COFrame 概要设计说明书

文档修订记录

序号	版本号	修订日期	修订概述	修订人	审核人	批准人	备注
1.							
2.							
3.							
4.							
5.							
6.							
7.							
8.							

目　录

1 引言

1.1 编写目的

本设计说明书文档包括该项目的建设背景、目标、建设内容、系统架构、接口、数据模型、功能模型、部署模型、功能设计等的描述，用于指导该项目的开发与部署，同时，作为该项目的重要技术资料，作为系统未来维护或扩展的参考。

本文档的阅读者为本系统的设计、开发人员、接口系统的开发人员、系统维护人员。

名词术语

1.2 名词术语

COFrame 应用基础框架

1.3 参考资料

COFrame 数据模型

EOS7 开发手册

EOS7 帮助文档

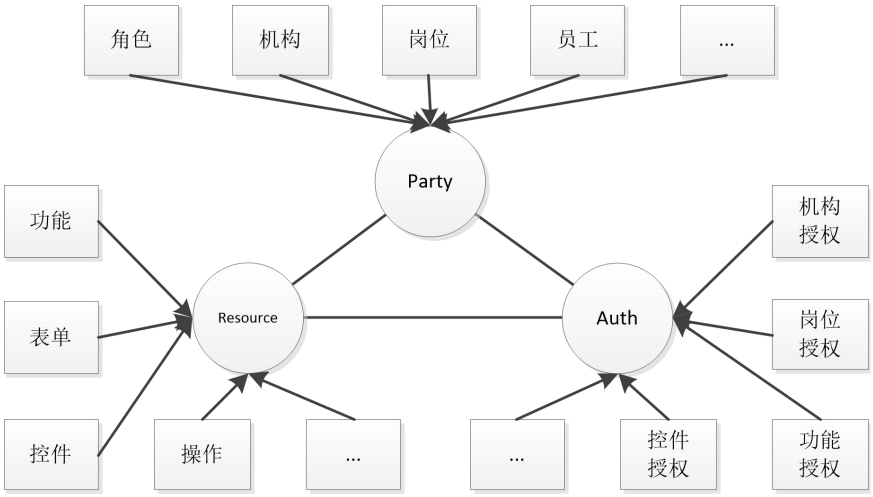
1.4 文档约定

2 参与者与组织机构设计

EOS7 采用新的参与者模型设计，抽象出了 Party（参与者）、Resource（资源）与 Auth

（授权）等概念以及三者的关系描述。基于参与者、资源与授权等概念可扩展开发出符合用户个性化需求的参与者模型。

EOS7 的参与者、资源与授权的关系如下：



Party 是对系统参与者的抽象，可以实例化为角色、机构、岗位、员工和用户等内容，Resource 是对业务资源的抽象，可以实例化为功能、表单、控件和操作等内容，授权则是角色对资源和其他参与者的权限设置。

基于该模型，用户实现个性化的组织机构及授权相关接口时需要实现：

1） 参与者类型接口，定义参与者类型，如机构、岗位、员工等。

2） 资源相关接口，如功能、表单、表单控件、视图等。

3） 授权相关接口，资源授权和参与者授权。

4） COFrame 提供默认的组织机构授权实现。

2.1 参与者模型及接口说明

用户定义参与者时需要实现两个接口：

1） 参与者类型接口：定义参与者类型，如机构、岗位、员工等，具体接口定义如下：

```
package com.primeton.cap.party;

import java.util.List;
```

```
/**
 * partyType数据服务接口，不考虑增删改
 *
 * @author guwei (mailto:guwei@primeton.com)
 */
public interface IPartyTypeDataService{

    /**
     * 获取所有party的相关数据列表，比如获取应用中的角色列表
     * @param tenantID
     * @return
     */
    List<Party> getAllPartyList(String tenantID);

    /**
     * 获取顶级的party相关数据
     * @param tenantID
     * @return
     */
    List<Party> getRootPartyList(String tenantID);

    /**
     * 根据partyID获取Party数据
     *
     * @param partyID
     * @param tenantID
     * @return
```

```
     */
    Party getPartyByPartyID(String partyID, String tenantID);

}
```

2） 参与者关系接口：参与者关系接口要求实现两种参与者之间的关系，这种关系有两种：

a) 上下级关系：如机构员工关系、机构岗位关系、机构机构关系、岗位员工关系等。

b) 授权关系：如机构角色关系、岗位角色关系、员工角色关系等。

参与者关系接口定义如下：

```
package com.primeton.cap.party;

import java.util.List;

/**
 * 关联party数据服务接口，目前只考虑查询，增删改不考虑
 *
 * @author guwei (mailto:guwei@primeton.com)
 */
public interface IPartyTypeRefDataService{

    /**
     * 根据父partyID获取父子关系中的子party数据列表
     *
     * @return
     */
    List<Party> getChildrenPartyList(String parentPartyID, String tenantID);

    /**
```

```
     * 根据子partyID获取父子关系中的父Party数据，返回值设置为列表是因为存在授权
关系
     * 在授权关系中，一个人员可以属于多个角色
     *
     * @param childPartyID
     * @param tenantID
     * @return
     */
    List<Party> getParentPartyList(String childPartyID, String tenantID);
}
```

3） 用户定义的参与者类型和参与者关系接口需要向 EOS7 的参与者模型管理注册，注册的方式是调用参与者模型管理相关的接口。

a) 参与者类型必须在应用启动时通过调用 PartyTypeManager 的 putPartyType 方法注册参与者。

b) 参与者关系必须在应用启动时通过调用 PartyTypeRefManager 的 putPartyTypeRef 方法注册参与者关系。

c) COFrame 中在 org.gocom.components.coframe.auth 构件包的启动监听器 org.gocom.components.coframe.auth.startup.AuthStartupContributionListener.java 中调用了上述接口注册参与者，并通过解析 org.gocom.components.coframe.auth 构件包下的 META-INF/party-config.xml 配置文件注册参与者以及参与者的关系。

party-config.xml 的示例配置如下：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<PartyModel>
    <PartyTypeList>
        <PartyType typeID="role" name="角色"
partyTypeDataService="org.gocom.components.coframe.auth.party.impl.RolePart
yTypeDataService" description="" isRole="true" priority="2" icon16path=""
```

