**MAXWEB**

**概要设计说明书**

**MAXWEB概要设计说明书**

更改记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 日期 | 修改章节 | 修改类型 | 修改描述 | 修改人 | 版本 |
| 2013年05月10日 |  | A | 创建文档 | 金灿敏 | 0.1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

\*修改类型分为**A** - ADDED **M** - MODIFIED **D** – DELETED

文档编号：

|  |  |  |
| --- | --- | --- |
| 项目名称： | 拟制： | 审核： |
| SQA： | 批准： |
| 项目代号： | 收文： | |
| 产品版本： | 抄送： | |

**目录**

[1. 引言 5](#_Toc338339526)

[1.1. 目的 5](#_Toc338339527)

[1.2. 背景 5](#_Toc338339528)

[1.3. 目标读者 5](#_Toc338339529)

[1.4. 术语和缩略语 5](#_Toc338339530)

[2. 设计概述 6](#_Toc338339531)

[2.1. 系统描述 6](#_Toc338339532)

[2.2. 设计约束 6](#_Toc338339533)

[2.3. 设计策略 6](#_Toc338339534)

[3. 总体设计 7](#_Toc338339535)

[3.1. 物理结构图 7](#_Toc338339536)

[3.2. 功能结构图 7](#_Toc338339537)

[3.3. 顶层数据流图 9](#_Toc338339538)

[3.4. 程序运行流程图 10](#_Toc338339539)

[3.5. 运行环境 10](#_Toc338339540)

[4. 数据结构设计 11](#_Toc338339541)

[5. 功能模块设计 11](#_Toc338339542)

[5.1. 客户管理 11](#_Toc338339543)

[5.1.1. 查看客户列表 11](#_Toc338339544)

[6. 系统出错处理设计 12](#_Toc338339545)

[6.1. 出错信息设计 12](#_Toc338339546)

[6.2. 系统维护设计 13](#_Toc338339547)

**图形目录**

**未找到图形项目表。**

# 介绍

## 目的

该文档为MAX项目中WEB端概要设计说明书，根据需求规格说明书，描述系统中WEB端的实现逻辑及关键交互。

## 背景

MAX为基于云会议2.0开发的支持大方数会议产品，云会议2.0上线后，越来越多的大客户开始使用，由于云会议2.0无法满足大客户需求，因此公司针对网络会议产品，推出云会议MAX产品，以满足不断增加的大客户需求。

## 目标读者

包括但不限于下列人员：

MAX的产品设计人员

MAX项目管理人员

MAX软件开发人员

MAX系统测试人员

MAX质量管理人员

## 术语和缩略语

|  |  |
| --- | --- |
| **术语** | **说明** |
| MAX | 网络视频会议产品项目代号，包括相关的所有系统 |
| MAX Client | 网络视频会议产品客户端 |
| MAX WEB | 网络视频会议产品子系统，提供用户浏览及加入会议的WEB站点 |
| MAXUniform | 网络视频会议产品子系统，为MAXClient、MAXWEB提供统一接口 |
| PC2.0 | 网络视频会议产品项目代号，包括相关的所有系统 |
| CMS | 会议管理，TANG平台的一个模板 |
| UMS | 用户管理，Tang平台的一个模块 |
| MIS | 信息管理系统，负责站点和账号的维护 |
| LAMP | 由Linux+Apache+Mysql+PHP组成的WEB应用程序平台 |
| RESTful | REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。 |
| CURL | CURL是利用URL语法在命令行方式下工作的文件传输工具。 |

# 设计概述

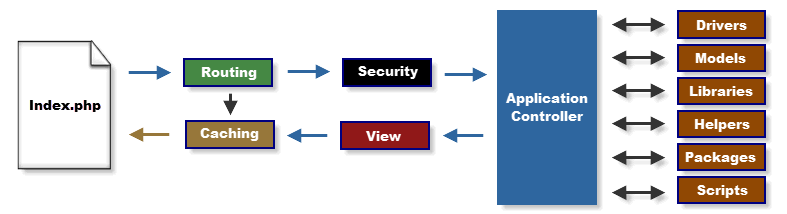
## 系统描述

MAXWEB端将基于LAMP(Linux+Apache+Mysql+PHP)平台开发并实现，引入轻量级（PHP）开发框架CI，表现层采用浏览器加载Html+JavaScript+CSS来展现产品界面。

MAXWEB端由两部分组成：MAXWEB+MAXUniform，WEB主要提供前端展示及后台管理；Uniform主要提供会议管理功能、通讯录管理功能以及各功能对外开放的接口。

