Directory Server Enterprise Edition 11g（11.1.1.5.1），并组成 2 主 10 从、互相复制的高可用系统架构。

5.2 LDAP API 使用介绍

LDAP 作为国际标准的协议，各种程序语言都有非常成熟的 API 可以使用。一个应用程序调用 LDAP API 一般有以下四步，详细的调用方法请参见“附录二”的样例代码。

5.2.1 建立与 LDAP Server 的连接，如使用函数调用 ldap_open()。

5.2.2 核对身份 LDAP Server 和（或）X.500 DSA，如使用 ldap_bind()。

5.2.3 执行 LDAP 操作并获得返回值，如 Search()。

5.2.4 断开连接。

5.3 下游应用接入美的 LDAP 实施规范

5.3.1 内部帐号信息获取规范

内部帐号是指内部员工的帐号。这部分帐号的信息，如姓名、员工号、职位、职级、证件类型、证件号、国籍等等，都同步于 E-HR 系统。这部分用户的信息存储在“ou=People,o=midea.com.cn,o=isp”节点下，过滤条件：(&(objectClass=midea-person)(midea-userType=0))。

5.3.2 内部临时帐号信息获取规范

内部临时帐号也是美的内部员工的帐号，但这部份员工由于刚入职或收购合并等原因暂时未录入中国 E-HR 系统，未能获取到工号。这部分用户的信息存储在“ou=People,o=midea.com.cn,o=isp”节点下，过滤条件：(&(objectClass=midea-person)(midea-userType=1))。

5.3.3 供应商帐号信息获取规范

供应商帐号信息存储在“ou=External,o=midea.com.cn,o=isp”节点下，过滤条件：(&(objectClass=midea-person)(midea-userType=5))。