```
icon32path="" showInTree="true" showAtRoot="true" isLeaf="false"/>
        <PartyType typeID="org" name="机构"
partyTypeDataService="org.gocom.components.coframe.org.party.impl.OrgPartyT
ypeDataService" description="" isRole="false" priority="1" icon16path=""
icon32path="" showInTree="true" showAtRoot="true" isLeaf="false"/>
        <PartyType typeID="position" name="岗位"
partyTypeDataService="org.gocom.components.coframe.org.party.impl.PositionP
artyTypeDataService" description="" isRole="false" priority="3" icon16path=""
icon32path="" showInTree="true" showAtRoot="false" isLeaf="false"/>
        <PartyType typeID="emp" name="员工"
partyTypeDataService="org.gocom.components.coframe.org.party.impl.EmpPartyT
ypeDataService" description="" isRole="false" priority="5" icon16path=""
icon32path="" showInTree="true" showAtRoot="false" isLeaf="true"/>
        <PartyType typeID="user" name="用户"
partyTypeDataService="org.gocom.components.coframe.auth.party.impl.UserPart
yTypeDataService" description="" isRole="false" priority="6" icon16path=""
icon32path="" showInTree="false" showAtRoot="false" isLeaf="true"/>
        <PartyType typeID="orgRole" name="机构角色"
partyTypeDataService="org.gocom.components.coframe.auth.party.impl.OrgRoleP
artyTypeDataService" description="" isRole="false" priority="4" icon16path=""
icon32path="" showInTree="true" showAtRoot="false" isLeaf="false"/>
    </PartyTypeList>


    <PartyTypeRefList>
        <PartyTypeRef refID="emp_user_ref" refName="员工用户" refType="p_c"
parentPartyTypeID="emp" childPartyTypeID="user"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.E
mpUserRefDataService"/>
```

```
        <!-- 组织机构内部关系 -->
        <PartyTypeRef refID="org_org_ref" refName="机构机构" refType="p_c"
parentPartyTypeID="org" childPartyTypeID="org"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.Or
gOrgRefDataService"/>
        <PartyTypeRef refID="org_pos_ref" refName="机构岗位" refType="p_c"
parentPartyTypeID="org" childPartyTypeID="position"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.Or
gPosRefDataService"/>
        <PartyTypeRef refID="org_emp_ref" refName="机构员工" refType="p_c"
parentPartyTypeID="org" childPartyTypeID="emp"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.Or
gEmpRefDataService"/>
        <PartyTypeRef refID="pos_pos_ref" refName="岗位岗位" refType="p_c"
parentPartyTypeID="position" childPartyTypeID="position"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.Po
sPosRefDataService"/>
        <PartyTypeRef refID="pos_emp_ref" refName="岗位员工" refType="p_c"
parentPartyTypeID="position" childPartyTypeID="emp"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.Po
sEmpRefDataService"/>

        <!-- 与角色相关的关系 -->
        <PartyTypeRef refID="role_user_ref" refName="角色用户" refType="r_p"
parentPartyTypeID="role" childPartyTypeID="user"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.R
oleUserRefDataService"/>
```

```
        <PartyTypeRef refID="role_emp_ref" refName="角色员工" refType="r_p"
parentPartyTypeID="role" childPartyTypeID="emp"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.R
oleEmpRefDataService"/>
        <PartyTypeRef refID="role_position_ref" refName="角色岗位"
refType="r_p" parentPartyTypeID="role" childPartyTypeID="position"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.R
olePositionRefDataService"/>
        <PartyTypeRef refID="role_org_ref" refName="角色机构" refType="r_p"
parentPartyTypeID="role" childPartyTypeID="org"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.R
oleOrgRefDataService"/>
    </PartyTypeRefList>
</PartyModel>
```

2.1.1 参与者类型

COFrame 实现了六种参与者：角色、机构、岗位、员工、用户、机构角色。

2.1.1.1 角色

角色是授权的基础，是最基本的参与者，COFrame 中实现角色的管理以及角色的参与者模型，配置如下：

```
<PartyType typeID="role" name="角色"
partyTypeDataService="org.gocom.components.coframe.auth.party.impl.RolePartyTyp
eDataService" description="" isRole="true" priority="2" icon16path=""
icon32path="" showInTree="true" showAtRoot="true" isLeaf="false"/>
```

注意：角色的类型名称必须是 role

角色对应的数据表为 CAP_ROLE。

字段	说明	数据类型
ROLE_ID	角色 ID	VARCHAR2
TENANT_ID	租户 ID	VARCHAR2
ROLE_CODE	角色代码	VARCHAR2
ROLE_NAME	角色名	VARCHAR2
ROLE_DESC	角色描述	VARCHAR2
CREATEUSER	创建者	VARCHAR2
CREATETIME	创建时间	TIMESTAMP

2.1.1.2 机构

机构是公司组织中重要组成单元， COFrame 中实现机构的管理以及机构的参与者模型，配置如下：

```
<PartyType typeID="org" name="机构"
partyTypeDataService="org.gocom.components.coframe.org.party.impl.OrgPartyTypeD
ataService" description="" isRole="false" priority="1" icon16path=""
icon32path="" showInTree="true" showAtRoot="true" isLeaf="false"/>
```

机构对应的数据库表为：ORG_ORGANIZATION。

字段	说明	字段类型
ORGID	机构编号	DECIMAL
ORGCODE	机构代码	VARCHAR
ORGNAME	机构名称	VARCHAR
ORGLEVEL	机构级别	DECIMAL

ORGDEGREE	机构等级	VARCHAR
PARENTORGID	上级机构编号	DECIMAL
ORGSEQ	序列号	VARCHAR
ORGTYPE	机构类型	VARCHAR
ORGADDR	机构地址	VARCHAR
ZIPCODE	邮编	VARCHAR
MANAPOSITION	机构主管岗位	DECIMAL
MANAGERID	机构主管编号	DECIMAL
ORGMANAGER	机构主管人	VARCHAR
LINKMAN	联系人	VARCHAR
LINKTEL	联系电话	VARCHAR
EMAIL	电子邮件	VARCHAR
WEBURL	网站地址	VARCHAR
STARTDATE	生效日期	VARCHAR
ENDDATE	失效日期	DATE
STATUS	机构状态	DATE
AREA	所属地域	VARCHAR
CREATETIME	建立时间	TIMESTAMP
LASTUPDATE	上一次更新时间	TIMESTAMP
UPDATOR	更新的操作人	DECIMAL
SORTNO	排列顺序	INT
ISLEAF	是否是叶子节点	CHAR
SUBCOUNT	子节点个数	DECIMAL
REMARK	备注	VARCHAR
TENANT_ID	租户ID	VARCHAR
APP_ID	应用ID	VARCHAR

2.1.1.3 岗位

岗位在公司组织中起着重要的作用， COFrame 中实现岗位的管理以及岗位的参与者模型，配置如下：