## 设计约束

## 设计策略



1. index.php 作为前端控制器，初始化运行 CodeIgniter 所需要的基本资源。
2. Router 检查 HTTP 请求，以确定谁来处理请求。
3. 如果缓存(Cache)文件存在，它将绕过通常的系统执行顺序，被直接发送给浏览器。
4. 安全(Security)。应用程序控制器(Application Controller)装载之前，HTTP 请求和任何用户提交的数据将被过滤。
5. 控制器(Controller)装载模型、核心库、辅助函数，以及任何处理特定请求所需的其它资源。
6. 最终视图(View)渲染发送到 Web 浏览器中的内容。如果开启缓存(Caching)，视图首先被缓存，所以将可用于以后的请求。

## 运行环境

Linux+Apache+Mysql+PHP

## 尚未解决的问题

# 结构设计

## 系统结构图



# 功能模块设计

## 用户登录

与PC2.0预约型产品一致

## 帐号管理

与MIS关联，目前MIS还在需求分析阶段

## 会议管理

### 查询历史会议

|  |  |
| --- | --- |
| **描述** | 获取90天内自己安排的前10场会议 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-001 |
| **访问链接** | [https://maxuniform.quanshi.com/arrangeconf/getHistoryEvents](http://max.quanshi.com/pcclient/getHistoryEvents) |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID，在session中获取，无需传 | | $request\_data | 请求数据，包含以下字段：  keyword：搜索关键字；  limit：查询记录条数 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 若搜索到结果，则返回相应格式的结果  没有结果，则返回空 | | content\_type | 返回结果的数据类型(json/xml) | |
| **依赖外部系统** | N/A |
| **逻辑描述** | 查询历史会议.jpg   1. MAXClient向服务器端发出获取历史会议的请求 2. 服务器根据请求链接，指定由pc客户端接口控制器arrangeconf处理请求 3. 控制器获取POST参数，放入request\_data数组 4. 控制器对GET参数进行安全过滤检查 5. 控制器调用pc客户端接口逻辑类arrangeconf\_manger中的获取历史会议函数getHistoryEvents()   $this ->load->library(‘arrangeconf\_manger);  $historylist = $this->arrangeconf\_manger->getHistoryEvents ();   1. 逻辑类中getHistoryEvents ()函数调用安排会议数据模型类中的获取历史会议函数selectHistoryList()   $this ->load->model(‘arrangeconf\_model’);  $ historylist = $this-> arrangeconf\_model’->selectHistoryList ();   1. 数据模型类返回历史会议数据给逻辑类，逻辑类返回客户列表给控制器 2. 控制器分析逻辑类返回结果，指定格式返回给MAXClient端响应   $this->responseV(8001,$result,'text/plain'); |
| **备注** | userId不为空， keyword为空时有两处调用：   1. 判断客户端是否显示历史会议的选项； 2. 当搜索不到数据时，点击历史会议选项，搜索limit条数据   userId不为空，keyword不为空，若根据keyword搜索到数据则显示数据，若没搜索到，则  返回空数据。 |

### 导入参会人

|  |  |
| --- | --- |
| **描述** | 解析并检测客户端上传的参会人文件，将数据插入到数据库 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-002 |
| **访问链接** | https://maxuniform.quanshi.com/arrangeconf/uploadParty |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID，在session中获取，无需传 | | $request\_data | 请求数据，包含以下字段：  filename：上传文件的对象标识； | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 导入成功：1  导入失败：   1. 数据格式有误，result中加fileurl项，将生成文件的url地址返回 2. 其他错误，result为0 | | content\_type | 返回结果的数据类型(json/xml) | |
| **依赖外部系统** | N/A |
| **逻辑描述** | 导入参会人.jpg   1. MAXClient向服务器端发出导入参会人的请求 2. 服务器根据请求链接，指定由pc客户端接口控制器arrangeconf处理请求 3. 控制器获取POST参数，放入request\_data数组 4. 控制器对参数进行安全过滤检查 5. 控制器调用pc客户端接口逻辑类arrangeconf\_manger中的导入参会人的函数upload\_party()   $this ->load->library(‘arrangeconf\_manger);  $rs = $this->arrangeconf\_manger->upload\_party ();   1. 逻辑类中的upload\_party函数判断文件类型是否正确   判断文件大小是否超限  若不符合规则，返回失败给客户端  $this->responseV(8001,$result,'text/plain');   1. 逻辑类中的upload\_party函数调用phpexcel解析类解析上传的文件，根据解析的结果   判断记录条数是否超限  判断文件表头是否正确  判断每条记录是否至少存在邮箱或手机  上述判断若有不符合规则的情况，返回失败给客户端  $this->responseV(8001,$result,'text/plain');  判断文件字段是否合法，若字段不合法，给该条记录加对应不合法标识  判断是否有重复，若有重复给该条记录加对应重复标识   1. 逻辑类中upload\_party ()函数调用安排会议数据模型类中的删除参会人函数delete\_tmp\_party()   再调用增加参会人到临时表(max\_tmp\_party)函数add\_tmp\_party()  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’->add\_tmp\_party ();   1. 数据模型类返回添加结果给逻辑类，   若字段有不合法或重复的情况，则查询max\_tmp\_party表调用phpexcel类生成带错误标识的更改文件，向客户端返回文件url提供下载。  若没有字段不合法或重复的情况，则返回成功给控制器   1. 控制器分析逻辑类返回结果，指定格式返回给MAXClient端响应   $this->responseV(8001,$result,'text/plain'); |
| **备注** | POST参数：  userId：用户id  filename:上传文件的标识对象 |