5.3.4 组织机构信息获取规范

组织机构信息存储在“ou=departmentnumber,o=mideainfo”节点下，过滤条件：(objectClass=organization)。

5.3.5 职位信息获取规范

职位信息存储在“ou=midea-position,o=mideainfo”节点下，系统通过用户对象上的职位编码和这个节点下的条目关联。

5.3.6 接入服务总线规范

接入服务只提供认证服务和数据同步，统一信息用户管理系统使用目录服务器来提供静态口令认证服务，仍然使用 LDAP 协议来认证服务的接入。

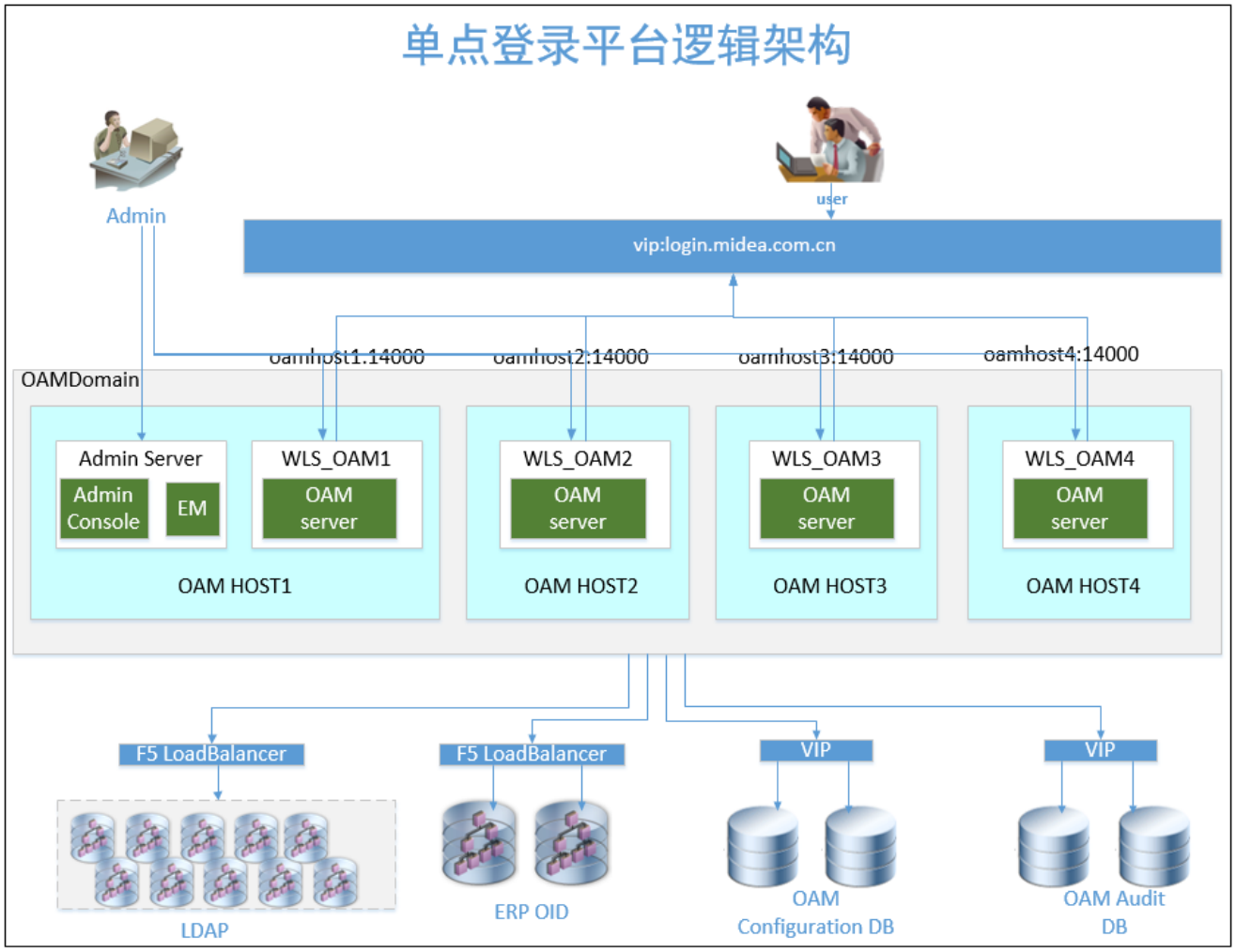
5.3.7 接入授权服务规范

应用系统如需接入美的 LDAP 进行口令认证或获取信息，应用系统管理员要在 MIP 上提交《应用接入 LDAP 申请表》，具体流程位置和流程样例参见“附录三”。经部门总监审批通过后，由美的 LDAP 管理员创建接入绑定帐号，并分配合适的权限。应用系统管理员通过绑定帐号，使用标准 LDAP 协议接入美的 LDAP，实现静态口令。注意：应用帐号“谁申请，谁负责”，申请人必须妥善保管好应用帐号的密码，不得向任何人透露，必要时在配置文件中使用加密的密码，以防发生信息安全事故。

6 统一认证管理接入规范

6.1 认证管理系统架构

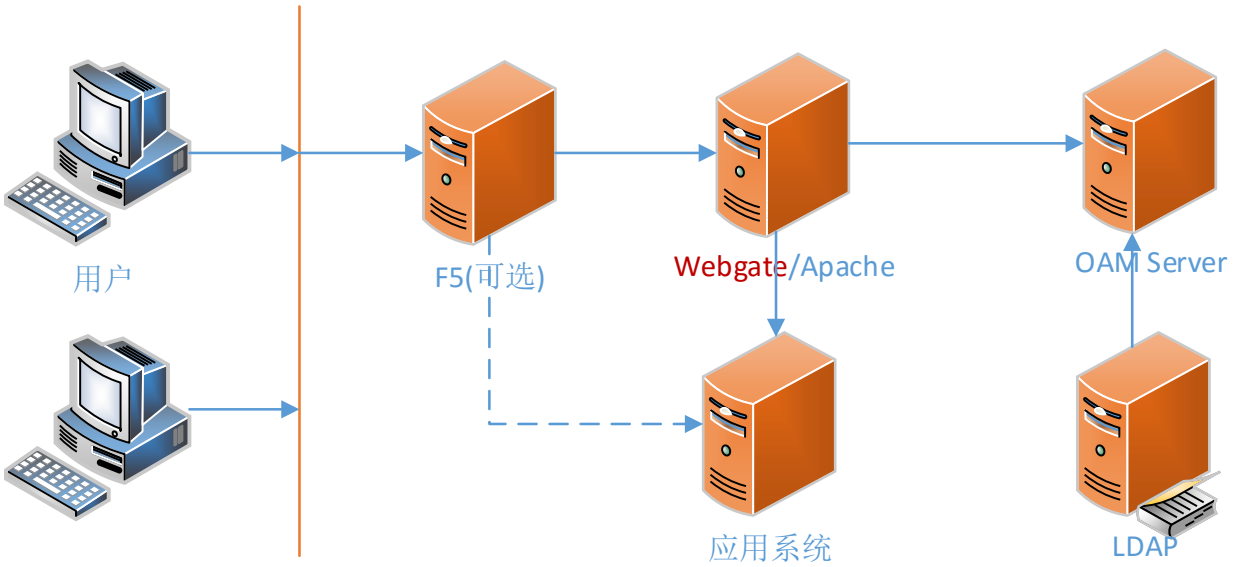
6.1.1 认证管理总体架构



美的单点登录平台采用业界标准的 Oracle 产品 Oracle Access Manager 进行构建，系统架构采用了 4 台高性能服务器的集群架构，前端使用 F5 作负载均衡，4 台服务器具备会话保持、Fail-Over 处理能力。测试结果证明，平台并发能力能够达到 2500 左右，远远超过目前业务的并发需求。

6.1.2 应用接入逻辑架构

6.1.2.1 Webgate 方式集成

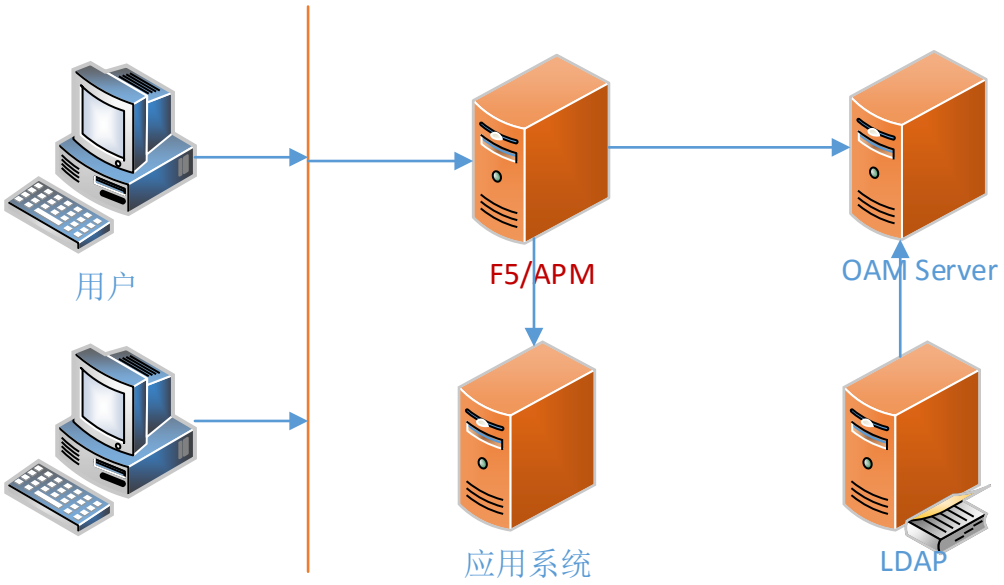


注意：这种集成方式需要在应用前端部署 Apache+Webgate，且通常是直接安装在应用服务器上面，但如果应用系统本身就已经部署了 IIS、IHS 或 OHS 代理服务器，则不需要再安装 Apache。单点登录集成前，用户访问的请求先经过代理服务器，再到应用服务器，而在集成后，用户访问的请求流向会发生改变。