```
<PartyType typeID="position" name="岗位"
partyTypeDataService="org.gocom.components.coframe.org.party.impl.PositionParty
TypeDataService" description="" isRole="false" priority="3" icon16path=""
icon32path="" showInTree="true" showAtRoot="false" isLeaf="false"/>
```

岗位对应的数据库表为：ORG_POSITION。

字段	说明	字段类型
POSITIONID	岗位编号	DECIMAL
POSICODE	岗位代码	VARCHAR
POSINAME	岗位名称	VARCHAR
POSILEVEL	岗位级别	DECIMAL
MANAPOSI	副岗位	DECIMAL
DUTYID	职务编号	DECIMAL
ORGID	所属机构编号	DECIMAL
POSITIONSEQ	岗位序列号	VARCHAR
POSITYPE	岗位类型	VARCHAR
CREATETIME	建立时间	TIMESTAMP
LASTUPDATE	上一次更新时间	TIMESTAMP
UPDATOR	更新的操作人	DECIMAL
STARTDATE	生效时间	DATE
ENDDATE	失效时间	DATE
STATUS	岗位状态	VARCHAR

ISLEAF	是否是叶子节点	CHAR
SUBCOUNT	子节点个数	DECIMAL
TENANT_ID	租户ID	VARCHAR
APP_ID	应用ID	VARCHAR

2.1.1.4 员工

COFrame 中实现员工的管理以及员工的参与者模型，配置如下：

```
<PartyType typeID="emp" name="员工"
partyTypeDataService="org.gocom.components.coframe.org.party.impl.EmpPartyTypeD
ataService" description="" isRole="false" priority="5" icon16path=""
icon32path="" showInTree="true" showAtRoot="false" isLeaf="true"/>
```

员工对应的数据库表为：ORG_EMPLOYEE。

字段	说明	字段类型
EMPID	员工编号	DECIMAL
EMPCODE	员工代码	VARCHAR
OPERATORID	操作员的编号	DECIMAL
USERID	用户登录名（操作员）	VARCHAR
EMPNAME	员工姓名	VARCHAR
REALNAME	英文全名	VARCHAR
GENDER	性别	VARCHAR
BIRTHDATE	出生日期	DATE
POSITION	员工岗位	DECIMAL
EMPSTATUS	人员状态	VARCHAR

CARDTYPE	证件类型	VARCHAR
CARDNO	证件号码	VARCHAR
INDATE	入职日期	DATE
OUTDATE	离职日期	DATE
OTEL	办公室电话	VARCHAR
OADDRESS	办公地址	VARCHAR
OZIPCODE	办公室邮编	VARCHAR
OEMAIL	办公室邮件	VARCHAR
FAXNO	传真号码	VARCHAR
MOBILENO	手机号码	VARCHAR
QQ	QQ号码	VARCHAR
HTEL	家庭电话	VARCHAR
HADDRESS	家庭地址	VARCHAR
HZIPCODE	家庭邮编	VARCHAR
PEMAIL	私人邮箱	VARCHAR
PARTY	政治面貌	VARCHAR
DEGREE	职级	VARCHAR
MAJOR	直接主管	DECIMAL
SPECIALTY	可管理角色	VARCHAR
WORKEXP	工作描述	VARCHAR
REGDATE	注册日期	DATE
CREATETIME	建立时间	TIMESTAMP
LASTMODYTIME	上次修改时间	TIMESTAMP
ORGIDLIST	可管理机构	VARCHAR
ORGID	所属机构编号	DECIMAL
REMARK	备注	VARCHAR
TENANT_ID	租户ID	VARCHAR
APP_ID	应用ID	VARCHAR

WEIBO	微博	VARCHAR

设计变更：在 Coframe4.0.0 版本中，可管理机构的设置，由员工改到了角色之下，可管理机构的数据不再保存在 ORG_EMPLOYEE 的 ORGIDLIST 字段中，而是保存在数据库表 cap_resauth 中。原来保存在 ORG_EMPLOYEE 的 ORGIDLIST 字段中的可管理机构需要在授权管理>>机构 TAB 页中重新设置，原有数据不需要删除，对后续功能不会有影响。

2.1.1.5 用户

用户是 COFrame 中用于登录的基本元素，保存用户名密码。用户与员工有关联但是又不同：一个员工可以有一个登录用户，但是可以登录系统的不一定是员工，也可以是管理员或其他身份的用户。COFrame 中实现员工的管理以及员工的参与者模型，配置如下：

```
<PartyType typeID="user" name="用户"
partyTypeDataService="org.gocom.components.coframe.auth.party.impl.UserPartyTyp
eDataService" description="" isRole="false" priority="6" icon16path=""
icon32path="" showInTree="false" showAtRoot="false" isLeaf="true"/>
```

用户对应的数据库表为：CAP_USER。

字段	说明	数据类型
OPERATOR_ID	操作员 ID	DECIMAL
TENANT_ID	租户 ID	VARCHAR
USER_ID	登录用户名	VARCHAR
PASSWORD	密码	VARCHAR
INVALDATE	密码失效日期	DATE

USER_NAME	用户名称	VARCHAR
AUTHMODE	本地密码认证、LDAP 认证、等	VARCHAR
STATUS	正常，挂起，注销，锁定...	VARCHAR
UNLOCKTIME	当状态为锁定时，解锁的时间	TIMESTAMP
MENUTYPE	用户登录后菜单的风格	VARCHAR
LASTLOGIN	最后登录时间	TIMESTAMP
ERRCOUNT	密码错误次数	DECIMAL
STARTDATE	有效开始时间	DATE
ENDDATE	有效截止时间	DATE
VALIDTIME	定义一个规则表达式，表示允许操作的有效时间范围	VARCHAR
MACCODE	允许设置多个 MAC	VARCHAR
IPADDRESS	允许设置多个 IP 地址	VARCHAR
EMAIL	邮箱地址	VARCHAR
CREATEUSER	创建人	VARCHAR
CREATETIME	创建时间	TIMESTAMP

2.1.1.6 机构角色

机构角色是为了方便用户查询而设置的虚拟参与者，譬如某机构的部门经理。COFrame 中实现机构角色的参与者模型，配置如下：