### 取消导入参会人

|  |  |
| --- | --- |
| **描述** | 客户端点击取消清空本地参会人上传临时表 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-003 |
| **访问链接** | https://maxuniform.quanshi.com/arrangeconf/cancelUploadParty |
| **请求方式** | GET |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID，在session中获取，无需传 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 0：取消失败  1：取消成功 | | content\_type | 返回结果的数据类型(json/xml) | |
| **依赖外部系统** | N/A |
| **逻辑描述** | 取消导入参会人.jpg   1. MAXClient向服务器端发出导入参会人的请求 2. 服务器根据请求链接，指定由max客户端接口控制器arrangeconf处理请求 3. 控制器获取POST参数，放入request\_data数组 4. 控制器对GET参数进行安全过滤检查 5. 控制器调用max客户端接口逻辑类arrangeconf\_manger中的取消导入参会人   的函数cancel\_upload\_party()  $this ->load->library(‘arrangeconf\_manger);  $rs = $this->arrangeconf\_manger-> cancel\_upload\_party ();   1. 逻辑类中的cancel\_upload\_party函数调用unlink删除服务器上的文件   先判断是否存在文件，若存在则删除   1. 数据模型类返回结果给逻辑类，逻辑类根据删除文件结果返回给控制器 2. 控制器分析逻辑类返回结果，指定格式返回给MAXClient端响应   $this->responseV(8001,$result,'text/plain'); |
| **备注** | POST参数：  userId：用户id |

### 获取参会人信息

|  |  |
| --- | --- |
| **描述** | 根据用户userId获取参会人信息 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-004 |
| **访问链接** | https://maxuniform.quanshi.com/arrangeconf/getPartyInfo |
| **请求方式** | GET |
| **数据类型** | 无 |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID，在session中获取，无需传 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 若搜索到结果，则返回相应格式的结果  没有结果，则返回空 | | content\_type | 返回结果的数据类型(json/xml) | |
| **依赖外部系统** | N/A |
| **逻辑描述** | 获取参会人信息.jpg   1. MAXClient向服务器端发出获取邀请参会人信息的请求 2. 服务器根据请求链接，指定由pc客户端接口控制器arrangeconf处理请求 3. 控制器获取POST参数，放入request\_data数组 4. 控制器对参数进行安全过滤检查 5. 控制器调用max客户端接口逻辑类arrangeconf\_manger中的获取参会人信息   的函数getPartyInfo()  $this ->load->library(‘arrangeconf\_manger);  $rs = $this->arrangeconf\_manger->getPartyInfo ();   1. 逻辑类中的getPartyInfo函数调用安排会议数据模型类中的根据userid获取参会人信息   函数getPartyInfo ($userId)  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’-> getPartyInfo ($userId);   1. 数据模型类返回结果给逻辑类，逻辑类整理结果返回给控制器 2. 控制器分析逻辑类返回结果，指定格式返回给MAXClient端响应   $this->responseV(8001,$result,'text/plain'); |
| **备注** | POST参数：  userId：用户id |

### 搜索通讯录

本地化搜索

### 查询会议信息

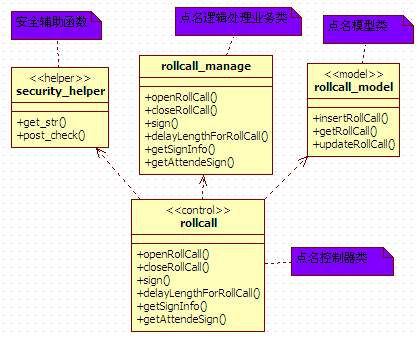
|  |  |
| --- | --- |
| **描述** | 根据预约会议id查询会议信息 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-007 |
| **访问链接** | https://maxuniform.quanshi.com/common/getConferenceBaseInfoById/appId/conferenceId |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **依赖外部系统** | N/A |
| **逻辑描述** | 在原pc2.0接口基础上增加返回字段，其余逻辑不变  增加的字段：  密码类型，会议设置，通知方式，参会人总数 |
| **备注** | GET参数：  appId：产品id  conferenceId：预约会议id |