具体说明如下：

步骤	集成前	集成后
1	用户访问应用系统资源	用户访问应用系统资源
2	F5 将请求分发到应用系统，如果用户已经登录过该系统，则执行步骤 5，否则执行步骤 3	Webgate 检查用户是否已经登录过 4A 平台，如果已经登录过，执行步骤 5，否则执行步骤 3
3	显示应用系统登录界面	显示 4A 平台统一单点登录界面
4	用户输入帐号密码	用户输入帐号密码
5	成功访问应用系统资源	成功访问应用系统资源

6.1.2.2 APM 方式集成



类似，通过 F5 APM 模块集成到 4A 平台，实现单点登录后，用户访问的请求流向也会发生改变。

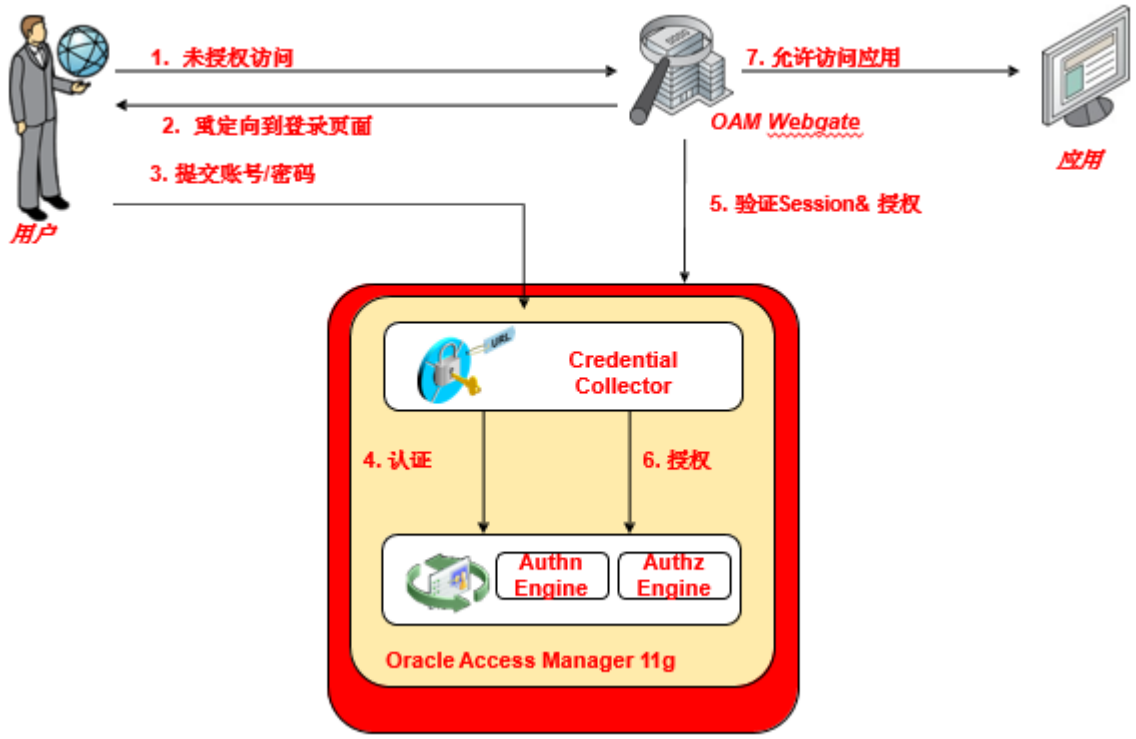
具体说明如下：

步骤	集成前	集成后
1	用户访问应用系统资源	用户访问应用系统资源
2	F5 将请求分发到应用系统，如果用户已经登录过该系统，则执行步骤 5，否则执行步骤 3	F5 APM 模块检查用户是否已经登录过 4A 平台，如果已经登录过，执行步骤 5，否则执行步骤 3
3	显示应用系统登录界面	显示 4A 平台统一单点登录界面
4	用户输入帐号密码	用户输入帐号密码
5	成功访问应用系统资源	成功访问应用系统资源

6.1.2.3 集成方式比较

集成方式	优点	缺点
Webgate	● 集成案例多 ● 参考资料多	● 需要在应用前端安装代理服务器和 Webgate ● Webgate 对某些操作系统版本支持不好
F5 APM	● 架构简单 ● 应用端无须安装额外的代理服务器或 Webgate ● 跨平台，对操作系统版本无依赖	● 集成案例少 ● 参考资料少