```
<PartyType typeID="orgRole" name="机构角色"
partyTypeDataService="org.gocom.components.coframe.auth.party.impl.OrgRoleParty
TypeDataService" description="" isRole="false" priority="4" icon16path=""
icon32path="" showInTree="true" showAtRoot="false" isLeaf="false"/>
```

机构角色通过机构与角色的关联查询实现。

2.1.2 参与者关系

EOS7 的参与者之间为父子关系，根据父参与者获取所有子参与者或根据子参与者获取父参与者。COFrame 实现了六种参与者关系：员工用户、机构机构、机构岗位、机构员工、岗位岗位和岗位员工关系。

2.1.2.1 员工用户

在 COFrame 中，一个员工只能对应一个用户，该对应关系是通过员工表中的 OPERATORID 逻辑关联，在数据库表中并没有约束。

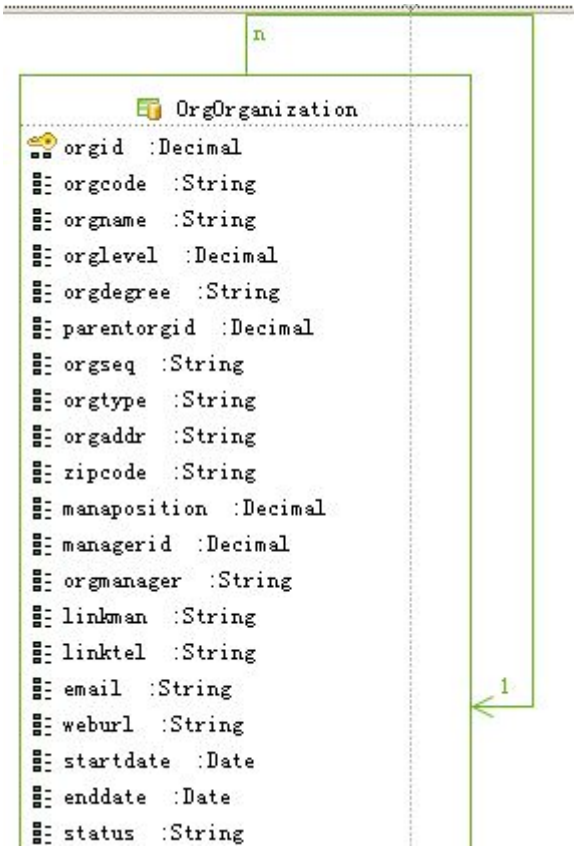
员工管理功能与组织机构树管理在同一个界面中，与组织机构树管理不可分割。而用户的管理则有两处：一是单独的用户管理功能，二是添加员工时自动添加用户。

员工用户关系在 COFrame 中配置如下：

```
<PartyTypeRef refID="emp_user_ref" refName="员工用户" refType="p_c"
parentPartyTypeID="emp" childPartyTypeID="user"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.EmpUs
erRefDataService"/>
```

2.1.2.2 机构机构

在 COFrame 中，机构是自关联的，一个机构下面可以放多个子机构，该对应关系是通过机构表中的 PARENTORGID 关联。

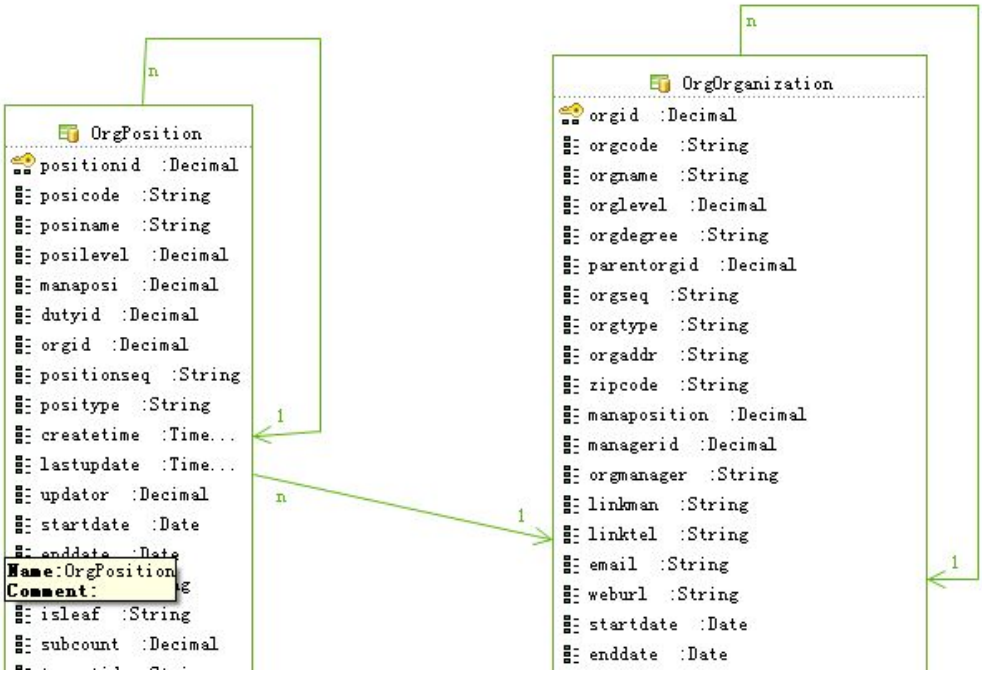


机构机构关系在 COFrame 中配置如下：

```
<PartyTypeRef refID="org_org_ref" refName="机构机构" refType="p_c"
parentPartyTypeID="org" childPartyTypeID="org"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.OrgOrg
RefDataService"/>
```

2.1.2.3 机构岗位

在 COFrame 中，机构与岗位是一对多的关系，一个机构可以拥有多个岗位。机构与岗位的关系关系如下：

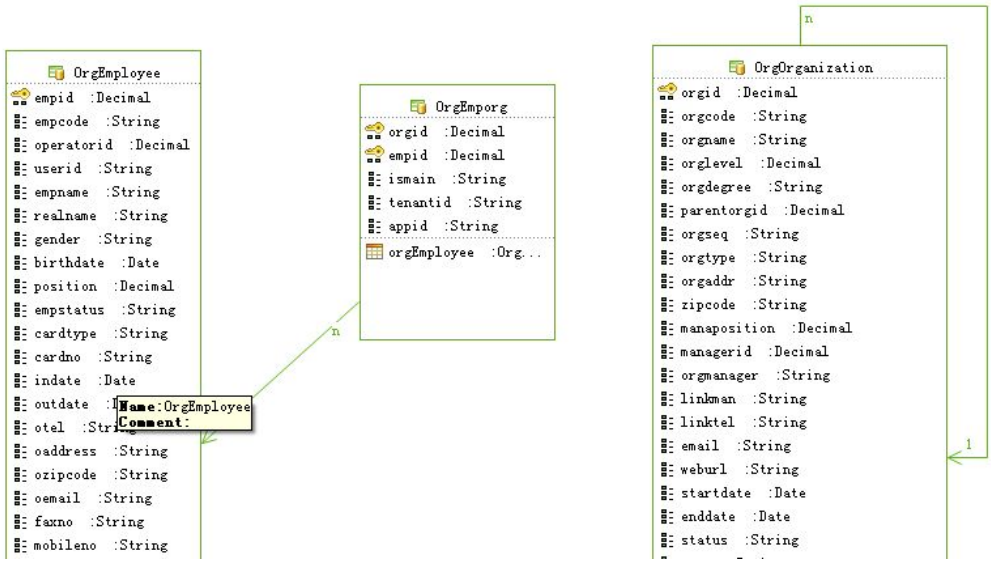


机构岗位关系在 COFrame 中配置如下：