### 创建预约会议

|  |  |
| --- | --- |
| **描述** | 创建预约会议并发送邀约 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-008 |
| **访问链接** | https://maxuniform.quanshi.com/common/reservConference/appId |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID，在session中获取，无需传 | | $request\_data | 请求数据，包含以下字段：  title：会议主题  startTime：开始时间  Length：时长  Timezone：时区  Baseinfo：参会人信息  allow\_show:显示参会人列表  allow\_remove\_slience:是否允许自我解除静音  allow\_share\_vidwo:是否允许共享视频  is\_one\_meet:是否针对本场会议  password\_type:密码类型  send\_type:发送方式 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 0：创建失败  1：创建成功 | | content\_type | 返回结果的数据类型(json/xml) | |
| **依赖外部系统** | cms,mss,短信平台 |
| **逻辑描述** | 创建预约会议.jpg   1. MAXClient向服务器端发出创建预约会议的请求 2. 服务器根据请求链接，指定由公共接口控制器common处理请求 3. 控制器获取POST参数，放入request\_data数组 4. 控制器对参数进行安全过滤检查 5. 获得传入值的开始时间，会议标题   获取会中提示语言  获取邮件发送语言  对会议length判断，小于24小时，返回失败通知客户端  $this->responseV(8001,$result,'text/plain');  获取提前入会时间  修改开始时间的格式   1. 控制器调用公共接口逻辑类m\_common中的创建预约会议函数   的函数reservConference ()  $this ->load->library(‘m\_common);  $rs = $this->arrangeconf\_manger-> reservConference ();   1. 逻辑类中的reservConference函数进行提示语的语言转化   根据开始时间进行会议冲突判断  根据账号信息拼接请求cms创建预约会议的链接  创建会议成功   1. 逻辑类保存会议设置：会议主题、会议开始时间、会议时长、发送邀约方式、密码类型   保存到表max\_confinfo  $r = $this->CI->uniform\_roles->insertInfo('max\_confinfo',$datas);  保存到表max\_user\_conference  $r = $this->CI->uniform\_roles->insertInfo('max\_confinfo\_mail',$datas);   1. 逻辑类保存参会人设置   针对所有会议：将待插入设置表（max\_party\_set）的conferenceId置为0，同时删除设置表表此用户对其他会议的设置  只针对此次会议：删除设置表中此用户conferenId为0的设置  调用安排会议数据模型类中的保存参会人设置的  函数save\_party\_set ()，改函数将先检测所传插入数组中的conferenceId是否已经存在，若存在，则先删除，再执行插入  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’-> save\_party\_set ($userId);   1. 逻辑类调用发送邮件逻辑类m\_mss，循环email数组插入到mss数据库中   $this ->load->library(‘m\_mss);  $rs = $this->m\_mss-> send\_email ();  m\_mss逻辑类将传来的数据插入到mss库的sme\_mail\_sending\_task表中，mss邮件系统会每次从sme\_mail\_sending\_task表中取150条，每秒发4条  调用发送短信逻辑类m\_sms,循环手机号发送短信  $this ->load->library(‘m\_sms);  $rs = $this->m\_sms-> send\_sms ();  m\_sms逻辑类将传来的手机号码和短信内容组成xml，用curl发送给短信平台地址  将email，手机号，姓名，预约会议id组成待插入数组  调用数据模型类arrangeconf\_model中增加会议参会人信息的函数add\_party()，插入到表  max\_party  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’-> add\_party ($userId);   1. 数据模型类返回结果给逻辑类，逻辑类整理结果返回给控制器 2. 控制器分析逻辑类返回结果，指定格式返回给MAXClient端响应   $this->responseV(8001,$result,'text/plain'); |
| **备注** | POST参数：  title：会议主题  startTime：开始时间  Length：时长  Timezone：时区  Baseinfo：参会人信息 |