6.2 单点登录原理说明



过程说明：

6.2.1 用户访问被 4A 平台保护的应用系统资源。

6.2.2 OAM 拦截器，即 Webgate 或 F5 APM，检查该用户是否已经登录过 4A 平台；如果已经登陆过，则执行步骤 5，否则执行步骤 3。

6.2.3 用户在统一单点登录界面输入帐号密码。

6.2.4 OAM 服务端校验用户帐号密码，生成 session。

6.2.5 验证用户的 session 信息是否有效，验证用户是否有权限访问该应用系统资源。

6.2.6 OAM 服务端将验证结果返回给 OAM 拦截器，同时在客户端浏览器中生成了访问系统需要的 Cookie 和 Header 变量值。

6.2.7 成功访问应用系统资源。

6.3 应用接入条件

6.3.1 已经统一帐号管理，集成 LDAP 帐号认证和帐号同步。

6.3.2 定制类应用集成需要做少量的认证逻辑改造，应用方须具备一定的开发能力。

6.3.3 产品类应用集成一般能够通过产品配置完成，但是产品如果不支持 Oracle 单点登录配置，亦需做一定的开发才能完成集成。

6.3.4 能够提供独立的测试环境给 4A 项目组做集成验证。

6.4 应用接入流程

步骤	说明	负责人
接入申请	由应用方发现接入申请，提供本文档“附录四”要求的相关信息	应用管理员

准备测试环境	给 4A 项目组提供一套用于集成测试的应用环境，并授权 4A 项目组成员登录其服务器。由于配置过程中需要有可能会影响应用，该环境必须是独立的环境，以免影响用户测试	应用管理员
沟通集成方案	4A 项目组成员详细了解系统的部署情况，再和应用管理员介绍、讨论集成方案	4A 项目组 应用管理员
发送改造指引	4A 项目组发送相关集成改造的指引文档	4A 项目组
应用集成改造	应用方安排开发资源完成认证逻辑的改造，并部署到集成测试环境	应用管理员
安装拦截器	4A 项目组完成 OAM 服务器端和应用服务器端的安装和配置，并协调网络组同事修改 F5 资源池信息	4A 项目组
集成测试	应用方和 4A 项目组共同测试单点登录效果，关注应用的业务功能和单点登录的基本功能	4A 项目组 应用管理员

6.5 应用接入规范

6.5.1 定制开发应用接入规范

6.5.1.1 Header 方式接入

对于定制开发类应用，需要应用方做一定的定制开发，实现从 OAM 服务端获取 Header 变量 OAM_REMOTE_USER 的值作为用户登录名。代码示例如下：

```
//获取Header变量OAM_REMOTE_USER的值作为用户登录名
String userName = request.getHeader("OAM_REMOTE_USER");
if (userName == null || "".equals(userName)) {
    //跳转到登录页面，让用户认证
}
//根据需要设置登录当前系统的相关信息
```

关于 Header 变量的说明：

- Header 变量的值是由 OAM 服务器自动生成的。能根据应用的实际需要，在 OAM 控制台中配置 Header 变量的名称和值。
- Header 变量的安全性是可以保证的。用户的每一次请求都会被 Webgate 拦截并验证，只有验证通过之后，系统才会生成 OAM_REMOTE_USER 的值，否则其值为空。对于 Header 变量值的篡改，一般是由客户端发起的一个新的请求，而这个请求肯定是无法通过 Webgate 验证的，所以 Header 是安全的。
- 须确保用户访问应用系统任何被 4A 平台保护的资源 URL 都能执行上述代码，而一般做法是将这些代码加到过滤器中。

6.5.1.2 API 方式接入

API 是另一种单点登录集成方式，无须安装拦截器。这种方式只在应用系统无法通过安装拦截器的方式集成单点登录时使用。

对于这类应用系统集成，需要对应用系统进行改造：

- 改造应用系统的登录逻辑，使其调用 4A 平台提供的 Webservice 接口获取一个 token，并将 token 作为 cookie 的值，注意 cookie 名称必须为 ObSSOCookie。该改造实现用户登录该应用系统能够单点登录到集成了 4A 平台的其它应用系统。代码示例如下，建议应用系统自己验证成功之后，再执行：

```
// 创建WebService接口的代理实例
ProxyProvider provider = ProxyProvider.getInstance();
AccessServiceImplProxy proxy = provider.getProxy();

// 创建请求对象
LoggedInRequest = new LoggedInRequest(aUserID,
                aPassword);
try {
    // 调用WebService中的登录方法,发送请求并从响应消息中取出OAM认证Cookie字符串
    String sessionToken = proxy.loggedIn(loggedInRequest)
                    .getSessionToken();
    // 装载配置文件sso.properties
    PropertiesReader propReader = PropertiesReader.getInstance();
    // 读取配置文件中的OAM认证Cookie名称配置值:ObSSOCookie
    String tokenName = propReader.getValue(ConstValue.MS_SESSION_TOKEN);
    // 以ObSSOCookie为名称将OAM认证Cookie字符串包装成客户端Cookie
    Cookie obssoCookie = new Cookie(tokenName, sessionToken);
            obssoCookie.setPath("/");
    // 将Cookie写到请求对象的父域上面 --获取访问时的域名，不取配制文件
    String host = request.getServerName();
            obssoCookie.setDomain(host.substring(host.indexOf(".")));
            response.addCookie(obssoCookie);
} catch (RemoteException re) {
    System.out.println(re.getMessage());
}
```

- 改造过滤器，使其调用 4A 平台提供的 Webservice 接口，从 ObSSOCookie 的值中获取用户名。该改造实现用户能够从集成了 4A 平台的其它应用系统单点登录到该应用系统。代码示例如下：

```
Cookie[] cookies = request.getCookies();
for (int i = 0, size = cookies.length; i < size; i++) {
    Cookie cookie = cookies[i];
    if (cookies != null) {
        String cookieValue = cookie.getValue();
        // 创建WebService接口的代理实例
        ProxyProvider provider = ProxyProvider.getInstance();
        AccessServiceImplProxy proxy = provider.getProxy();
        // 创建请求对象
        GetUserNameRequest = new GetUserNameRequest(cookieValue);
        // 调用WebService中的解释Cookie方法,发送请求并从响应消息中取出用户名
        GetUserNameResponse userNameResponse = proxy.getUserName(getUserNameRequest);
        if (userNameResponse != null) {
            String userName = userNameResponse.getUserName();
            if (userName == null || "".equals(userName)) {
                //跳转到登录页面，让用户认证
            }
```

```
            //根据需要设置登录当前系统的相关信息

        }
    }
}
```