```
<PartyTypeRef refID="org_pos_ref" refName="机构岗位" refType="p_c"
parentPartyTypeID="org" childPartyTypeID="position"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.OrgPos
RefDataService"/>
```

2.1.2.4 机构员工

在 COFrame 中，机构与员工是多对多的关系，一个机构可以拥有多个员工，而一个员工也可以隶属于多个机构。机构与员工的关联关系如下：



机构员工的关系通过机构员工表 ORG_EMPORG 关联

字段	说明	字段类型
ORGID	机构编号	DECIMAL
EMPID	员工编号	DECIMAL
ISMAIN	是否是主机构	CHAR
TENANT_ID	租户ID	VARCHAR
APP_ID	应用ID	VARCHAR

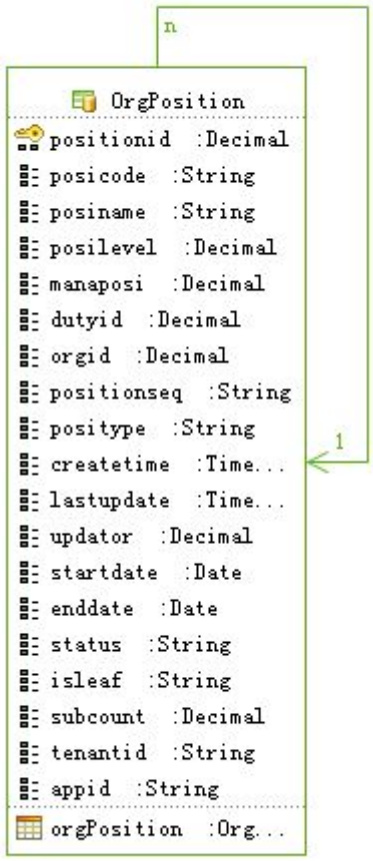
机构员工关系在 COFrame 中配置如下：

```
<PartyTypeRef refID="org_emp_ref" refName="机构员工" refType="p_c"
parentPartyTypeID="org" childPartyTypeID="emp"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.OrgEmp
RefDataService"/>
```

2.1.2.5 岗位岗位

在 COFrame 中，岗位是自关联的，一个岗位下面可以放多个子岗位，该对应关系是通过岗位表中的 MANAPOSI 字段关联。

岗位的自关联关系如下：

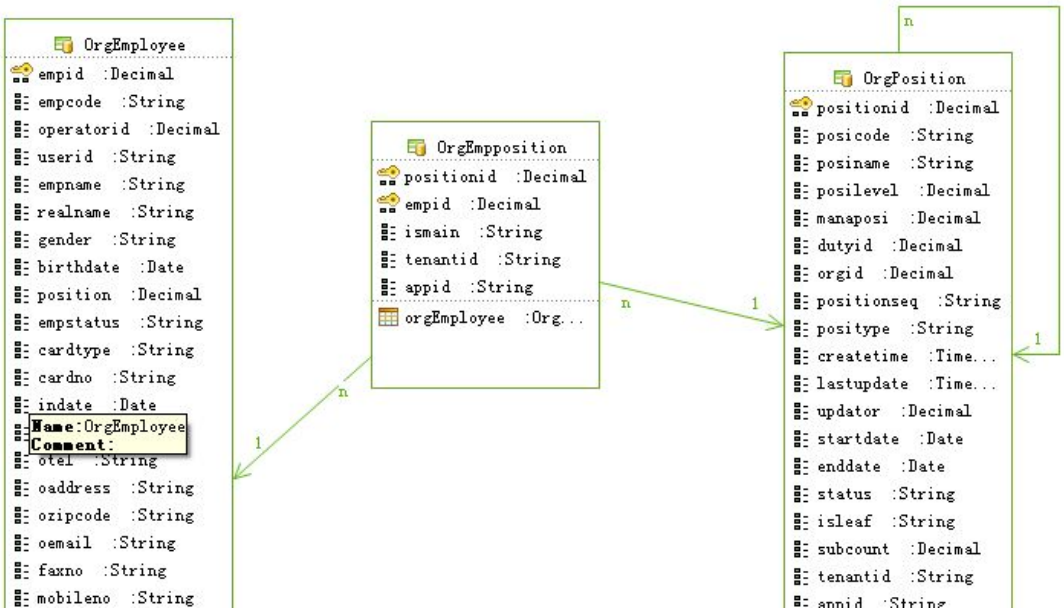


岗位岗位关系在 COFrame 中配置如下：

```
<PartyTypeRef refID="pos_pos_ref" refName="岗位岗位" refType="p_c"
parentPartyTypeID="position" childPartyTypeID="position"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.PosPos
RefDataService"/>
```

2.1.2.6 岗位员工

在 COFrame 中，岗位与员工是多对多的关系，一个岗位可以拥有多个员工，而一个员工也可以隶属于多个岗位。岗位与员工的关联关系如下：



岗位员工的关系通过员工岗位表 ORG_EMPPOSITION 关联

字段	说明	字段类型
POSITIONID	岗位编号	DECIMAL
EMPID	员工编号	DECIMAL
ISMAIN	是否是主机构	CHAR
TENANT_ID	租户ID	VARCHAR
APP_ID	应用ID	VARCHAR

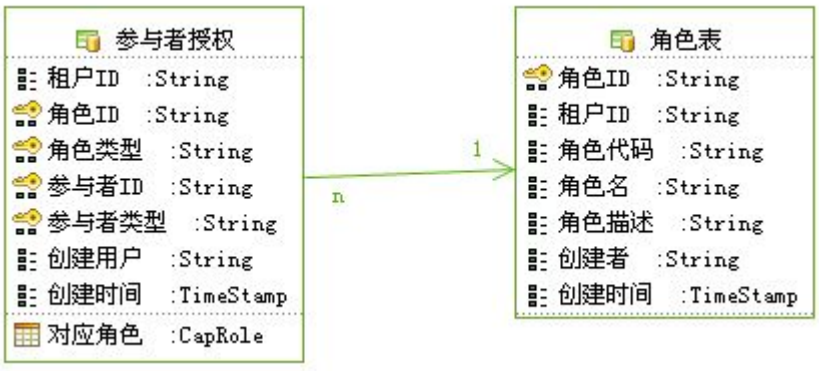
岗位员工关系在 COFrame 中配置如下：

```
<PartyTypeRef refID="pos_emp_ref" refName="岗位员工" refType="p_c"
parentPartyTypeID="position" childPartyTypeID="emp"
partyTypeRefDataService="org.gocom.components.coframe.org.party.ref.impl.PosEmp
RefDataService"/>
```

2.1.3 参与者授权

EOS7 的参与者授权也是一种参与者关系，参与者授权中角色是父而其他参与者是子。COFrame 中的参与者授权也是一种参与者关系，参与者授权包括角色用户、角色员工、角色岗位和角色机构授权。

参与者授权信息存放在数据表 CAP_PARTYAUTH 中，CAP_PARTYAUTH 表与角色的关系如下：



CAP_PARTYAUTH 表内容如下：

字段	说明	
TENANT_ID	租户 ID	VARCHAR
ROLE_ID	角色 ID	VARCHAR
ROLE_TYPE	角色类型	VARCHAR
PARTY_ID	参与者 ID	VARCHAR
PARTY_TYPE	参与者类型	VARCHAR
CREATEUSER	创建用户	VARCHAR
CREATETIME	创建时间	TIMESTAMP

2.1.3.1 角色用户

COFrame 中的角色用户关系是一种授权关系，用来做用户授权，一个用户可以分配多个角色，而一个角色也可以分配给多个用户。角色用户中角色是父，而用户是子。

COFrame 中角色用户关系配置如下：