### 编辑预约会议

|  |  |
| --- | --- |
| **描述** | 变更预约会议并发送邀约 |
| **HLD ID** | MAXUNIFORM-HLD-ARRANGECONFERENCE-009 |
| **访问链接** | https://maxuniform.quanshi.com/common/reservConference/appId |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID，在session中获取，无需传 | | $request\_data | 请求数据，包含以下字段：  title：会议主题  startTime：开始时间  Length：时长  Timezone：时区  Baseinfo：参会人信息  conferenceId：预约会议id  allow\_show:显示参会人列表  allow\_remove\_slience:是否允许自我解除静音  allow\_share\_vidwo:是否允许共享视频  is\_one\_meet:是否针对本场会议  password\_type:密码类型  send\_type:发送方式  edittype:重开或编辑会议 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 0：创建失败  1：创建成功 | | content\_type | 返回结果的数据类型(json/xml) | |
| **依赖外部系统** | cms,mss,短信平台 |
| **逻辑描述** | 编辑、重开预约会议.jpg   1. MAXClient向服务器端发出创建预约会议的请求 2. 服务器根据请求链接，指定由公共接口控制器common处理请求 3. 控制器获取POST参数，放入request\_data数组 4. 控制器对参数进行安全过滤检查 5. 获得传入值的开始时间，会议标题   获取会中提示语言  获取邮件发送语言  对会议length判断，小于24小时，返回失败通知客户端  $this->responseV(8001,$result,'text/plain');  获取提前入会时间  修改开始时间的格式   1. 控制器调用公共接口逻辑类m\_common中的变更预约会议函数   的函数updatereservConference ()  $this ->load->library(‘m\_common);  $rs = $this->arrangeconf\_manger-> updatereservConference ();   1. 逻辑类中的updatereservConference函数进行提示语的语言转化   根据开始时间进行冲突判断，排除掉自身  要变更会议的结束时间小于当前时间  变更的时间和原来的时间不等时要判断变更后的开始时间是否小于当前时间  获取以前会议的信息拼接请求cms变更预约会议的链接  $this->CI->cms->modifyConf()  变更会议成功   1. 逻辑类更新会议设置：会议主题、会议开始时间、会议时长、发送邀约方式、密码类型   更新到表max\_confinfo  $r = $this->CI->uniform\_roles->updateInfo('max\_confinfo',$datas);  保存到表max\_user\_conference  $r = $this->CI->uniform\_roles->updateInfo('max\_confinfo\_mail',$datas);   1. 逻辑类保存参会人设置   针对所有会议：将待插入设置表（max\_party\_set）的conferenceId置为0，同时删除设置表表此用户对其他会议的设置  只针对此次会议：删除设置表中此用户conferenId为0的设置  调用安排会议数据模型类中的保存参会人设置的  函数save\_party\_set ()，改函数将先检测所传插入数组中的conferenceId是否已经存在，若存在，则先删除，再执行插入  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’-> save\_party\_set ($userId);   1. 逻辑类获取以前会议的邀约信息，结合现在的邀约参会人，将邀约参会人分成3个   数组，取消会议组（发取消邮件和短信），新增会议组（发新增邮件和短信），编辑会议组（发编辑邮件和短信），当重新召开会议时，全部都为新增会议组。   1. 逻辑类调用发送邮件逻辑类m\_mss，分别循环3个数组，将email插入到mss数据库中   $this ->load->library(‘m\_mss);  $rs = $this->m\_mss-> send\_email ();  m\_mss逻辑类将传来的数据插入到mss库的sme\_mail\_sending\_task表中，mss邮件系统会每次从sme\_mail\_sending\_task表中取150条，每秒发4条  调用发送短信逻辑类m\_sms,分别循环3个数组，给手机号发送短信  $this ->load->library(‘m\_sms);  $rs = $this->m\_sms-> send\_sms ();  m\_sms逻辑类将传来的手机号码和短信内容组成xml，用curl发送给短信平台地址  将email，手机号，姓名，预约会议id组成待插入数组  调用数据模型类arrangeconf\_model中删除会议参会人信息delete\_party(),再增加会议参会人信息的函数add\_party()，插入到表max\_party  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’-> delete\_party ($userId);  $ rs = $this-> arrangeconf\_model’-> add\_party ($userId);  调用数据模型类arrangeconf\_model中删除表max\_user\_conference中的信息,再增加会议参会人信息的函数add\_user\_conferce()  $this ->load->model(‘arrangeconf\_model’);  $ rs = $this-> arrangeconf\_model’-> delete\_user\_conference ($userId);  $ rs = $this-> arrangeconf\_model’-> add\_user\_conference ($userId);   1. 数据模型类返回结果给逻辑类，逻辑类整理结果返回给控制器 2. 控制器分析逻辑类返回结果，指定格式返回给MAXClient端响应 |
| **备注** | POST参数：  title：会议主题  conferenceId：预约会议id  startTime：开始时间  Length：时长  Timezone：时区  Baseinfo：参会人信息 |