通过这种 API 的方式集成，只能实现单域名的单点登录，即应用登录 abc.midea.com，只能单点登录到其它.midea.com 的应用，不能单点登录到.midea.com.cn 的其它应用。这种情况下，如果应用系统需要支持跨域，需增加一个应用改造地方：应用系统登录成功之后，通过 iframe 的方式调用远程链接，生成另一个域下的 cookie，cookie 的名称为 ObSSOCookie。

一、如果应用是.midea.com.cn，则调用：

```
<iframe src="<iframe src="
http://login.midea.com:9090/CookieService/setCookie?name=ObSSOCookie&token=the value of
ObSSOCookie" width="0" height="0" frameborder="0"></iframe>" width="0" height="0"
frameborder="0"></iframe>
```

二、如果应用是.midea.com，则调用：

```
<iframe src="<iframe src="
http://login.midea.com.cn:9090/CookieService/setCookie?name=ObSSOCookie&token=the value of
ObSSOCookie" width="0" height="0" frameborder="0"></iframe>" width="0" height="0"
frameborder="0"></iframe>
```

6.5.2 成熟产品应用接入规范

对于成熟产品类应用，一般都可以通过产品配置来实现单点登录。

6.5.3 单点登出

不管是定制开发应用还是成熟产品应用，只需配置统一单点登出链接：

http://login.midea.com.cn/mideaidm-logout?end_url=应用单点登录访问地址

6.6 系统运维注意

6.6.1 应用服务器重新启动时，最好也重新启动代理服务器 Apache/IIS/HIS。

6.6.2 对于访问量较大的应用系统，需要注意 Apache 和 Webgate 的日志量，以免磁盘空间不足。

附录一 LDAP Schema 定义

1.用户对象属性表

下游应用系统同步用户数据，需使用 uniqueIdentifier 作为唯一判断，新增或更新数据。由于 LDAP 不同于关系型数据库，其字段长度是没有限制的，如果应用无特别要求，建议全部统一成 1024。

属性名称	类型	是否多值	说明
uid	DirectoryString	否	帐号 ID
uniqueIdentifier	DirectoryString	否	同步唯一标识
midea-userType	DirectoryString	否	帐号类型，代号含义： 0 - 内部用户 / 1 - 内部临时

			2 - 测试账户 / 3 - 应用账号 4 - 职能账户 / 5 - 外部用户 6 - 公共通讯录/7 - 供应商用户 8 - TC 门店用户/9 - 经销商 10 - 产业链用户
midea-alias	DirectoryString	否	帐号别名
mail	DirectoryString	否	邮件地址
mailalternateaddress	DirectoryString	否	邮件别名
cn	DirectoryString	否	全名
midea-py	DirectoryString	否	全名拼音
sn	DirectoryString	否	姓氏
givenname	DirectoryString	否	名字
midea-gender	DirectoryString	否	性别，代号含义： 1 - 男 / 2 - 女
createtimestamp	GeneralizedTime	否	创建时间
modifytimestamp	GeneralizedTime	否	上次修改时间
nsaccountlock	DirectoryString	否	帐号状态 锁定：true / 正常：null, false, 空白
employeenumber	DirectoryString	否	员工编号
midea-empstatus	DirectoryString	否	员工状态，代号含义： 0 - 未入司 / 1 - 试用期 2 - 正式 / 3 - 离职 4 - 退休 / 5 - 内退 6 - 离休 / 7 - 待岗 8 - 实习 / 9 - 调动中 10 - 派遣工
manager	DirectoryString	是	HR 设置的上级领导
midea-jobclass	DirectoryString	否	职种
midea-companyname	DirectoryString	否	公司名称
midea-companynumber	DirectoryString	否	公司编码
midea-departmentname	DirectoryString	否	部门名称
departmentnumber	DirectoryString	否	部门编码
midea-positiontype	DirectoryString	否	职位类型
midea-positionname	DirectoryString	否	职位名称

midea-position	DirectoryString	否	职位编码
midea-rank	DirectoryString	否	职等
postaladdress	DirectoryString	否	邮寄地址
postalcode	DirectoryString	否	邮政编码
mobile	DirectoryString	否	移动电话
telephonenumber	DirectoryString	否	座机
facsimileTelephoneNumber	DirectoryString	否	传真
description	DirectoryString	否	描述

2.部门对象属性表

属性名称	类型	是否多值	说明
o	DirectoryString	否	组织机构编号，同步唯一标识
postaladdress	DirectoryString	否	部门地址
midea-ostyle	DirectoryString	否	组织类型，代码含义： 0 - 内部行政组织 / 1 - 内部临时组织 2 - 系统帐号组织 / 3 - 职能帐号组织 4 - 应用帐号组织 / 5 - 外部帐号组织 6 - 公共通讯录组织
midea-vparentid	DirectoryString	否	虚拟上级组织机构节点
midea-otype	DirectoryString	否	组织单元类型
midea-orgarea	DirectoryString	否	所属地区
midea-isroot	DirectoryString	否	是否根节点
midea-departmentname	DirectoryString	否	组织单元名称
departmentnumber	DirectoryString	否	部门编号
midea-reportcode	DirectoryString	否	组织汇报关系
midea-labelcode	DirectoryString	否	组织标签编码
midea-attrident	DirectoryString	否	业务属性
midea-style	DirectoryString	否	组织形式
midea-orgtype	DirectoryString	否	组织类型
midea-parentid	DirectoryString	否	上级部门编号
midea-displayid	DirectoryString	否	显示顺序
displayname	DirectoryString	否	部门显示名称
lastparentid	DirectoryString	否	记录挂靠组织原父组织编号
midea-personIncharge	DirectoryString	是	部门主管