```
<PartyTypeRef refID="role_user_ref" refName="角色用户" refType="r_p"
parentPartyTypeID="role" childPartyTypeID="user"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.RoleU
serRefDataService"/>
```

2.1.3.2 角色员工

COFrame 中的角色员工关系是一种授权关系，用来做员工授权，一个员工可以分配多个角色，而一个角色也可以分配给多个员工。角色员工中角色是父，而员工是子。

COFrame 中角色员工关系配置如下：

```
<PartyTypeRef refID="role_emp_ref" refName="角色员工" refType="r_p"
parentPartyTypeID="role" childPartyTypeID="emp"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.RoleE
mpRefDataService"/>
```

2.1.3.3 角色岗位

COFrame 中的角色岗位关系是一种授权关系，用来做岗位授权，一个岗位可以分配多个角色，而一个角色也可以分配给多个岗位。角色岗位中角色是父，而岗位是子。

COFrame 中角色岗位关系配置如下：

```
<PartyTypeRef refID="role_position_ref" refName="角色岗位" refType="r_p"
```

```
parentPartyTypeID="role" childPartyTypeID="position"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.RolePositionRefDataService"/>
```

2.1.3.4 角色机构

COFrame 中的角色机构关系是一种授权关系，用来做机构授权，一个机构可以分配多个角色，而一个角色也可以分配给多个机构。角色机构中角色是父，而机构是子。

COFrame 中角色机构关系配置如下：

```
<PartyTypeRef refID="role_org_ref" refName="角色机构" refType="r_p"
parentPartyTypeID="role" childPartyTypeID="org"
partyTypeRefDataService="org.gocom.components.coframe.auth.party.ref.impl.RoleOrgRefDataService"/>
```

2.1.4 参与者与工作流的关系

COFrame 中的参与者与 BPS 中参与者之间有类型的映射，一般情况下，通过参与者映射可直接把 COFrame 中的参与者转换为 BPS 中的参与者。在 COFrame 的 org.gocom.components.coframe.bps.om 构件包中实现了 BPS 的参与者相关参与者接口：com.eos.workflow.omservice.IWFOMService 和 com.eos.workflow.omservice.IWFPermissionService。

在 COFrame 的 org.gocom.components.coframe.bps.om 构件包的配置文件 META-INF/contribution.eosinf 中配置了 COFrame 的参与者与 BPS 参与者的映射。通过该映射关系，BPS 的参与者展现为树状结构。映射关系示例如下：

```
<module name="PartyTypeAdapter">
        <group name="org">
            <configValue key="prefix">0</configValue>
```

```
            <configValue key="code">org</configValue>
            <configValue key="displayName">机构</configValue>
            <configValue key="description">机构</configValue>
            <configValue key="showAtRootArea">true</configValue>
            <configValue key="priority">4</configValue>
            <configValue key="leafParticipant">false</configValue>
            <configValue
key="juniorParticipantTypeCodes">org,position,emp</configValue>
            <configValue key="jointParticipantType">false</configValue>
            <configValue key="jointTypeCodeList"></configValue>
        </group>
        <group name="role">
            <configValue key="prefix">R</configValue>
            <configValue key="code">role</configValue>
            <configValue key="displayName">角色</configValue>
            <configValue key="description">角色</configValue>
            <configValue key="showAtRootArea">true</configValue>
            <configValue key="priority">2</configValue>
            <configValue key="leafParticipant">false</configValue>
            <configValue key="juniorParticipantTypeCodes">emp</configValue>
            <configValue key="jointParticipantType">false</configValue>
            <configValue key="jointTypeCodeList"></configValue>
        </group>
        <group name="emp">
            <configValue key="prefix">E</configValue>
            <configValue key="code">emp</configValue>
            <configValue key="displayName">员工</configValue>
            <configValue key="description">员工</configValue>
```

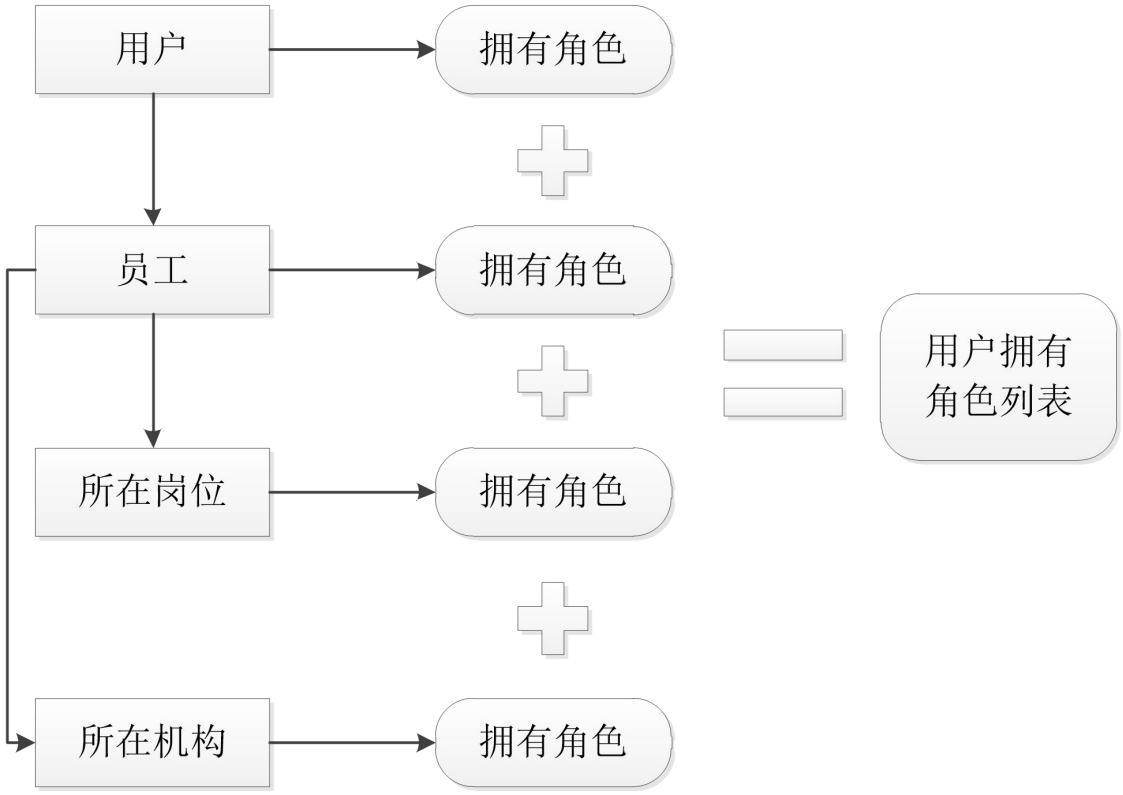
```
            <configValue key="showAtRootArea">false</configValue>
            <configValue key="priority">1</configValue>
            <configValue key="leafParticipant">true</configValue>
            <configValue key="juniorParticipantTypeCodes"></configValue>
            <configValue key="jointParticipantType">false</configValue>
            <configValue key="jointTypeCodeList"></configValue>
        </group>
        <group name="position">
            <configValue key="prefix">P</configValue>
            <configValue key="code">position</configValue>
            <configValue key="displayName">岗位</configValue>
            <configValue key="description">岗位</configValue>
            <configValue key="showAtRootArea">false</configValue>
            <configValue key="priority">1</configValue>
            <configValue key="leafParticipant">fales</configValue>
            <configValue
key="juniorParticipantTypeCodes">position,emp</configValue>
            <configValue key="jointParticipantType">false</configValue>
            <configValue key="jointTypeCodeList"></configValue>
        </group>
    </module>
```

2.2 参与者权限设计

参与者的权限是可以继承的，譬如张三在财务部，所在岗位是财务部经理，所以张三有财务部以及财务部经理的所有权限；财务部经理则拥有财务部的所有权限。

2.2.1 参与者权限计算

一般情况下，参与者拥有的权限是自身的角色以及父参与者的角色集合。COFrame 的参与者权限计算规则如下：



参与者权限计算需要实现参与者权限计算接口，接口定义如下：