点名模块类图 1

### **开启点名**

|  |  |
| --- | --- |
| **描述** | 开启点名 |
| **HLD ID** | MAX-HLD-ROLLCALL-001 |
| **访问链接** | https://maxuniform.quanshi.com/client/openRollCall |
| **请求方式** | post |
| **参数列表** | |  |  | | --- | --- | | 参数 | 说明 | | $tempConferenceId | 临时会议id | | $length | 点名持续时长，单位为秒 | |
| **返回结果** | |  |  | | --- | --- | | 参数 | 说明 | | response | {"result":"开启点名时间戳","status":"成功:0失败:50001"} | | content-type | 返回数据类型(application/json) | |
| **依赖外部系统** | N/A |
| 逻辑描述 | 1. 客户端通过ajax向web服务器maxuniform发送开启点名请求。 2. 服务器根据请求链接，指定点名模块控制器(即rollcall)里的openRollCall()方法处理请求。 3. Rollcall控制器调用点名业务逻辑类(即rollcall\_manager)里的openRollCall()方法来处理开启点名逻辑。   $this->load->library('rollcall\_manager');  $this->rollcall\_manager->openRollCall();   1. rollcall\_manager调用点名数据库模型(即rollcall\_model)里的save()方法生成一条点名记录。   $this->CI->load->model('rollcall\_model');  $this->CI->rollcall\_model->save();   1. rollcall\_model将插入结果返回给业务逻辑类。 2. 业务逻辑类将开启点名时间戳返回给点名控制器。 3. 点名控制器将结果返回给客户端。 |
| **备注** | 见数据库结构5.2.5点名时间表 |

### **延迟时长点名**

|  |  |
| --- | --- |
| **描述** | 延迟时长点名 |
| **HLD ID** | MAX-HLD-ROLLCALL-002 |
| **访问链接** | https://maxuniform.quanshi.com/client/openRollCall |
| **请求方式** | post |
| **参数列表** | |  |  | | --- | --- | | 参数 | 说明 | | $openRollcallTime | 开始点名时间戳 | | $tempConferenceId | 临时会议id | | $length | 延长时长,单位为秒 | |
| **返回结果** | |  |  | | --- | --- | | 参数 | 说明 | | response | {"result":"当前时间戳","status":"成功:0失败:50001"} | | content-type | 返回数据类型(application/json) | |
| 逻辑描述 | 1. 客户端通过ajax向web服务器maxuniform发出延长时长点名接口请求。 2. 服务器根据请求链接，指定点名控制器(即rollcall)里的delayLengthForRollCall()方法处理延迟时长点名请求。 3. 点名控制器调用点名业务逻辑类(即rollcall\_manager)类里的delayLengthForRollCall()方法来实现延长时长点名的逻辑。   $this->load->library('rollcall\_manager');  $this->rollcall\_manager->delayLengthForRollCall();   1. 点名业务逻辑类调用点名数据库模型(即rollcall\_model)里的update ()方法来更新点名时长。   $this->CI->load->model('rollcall\_model');  $this->CI->rollcall\_model->update();   1. 点名数据库模型类将更新结果返回给点名业务逻辑类。 2. 点名业务逻辑类根据结果返回相应的值给点名控制器。 3. 点名控制器将结果返回给客户端。 |
| **备注** | 见数据库结构5.2.5点名时间表 |