3.职位对象属性表

属性名称	类型	是否多值	说明
cn	DirectoryString	否	职位代码
midea-position	DirectoryString	否	职位名称
displayName	DirectoryString	否	显示名称
departmentNumber	DirectoryString	否	部门编号
midea-parentid	DirectoryString	否	上级职位编码
midea-parentname	DirectoryString	否	上级职位名称
midea-positionType	DirectoryString	否	职类类型
midea-departmentName	DirectoryString	否	部门名称
midea-jobclass	DirectoryString	否	职群、职种
midea-isRoot	DirectoryString	否	是否根节点 0 - 否 / 1 - 是
midea-rank	DirectoryString	否	职级
midea-positionGrade	DirectoryString	否	职等

附录二 LDAP API 代码示例

运行以下样例代码必须安装 JDK1.5 或以上的开发环境，另外，由于大部份属性都是允许空值的，同步数据时，需要注意做字符空处理和 null 处理。

1. 数据同步

```
//连接参数容器
Hashtable env = new Hashtable();
//JNDI 方式
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
//联系管理员获取连接的服务器地址和端口,示例里是现在的测试环境
env.put(Context.PROVIDER_URL, "ldap://10.16.15.252:389");
//静态密码认证方式
env.put(Context.SECURITY_AUTHENTICATION, "simple");
//通过流程申请获得连接用户名和密码
env.put(Context.SECURITY_PRINCIPAL, "uid=pms_bind,ou=Applications,o=midea.com.cn,o=isp");
env.put(Context.SECURITY_CREDENTIALS, "123456");
// Enable connection pooling (解决连接被重置问题 - Connection reset)
env.put("com.sun.jndi.ldap.connect.pool", "true");
LdapContext ctx = null;
try {
    ctx = new InitialLdapContext(env, null);
    //与或非写法示例
(&(midea-userType=0)(|(uid=xiejj3)(uid=yangps))(!(nsAccountLock=true)))
    String searchFilter = "(objectClass=*)";
    //设置查询的基准结点,参考规范文档
    String searchBase = "o=midea.com.cn,o=isp";
    SearchControls searchCtls = new SearchControls();
    //设置返回的属性，可以使用*来返回大部份的属性值，但如
```

```
nsaccountlock/passwordpolicysubentry/passwordexpirationtime/modifytimestamp/modifiersnamek
这些特殊属性需要明确写出来才会返回
    String returnedAtts[]={"*","nsaccountlock", "modifytimestamp"};
    searchCtls.setReturningAttributes(returnedAtts);
    //设置查询范围为所有子树下
    searchCtls.setSearchScope(SearchControls.SUBTREE_SCOPE);
    int totalResults = 0;
    //查询
    NamingEnumeration answer = ctx.search(searchBase, searchFilter, searchCtls);
    //循环读取返回数据
    while (answer.hasMoreElements()) {
        SearchResult sr = (SearchResult)answer.next();
        totalResults++;
        System.out.println(">>>" + sr.getName());
        //打印出数据信息
        Attributes attrs = sr.getAttributes();
        if (attrs != null) {
        try {
            //objectClass 是多值属性,循环获取
            for(int i = 0 ; i < attrs.get("objectclass").size(); i++)
                System.out.println(attrs.get("objectclass").get(i));
            System.out.println("   account: " + attrs.get("uid").get());
            System.out.println("   cn: " + attrs.get("cn").get());
            }
            catch (NullPointerException e) {
                System.err.println("Error listing attributes: " + e);
            }
        }
    }
    if  (answer !=  null)  {
     try  {
        answer.close();
     }  catch  (NamingException  ne)  {
       System.out.println("Failed  to  close  :  ");
     }
    }

System.out.println("Total results: " + totalResults);
} catch (NamingException e) {
    e.printStackTrace();
}
catch (Exception ex) {
    ex.printStackTrace();
}
finally
{
    try {
        if(ctx != null)
            ctx.close();
    } catch (Exception ex) {}
}
```

2.绑定帐号认证

```
boolean result = false;
//连接参数容器
Hashtable env = new Hashtable();
//JNDI 方式
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
//联系管理员可以获取连接的服务器地址和端口
env.put(Context.PROVIDER_URL, "ldap://10.16.15.252:389");
//静态密码认证
env.put(Context.SECURITY_AUTHENTICATION, "simple");
//设置需要认证的用户和密码
env.put(Context.SECURITY_PRINCIPAL, "uid=pms_bind,ou=Applications,o=midea.com.cn,o=isp");
env.put(Context.SECURITY_CREDENTIALS, "123456");
// Enable connection pooling (解决连接被重置问题 - Connection reset)
env.put("com.sun.jndi.ldap.connect.pool", "true");
try {
    LdapContext ctx = new InitialLdapContext(env, null);
    result = true; //认证成功
} catch (NamingException e) { //认证失败
    e.printStackTrace();
 }
catch (Exception ex) //认证失败
{
    ex.printStackTrace();
}
finally
{
    try {
        if (ctx != null) ctx.close();
    } catch (Exception ex){}
}
```