```
package org.gocom.components.coframe.auth.service;


import java.util.List;


import com.primeton.cap.party.Party;


/**
```

```
 * TODO 定义授权和party的接口类，根据用户ID获取所有的party及party关联的角色
 *
 * @author caozw (mailto:caozw@primeton.com)
 */
public interface IAuthPartyService {


    /**
     * 获取可继承权限的party的列表,根据该类型展开权限计算层级，譬如用户关联操作员，岗位关联职位，机构无关联
     *
     * @param partyId
     * @return
     */
    List<Party> getAssociateAuthPartyList(String partyId);


    /**
     * 获取直接授权的角色列表
     *
     * @param partyId
     * @return
     */
    List<Party> getAssociateAuthRoles(String partyId);
}
```

参与者权限计算规则需要向权限计算规则管理器中注册，COFrame 中提供配置文件配置参与者权限计算规则，配置文件位于 org.gocom.components.coframe.auth 构件包的 META-INF/contribution.eosinf。配置文件示例如下：

```
<!-- party管理服务配置 -->
```

```
    <module name="PartyRoleAuthService">
        <group name="user">
            <configValue
key="service">org.gocom.components.coframe.auth.service.impl.UserAuthPartyServi
ce</configValue>
        </group>
        <group name="emp">
            <configValue
key="service">org.gocom.components.coframe.auth.service.impl.EmpAuthPartyServic
e</configValue>
        </group>
        <group name="position">
            <configValue
key="service">org.gocom.components.coframe.auth.service.impl.PositionAuthPartyS
ervice</configValue>
        </group>
    </module>
```

2.2.2 用户登录

用户登录时有两个环节：一是登录密码校验，二是用户权限信息初始化。

2.2.2.1 登录密码校验

用户登录时与用户表保存的用户名和密码信息比对，登录用户表与其他表没有直接的关联，仅提供登录功能。

用户表 CAP_USER 的字段定义如下：

字段	说明	字段类型

OPERATOR_ID	操作员 ID	DECIMAL
TENANT_ID	租户 ID	VARCHAR
USER_ID	登录用户名	VARCHAR
PASSWORD	密码	VARCHAR
INVALDATE	密码失效日期	DATE
USER_NAME	用户名称	VARCHAR
AUTHMODE	本地密码认证、LDAP 认证、等	VARCHAR
STATUS	正常，挂起，注销，锁定...	VARCHAR
UNLOCKTIME	当状态为锁定时，解锁的时间	TIMESTAMP
MENUTYPE	用户登录后菜单的风格	VARCHAR
LASTLOGIN	最后登录时间	TIMESTAMP
ERRCOUNT	密码错误次数	DECIMAL
STARTDATE	有效开始时间	DATE
ENDDATE	有效截止时间	DATE
VALIDTIME	定义一个规则表达式，表示允许操作的有效时间范围	VARCHAR
MACCODE	允许设置多个 MAC	VARCHAR
IPADDRESS	允许设置多个 IP 地址	VARCHAR
EMAIL	邮箱地址	VARCHAR
CREATEUSER	创建人	VARCHAR
CREATETIME	创建时间	TIMESTAMP

2.2.2.2 用户权限信息初始化

用户登录后，根据用户名，需要获取用户的基本信息以及用户的权限信息，并存放到 session 中。EOS 采用 UserObject 对象存放用户信息以及权限信息。

EOS 定义了用户初始化的接口，在用户登录时、或 SSO 登录时会调用该接口初始化用户信息。用户初始化的接口定义如下：

```
package com.primeton.cap.party;


import com.eos.data.datacontext.IUserObject;