### **结束点名**

|  |  |
| --- | --- |
| **描述** | 结束点名 |
| **HLD ID** | MAX-HLD-ROLLCALL-03 |
| **访问链接** | https://maxuniform.quanshi.com/client/closeRollCall |
| **请求方式** | post |
| **参数列表** | |  |  | | --- | --- | | 参数 | 说明 | | $openRollcallTime | 开始点名时间戳 | | $tempConferenceId | 临时会议id | | $flag | 标记(end:结束点名，cancel：取消点名) | |
| **返回结果** | |  |  | | --- | --- | | 参数 | 说明 | | response | {"result":"已签到参会人信息","status":"成功:0失败:50001"} | | content-type | 返回数据类型(application/json) | |
| 逻辑描述 | 1. 客户端通过ajax向web服务器maxuniform发出结束/取消点名请求。 2. 服务器根据请求链接，指定点名控制器(即rollcall)里的closeRollCall()方法处理结束/取消点名请求。 3. 点名控制器调用点名业务逻辑类(即rollcall\_manager)里的closeRollCall ()方法来处理结束点名逻辑。   $this->load->library('rollcall\_manager');  $this->rollcall\_manager->closeRollCall();   1. 点名业务逻辑类先判断当前时间是否大于点名结束时间，若大于则更新条件中只有点名状态；若不大于则更新条件中有点名结束时间和点名状态。   $update = array();  if($now < $endTime){  $update = array('endTime'=>$now);  }  if($flag == 'end'){  $update = array('status'=>2);  }elseif($flag == 'cancel'){  $update = array('status'=>3);  }   1. 点名业务逻辑类调用点名数据库模型类(即rollcall\_model)里的update()方法更新点名状态及点名结束时间。   $this->CI->load->model('rollcall\_model');  $this->CI->rollcall\_model->update();   1. 点名数据库模型类将更新结果返回给点名业务逻辑类。 2. 点名业务逻辑类通过$flag的值来判断是否获取所有签到的参会人信息。   if($flag == 'end'){  $this->getSignInfo();  }   1. 点名业务逻辑类将组合的返回值返回给控制器。 2. 控制器将返回值返回给客户端。 |
| **备注** | 取消点名也用此接口(返回值不包含所有签到的参会人信息) |

### **签到**

|  |  |
| --- | --- |
| **描述** | 签到 |
| **HLD ID** | MAX-HLD-ROLLCALL-04 |
| **访问链接** | https://maxuniform.quanshi.com/client/closeRollCall |
| **请求方式** | post |
| **参数列表** | |  |  | | --- | --- | | 参数 | 说明 | | $openRollcallTime | 开始点名时间戳 | | $tempConferenceId | 临时会议id | | $attendeeSign | 签到人信息（必须有临时用户id，邮箱可以为空） | |
| **返回结果** | |  |  | | --- | --- | | 参数 | 说明 | | response | {"result":"","status":"业务状态码参考备注"} | | content-type | 返回数据类型(application/json) | |
| **依赖外部系统** | N/A |
| 逻辑描述 | 1. 客户端通过ajax向web服务器maxuniform发出签到请求。 2. 服务器根据请求链接，指定点名控制器(即rollcall)里的sign()处理请求。 3. 点名控制器调用点名业务逻辑类(即rollcall\_manager)里的sign()方法来处理签到逻辑。   $this->load->library('rollcall\_manager');  $this->rollcall\_manager->sign();   1. 点名业务逻辑类先判断点名状态是否允许，即调用点名数据库模型类(即rollcall\_model)里的get()方法。   $this->CI->load->model('rollcall\_model');  $this->CI->rollcall\_model->get();   1. 点名业务逻辑类根据点名状态及点名结束时间来判断是否允许签到,若不允许则直接返回。   if($status == 2 || $status == 3 || $endTime < $now){  return 50002;  }   1. 点名业务逻辑类再判断此人是否已经签到，即调用类里的方法获取某个参会人是否已签到。   $this->getAttendeeSign();   1. 点名业务逻辑类通过返回值来判断其是否已签到，若为0则已签到，若为1则未签到。 2. 点名业务逻辑类调用点名数据库模型类的save()方法。 3. 点名数据库模型类将插入结果返回给点名业务逻辑类。 4. 点名业务逻辑类将结果返回给控制器。 5. 控制器将结果返回给客户端。 |
| **备注** |  |

### **获取所有已签到的参会人(内部函数)**

|  |  |
| --- | --- |
| **描述** | 获取所有已签到的参会人 |
| **HLD ID** | MAX-HLD-ROLLCALL-020 |
| **访问链接** | https://maxuniform.quanshi.com/client/getSignInfo |
| **请求方式** | get |
| **参数列表** | |  |  | | --- | --- | | 参数 | 说明 | | $openRollcallTime | 开始点名时间戳 | | $tempConferenceId | 临时会议id | |
| **返回结果** | |  |  | | --- | --- | | 参数 | 说明 | | response | {"result":"所有的签到参会人信息(email/tempUserId)",  "status":"成功:0失败:50001"} | | content-type | 返回数据类型(application/json) | |
| **依赖外部系统** | N/A |
| 逻辑描述 | 1. 点名业务逻辑类(即rollcall\_manager)里的getSignInfo()方法调用点名数据库模型类来获取所有已经签到的参会人信息。   $this->CI->load->model('rollcall\_model');  $this->CI->rollcall\_model->get();   1. 点名数据库模型类将所有已经签到的参会人信息返回。 2. 点名业务逻辑类接收所有已经签到的参会人信息并作为返回值返回。 |
| **备注** | N/A |