3.用户帐号认证

```
    /**
     * @param useruid - 输入用户账号值 格式："liming"
     * @param userPassword - 输入用户密码 格式："123456"
     * @param applicationUsername - 输入绑定账号用户名 格式：
"uid=pms_bind,ou=Applications,o=midea.com.cn,o=isp"
     * @param applicationPassword - 输入绑定账号密码 格式："123456"
     * @return - 返回 true 或 false
     * @throws NamingException
     */
    @SuppressWarnings({ "unused", "rawtypes", "unchecked" })
    public boolean getAuthenticator(String useruid, String userPassword,String
applicationUsername, String applicationPassword) throws NamingException {
        boolean result = false;
        boolean flag = false;
        DirContext ctx = null;
        Hashtable env = new Hashtable();
        // 初始化绑定账号连接参数
```

```
        env.put(Context.INITIAL_CONTEXT_FACTORY,"com.sun.jndi.ldap.LdapCtxFactory");
        //LDAP 服务器地址，找 4A 项目组成员提供
        env.put(Context.PROVIDER_URL, "ldap://10.16.66.203:1389");
        // 静态密码认证
        env.put(Context.SECURITY_AUTHENTICATION, "simple");
        //链接 LDAP 的应用帐号，找 4A 项目组成员提供
        env.put(Context.SECURITY_PRINCIPAL, applicationUsername);
        //链接 LDAP 的应用账号密码，找 4A 项目组成员提供
        env.put(Context.SECURITY_CREDENTIALS, applicationPassword);
        try {
            ctx = new InitialDirContext(env);
            System.out.println("Applications Authentication Succeed");
            flag = true;// 绑定账号认证成功
        } catch (NamingException e) {
            if (ctx != null) {
                try {
                    ctx.close();// 关闭连接
                    System.out.println("-----------------------------------------");
                } catch (NamingException e1) {
                    System.out.println(" Failed to close : " + e1);
                }
            }
            System.out.println(" Failed to inactivated : " + e);
            flag = false;// 绑定账号认证失败
            return false;
        }
        SearchControls constraints = new SearchControls();
        // 认证成功后查询用户
        if (flag) {
            NamingEnumeration<SearchResult> en = null;
            try {

                constraints.setSearchScope(SearchControls.SUBTREE_SCOPE);
                // 设置返回属性 "*" 代表LDAP中用户所有属性
                String[] uids = { "*" };
                constraints.setReturningAttributes(uids);
                // 设置查询的基准结点，联系管理员来获得
                String basedn = "o=midea.com.cn,o=isp";
                // 用户账号值 格式： "liming"
                String searchFilter = "(uid=" + useruid + ")";
                en = ctx.search(basedn, searchFilter, constraints);
            } catch (Exception e) {
                System.out.println("Exception in search():" + e);
            }
            // 如果查询到用户账号在绑定账号中存在
            if (en.hasMoreElements()) {
                DirContext ctxUser = null;
                Hashtable envUser = new Hashtable();

    envUser.put(Context.INITIAL_CONTEXT_FACTORY,"com.sun.jndi.ldap.LdapCtxFactory");
                envUser.put(Context.PROVIDER_URL, "ldap://10.16.66.203:1389");
```

```
                envUser.put(Context.SECURITY_AUTHENTICATION, "simple");

                // 用户账号值 格式："liming"
                envUser.put(Context.SECURITY_PRINCIPAL, en.next().getNameInNamespace());
                // 用户密码 格式："123456"
                envUser.put(Context.SECURITY_CREDENTIALS, userPassword);
                try {
                    ctxUser = new InitialDirContext(envUser);
                    System.out.println("User Login Succeed");
                    result = true;// 用户认证成功
                    constraints.setTimeLimit(1000);
                    constraints.setDerefLinkFlag(false);
                    constraints.setReturningAttributes(new String[] { "*" });

                } catch (NamingException e) {
                    System.out.println("Failed to close -1 " + e);
                } finally {
                    if (ctxUser != null) {
                        try {
                            ctxUser.close();
                        } catch (NamingException e1) {
                            System.out.println("Failed to close -1 " + e1);
                        }
                    }
                }
            } else {
                result = false;
            }
            try {
                if (en != null) {
                    en.close();
                }

                if (ctx != null) {
                    ctx.close();
                }
            } catch (NamingException e) {
                System.out.println("Exception in search():" + e);
            }
        }

        return result;
    }
```