/**
 * 初始化参与者的接口
 *
 *
 * @author caozw (mailto:caozw@primeton.com)
 */
public interface IPartyUserInitService {
    /**
     * 初始化用户
     * 1.初始化登录信息
     * 2.初始化机构信息
     * 3.初始化角色信息
     * 4.初始化其他所需要的信息
     * @param userId
     * @param tenantId
     * @return
     */
    public IUserObject initUserObject(String userId, String tenantId);
}
```

用户实现初始化接口后，需要在应用启动时调用 PartyManagerServiceLoader 的 setCurrentPartyUserInitService 方法设置用户初始化接口。

COFrame 的 org.gocom.components.coframe.init.CoframePartyUserInitService.java 类实现了 IPartyUserInitService 接口，实现了基于 COFrame 的用户初始化。

CoframePartyUserInitService 中用户初始化的说明：

UserObject 属性	存放内容说明
userId	【**员工**】参与者的 ID
userName	【**用户**】参与者的名称（登录名）
userMail	【**员工**】参与者的邮箱
userRealName	【**员工**】参与者的姓名
userOrgName	员工所在主机构的【**机构**】参与者的名称
userOrgId	员工所在主机构的【**机构**】参与者的 ID
TEN_ANT_ID	租户 ID
EXTEND_USER_ID	【**用户**】参与者的 ID
roleList	用户拥有的【**角色**】参与者 ID 的列表，以”,”隔开

2.3 资源授权设计

COFrame 中的资源授权分为两个部分：一是资源注册，受管控的资源应当向资源管理器注册，注册的资源由 EOS 统一缓存；二是资源授权，授权的资源持久化保存在数据表 CAP_RESAUTH 中。

数据库表 CAP_RESAUTH 的结构如下：

字段	**说明**	**数据类型**
PARTY_ID	参与者 ID	VARCHAR2
PARTY_TYPE	参与者类型	VARCHAR2
RES_ID	资源 ID	VARCHAR2
RES_TYPE	资源类型	VARCHAR2
TENANT_ID	租户 ID	VARCHAR2
RES_STATE	资源状态	VARCHAR2
PARTY_SCOPE	参与者范围	CHAR

CREATEUSER	创建用户	VARCHAR2
CREATETIME	创建时间	TIMESTAMP

2.3.1 资源注册

需要授权的资源应当有全局唯一的名称空间，通过该名称空间可以确定资源的唯一性。资源加载时应当调用 ResourceRuntimeManager.getInstance().registerManagedResource 方法注册资源，资源更新时应当调用 ResourceRuntimeManager.getInstance().updateRegisteredManagedResource 方法更新资源，资源被删除时应当调用 ResourceRuntimeManager.getInstance().unRegisterManagedResource 方法注销资源。

2.3.2 资源授权

资源授权是设置资源对某一角色的状态，如某个资源是否可见、不可见、是否可编辑等。资源授权主要调用 com.primeton.cap.auth.manager.AuthRuntimeManager 的方法，常用方法如下：

方法	说明
getAuthResListByRole	获取某个角色的资源权限
getAuthResListWithChildrenByRole	获取某个角色的某个资源及其子资源的权限
getAuthResListByRole	获取某种类型的资源权限
getAuthResourceState	获取某个角色的资源状态
addOrUpdateAuthRes	给某个角色赋予某个资源权限
addOrUpdateAuthResBatch	给某个角色批量赋予资源权限
authResBatch	将一批资源授权给某个角色
delAuthRes	删除某个角色对某个资源的权限
delAuthResBatch	批量删除某个角色对某个资源的权限
getCurrentPartyResAuthState	获取当前 party 对于资源的权限状态，由于

	party 可能存在多个角色，所以可能会有多个状态，返回后由用户决定如何取舍
getCurrentPartyResAuthWithChildren	获取当前 party 对于资源及其子资源的权限状态，返回结果中不再有资源父子关系，都是平铺的方式

3 应用功能菜单设计

3.1 应用功能管理

应用功能维护了应用功能等新增、修改、删除等操作。这些操作涉及到应用、功能组、功能、功能资源等几个对象。功能被分成了不同的功能组并且归属于一个应用，功能下面可以包含多个相关资源，如页面流、JSP、表单、视图等。

应用功能菜单的数据库结构如下：

APP_APPLICATION 的结构如下：

字段	说明	字段类型
APPID	应用程序编号	DECIMAL
APPCODE	应用程序代码	VARCHAR
APPNAME	应用程序名称	VARCHAR
APPTYPE	应用程序类型	VARCHAR
ISOPEN	是否开通	CHAR
OPENDATE	开通日期	DATE
URL	应用上下文	VARCHAR
APPDESC	应用程序描述	VARCHAR
MAINTENANCE	maintenance	DECIMAL
MANAROLE	manarole	VARCHAR
DEMO	demo	VARCHAR
INIWP	iniwp	CHAR
INTASKCENTER	intaskcenter	CHAR
IPADDR	应用程序 IP	VARCHAR
IPPORT	应用程序端口	VARCHAR
APP_ID	应用信息	VARCHAR
TENANT_ID	租户信息	VARCHAR
protocol_type	协议类型	varchar

APP_FUNCGROUP 的表结构如下：

字段	说明	字段类型
FUNCGROUPID	功能组编号	DECIMAL
APPID	应用程序编号	DECIMAL

FUNCGROUPNAME	功能组名称	VARCHAR
PARENTGROUP	父功能组	DECIMAL
GROUPLEVEL	功能组层次	INT
FUNCGROUPSEQ	功能组序号	VARCHAR
ISLEAF	是否为叶子	CHAR
SUBCOUNT	subcount	DECIMAL
APP_ID	应用信息	VARCHAR
TENANT_ID	租户信息	VARCHAR

APP_FUNCTION

字段	说明	字段类型
FUNCCODE	功能编码	VARCHAR
FUNCGROUPID	功能组编号	DECIMAL
FUNCNAME	功能名称	VARCHAR
FUNCDESC	功能描述	VARCHAR
FUNCACTION	功能 URL	VARCHAR
PARAINFO	功能参数信息	VARCHAR
ISCHECK	ischeck	CHAR
FUNCTYPE	功能类型	VARCHAR
ISMENU	是否菜单	CHAR
APP_ID	应用信息	VARCHAR
TENANT_ID	租户信息	VARCHAR

APP_FUNCRESOURCE

字段	说明	字段类型
RESID	资源编号	DECIMAL
FUNCCODE	功能编号	VARCHAR
RESTYPE	资源类型	VARCHAR

RESPATH	资源路径	VARCHAR
COMPACKNAME	所属构件包	VARCHAR
RESNAME	资源名称	VARCHAR
APP_ID	应用信息	VARCHAR
TENANT_ID	租户信息	VARCHAR

3.2 菜单管理

菜单是系统中作为用户功能访问入口显示信息，COFrame 中菜单只是作为功能的排列显示，菜单本身不提供访问的 URL，访问或者点击菜单是通过与功能的关联来间接访问功能的。

APP_MENU 的数据库表结构如下：

字段	说明	字段类型
MENUID	菜单编号	VARCHAR
MENUNAME	菜单名称	VARCHAR
MENULABEL	菜单显示名称	VARCHAR
MENUCODE	菜单代码	VARCHAR
ISLEAF	是否叶子菜单	CHAR
MENUACTION	菜单 url	VARCHAR
PARAMETER	菜单参数	VARCHAR
UIENTRY	uientry	VARCHAR
MENULEVEL	菜单层次	SMALLINT
ROOTID	rootid	VARCHAR
PARENTSID	父菜单 id	VARCHAR
DISPLAYORDER	显示顺序	SMALLINT
IMAGEPATH	菜单闭合图片路径	VARCHAR
EXPANDPATH	菜单展开图片路径	VARCHAR
MENUSEQ	菜单序号	VARCHAR
OPENMODE	打开方式	VARCHAR

SUBCOUNT	子菜单数	DECIMAL
APPID	应用程序编号	DECIMAL
FUNCCODE	功能代码	VARCHAR
APP_ID	应用信息	VARCHAR
TENANT_ID	租户信息	VARCHAR