4.同步静态组用户

```
//用户对象
public class UserInfo {
```

```
    private String departmentName;
    private String cn;

    public String getDepartmentName() {
        return departmentName;
    }
    public void setDepartmentName(String departmentName) {
        this.departmentName = departmentName;
    }

    public String getCn() {
        return cn;
    }
    public void setCn(String cn) {
        this.cn = cn;
    }

}




package com.bizenit.util;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Hashtable;
import java.util.List;

import javax.naming.CompositeName;
import javax.naming.Context;
import javax.naming.Name;
import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.naming.directory.Attribute;
import javax.naming.directory.Attributes;
import javax.naming.directory.BasicAttribute;
import javax.naming.directory.DirContext;
```

```
import javax.naming.directory.InitialDirContext;
import javax.naming.directory.ModificationItem;
import javax.naming.directory.SearchControls;
import javax.naming.directory.SearchResult;
import com.bizenit.bean.UserInfo;

public class LdapUtil {

    public  DirContext ctx = null;
    /**
     * get LDAP connection
     */
    public  DirContext getConLDAP(String admin, String password, String LDAPip,
            String LDAPport) {
        Hashtable<String, String> env = new Hashtable<String, String>();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.jndi.ldap.LdapCtxFactory");// 必须这样写，无论用什么 LDAP 服务器。
        env.put(Context.PROVIDER_URL, "ldap://" + LDAPip + ":" + LDAPport);// LDAP 服务器
的地址:端口
        env.put(Context.SECURITY_AUTHENTICATION, "simple");// 授权界别，可以有三种授权级
别
        env.put(Context.SECURITY_PRINCIPAL, admin);// 载入登陆帐户和登录密码
        env.put(Context.SECURITY_CREDENTIALS, password);
        try {
            if (ctx == null){
                ctx = new InitialDirContext(env);// 初始化上下文
                LOG.info("认证成功");// 这里可以改成异常抛出。
            }
        } catch (javax.naming.AuthenticationException e) {
            LOG.info("认证失败");
        } catch (Exception e) {
            LOG.info("认证出错：" + e);
        }
        return ctx;
    }



    /**
     * 获取静态组成员
     */
```

```
        public List<UserInfo> getStaticGroupMember(String cn) {
            List<String> uniqueMembers = new ArrayList<String>();
            List<UserInfo> users = new ArrayList<UserInfo>();
            SearchControls constraints = new SearchControls();
            constraints.setSearchScope(SearchControls.SUBTREE_SCOPE);
            constraints.setReturningAttributes(new String[] { "*" });// 返回属性值
            String filter = "(&(!(objectClass=groupofurls))(cn=" + cn + "))";
            try {
                NamingEnumeration results =
ctx.search("ou=Groups,o=midea.com.cn,o=isp",filter, constraints);
                while (results != null && results.hasMore()) {
                    SearchResult sr = (SearchResult) results.next();
                    Attributes atts = sr.getAttributes();
                    Attribute cnAttr = atts.get("cn");
                    Attribute uniqueMemberAttr = atts.get("uniquemember");
                    if (uniqueMemberAttr != null) {
                        NamingEnumeration<Object> uniqueMes = (NamingEnumeration<Object>)
uniqueMemberAttr.getAll();
                        while (uniqueMes.hasMoreElements()) {
                            Object mObj = uniqueMes.nextElement();
                            uniqueMembers.add(mObj.toString());
                            String uniq = mObj.toString();
                            String uid = uniq.substring(uniq.indexOf("=")+1,
uniq.indexOf(","));
                            //获取用户对象
                            UserInfo userInfo = this.getUserInfoByUid(uid);
                            System.out.println("cn:"+userInfo.getCn() + "; uid:"+uid +";
deptName:"+userInfo.getDepartmentName());
                            users.add(userInfo);
                        }
                    }

                }
            } catch (NamingException e) {
                LOG.error("getStaticGroupsByUserID occur NamingException" + e);
            }
            return users;
        }
```

```
        /**
         * 根据uid获取用户信息
         * @param uid
         * @return
         */
        public   UserInfo getUserInfoByUid (String uid){
            UserInfo userInfo = null;
            SearchControls constraints = new SearchControls();
            constraints.setSearchScope(SearchControls.SUBTREE_SCOPE);
            constraints.setReturningAttributes(new String[] {"uniqueIdentifier","*"});// 返回
属性值
            String filter = "(&(objectClass=midea-person)(uid=" + uid +"))";
            try {
                NamingEnumeration results = ctx.search("o=midea.com.cn,o=isp", filter,
constraints);
                while (results != null && results.hasMore()){
                    userInfo =  new UserInfo();
                    SearchResult sr = (SearchResult) results.next();
                    String userDN = sr.getNameInNamespace();
                    userInfo.setUserDN(userDN);
                    Attributes attributes = sr.getAttributes();
                    //获取用户姓名
                    Attribute cnattribute =  attributes.get("cn");
                    if (cnattribute!=null){
                    String cn = (String) cnattribute.get();
                    userInfo.setCn(cn);
                    }
                    //获取用户组织
                    Attribute deptNameattribute =  attributes.get("midea-departmentName");
                    if (deptNameattribute!=null){
                    String deptName = (String) deptNameattribute.get();
                    userInfo.setDepartmentName(deptName) ;
                    }
                    //获取其他属性需要自己填写
                }
            } catch (NamingException e) {
                LOG.error("getUserInfoByUid occur NamingException"+e);
            }

            return userInfo;
```

```
        }


        /**
         * close LDAP connection
         */
        public  void closeLDAPCon() {
            try {
                if (ctx != null)
                    ctx.close();
                System.out.println("关闭连接!! ");
            } catch (Exception e) {
                System.out.println("关闭连接异常!! ");
            }
        }


        public static void main (String [] args) {
            LdapUtil util = new LdapUtil();
            util.getConLDAP("cn=Directory Manager", "!QAZMidea", "10.16.15.54", "8389");
            util.getStaticGroupMember("ec_vip");

        }

    }
```

附录三 LDAP 应用接入申请流程

1. 流程位置

集团管控流程_08 流程与 IT 管理_0802 运营维护_080202IT 基础架构资源管理_08020201 网络及安全资源管理_0802020101 应用接入美的邮件系统及 LDAP 系统绑定帐号申请流程

2. 流程样例

应用系统接入美的邮件系统及LDAP或AD系统绑定帐号申请流程

附录四 应用接入信息收集表

内容	测试环境	生产环境
系统名称	合并预算系统	合并预算系统
英文简称	HBYC	HBYC
操作系统	Red Hat Enterprise Linux Server Release 5.5	Red Hat Enterprise Linux Server Release 5.5
系统位数	64 位	64 位
服务器 IP	10.16.66.45	10.16.15.38
服务器帐号和密码	hbyc/Abcd1234	hbyc/Abcd1234
应用内网地址	http://10.16.66.45:19000/workspace	http:// 10.16.15.38:19000/workspace
应用外网地址	无	无
应用测试帐号和密码	fangxx/Abcd1234	fangxx/Abcd1234
应用服务器	Websphere 6.0	Websphere 6.0
代理服务器	IHS6.0	IHS6.0
部署架构	单机	双机集群
是否已集成 LDAP	是	是

附录五 参考资料

[1] JDNI 指导

http://java.sun.com/j2se/1.4.2/docs/guide/jndi

[2] RFC1777 LDAP

http://www.faqs.org/rfcs/rfc1777.html

[3] API 参考

http://docs.oracle.com/javase/1.4.2/docs/api/javax/naming/directory/DirContext.html

http://ldaptemplate.sourceforge.net/quickstart.pdf

http://docs.spring.io/spring-ldap/docs/1.3.2.RELEASE/reference/html/user-authentication.html

[4]连接池使用参考

http://docs.oracle.com/javase/jndi/tutorial/ldap/connect/pool.html

http://anzuo.blog.sohu.com/137895406.html

[5]美的 LDAP 或单点登录集成 QQ 工作群：289079359，申请加入时请注明系统名称和管理员姓名。