### **获取某个参会人是否已经签到**

|  |  |
| --- | --- |
| **描述** | 获取某个参会人是否已经签到 |
| **HLD ID** | MAX-HLD-ROLLCALL-06 |
| **访问链接** | https://maxuniform.quanshi.com/client/getAttendeeSign |
| **请求方式** | get |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 参数 | 说明 | | $openRollcallTime | 开始点名时间戳 | | $tempConferenceId | 临时会议id | | $attendeeSign | 签到人信息（临时用户id，邮箱） | |
| **返回结果** | |  |  | | --- | --- | | 参数 | 说明 | | response | {"result":"当前时间戳","status":"未签到1，已签到0"} | | content-type | 返回数据类型(application/json) | |
| 逻辑描述 | 1. 客户端通过ajax向web服务器maxunifrom发出获取某个参会人是否已签到的请求 2. 服务器根据请求链接，指定点名控制器(即rollcall)里的getAttendeeSign()方法来判断此人是否已签到。 3. 点名控制器调用点名业务逻辑类(即rollcall\_manager)里的getAttendeeSign ()方法来获取此参会人的签到信息。   $this->load->library('rollcall\_manager');  $this->rollcall\_manager-> getAttendeeSign();   1. 点名业务逻辑类调用点名数据库模型类(即rollcall\_model)里的get()方法来获取参会人签到信息。   $this->CI->load->model('rollcall\_model');  $this->CI->rollcall\_model->get();   1. 点名数据库模型类将参会人签到信息返回给点名业务逻辑类。 2. 点名业务逻辑类根据参会人签到信息来判断是否已经签到过并将结果返回给点名控制器。 3. 控制器根据manage类的返回值处理并返回给客户端。 |
| **备注** | N/A |

### **入会情况进度指示条**

## 通讯录

### 导入通讯录

|  |  |
| --- | --- |
| **描述** | 客户端发送post请求，上传通讯录文件到服务器，Uniform读取上传的通讯录文件，校验其中的字段，对每个字段加入枚举标识。 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-EXCEL-001 |
| **访问链接** | <http://maxuniform.quanshi.com/excel/import>/{$userId} |
| **请求方式** | POST |
| **数据类型** | application/vnd.ms-excel |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID | | $data | 通讯录文件 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码 | | result |  | | content\_type | application/xml | |
| **依赖外部系统** |  |
| 逻辑描述 | 1. 客户端发送post请求，调用控制器excel类的import函数 2. 控制器excel调用管理类excel\_manager的import函数   $this->load->manager(‘excel\_manager’);  $this->excel\_manager->import();   1. 管理类excel\_manger调用contacter\_helper的vilidate\_file()函数，验证文件的有效性，如果不合要求则直接返回错误信息。   $this->load->helper(‘contacter\_helper’);  $this->contacter\_helper->validate\_file();   1. 管理类excel\_manager调用自身的reader方法，读出文件中的数据，放入数组五个数组：$head\_arr, $data\_arr, $group\_arr, $email\_arr, $moblie\_arr    1. 对$head\_arr 检验，不符合要求则返回错误代码。    2. 对 $group\_arr 去重，计算长度，如果大于100，返回错误码    3. 对 $group\_arr 的每条记录进行合法性验证，添加在该数组中添加is\_valid字段，调用模型类group\_model的move和insert\_batch方法，将数据标记为历史数据并批量插入group表   $this->load->model(‘group\_model’);  $this->group\_mdl->move();  $this->group\_mdl->insert\_batch($data\_arr);   * 1. 查询获得所插入的group信息，存进数组$group\_from\_base   2. 使用[array\_count\_values()](http://www.w3school.com.cn/php/func_array_count_values.asp)方法获取$email\_arr 和 $mobile\_arr 中重复的值   3. 根据4.5 对$data\_arr 中重复值（email 或手机重复）去重，并记录最后一条可显示   4. 调用帮助类 contact\_helper 中各个字段的valid方法，验证每个字段的有效性，并在$data\_arr添加每个字段的status值  1. 调用模型类 contact\_model的to\_history方法，将以前的数据标记为历史数据，然后调用模型类contact\_model的insert\_batch方法将$data\_arr 插入表contact   $this->load->model(‘contact\_model’);  $this->contact\_model->to\_history();  $this->contact\_model->insert\_batch($data\_arr); |
| **备注** | 关于回滚操作  见4.4.2 |

### 回滚(内部函数)

|  |  |
| --- | --- |
| **描述** | 客户端发生请求超时或者5xx错误时发送get请求，uniform做回滚操作 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-EXCEL-002 |
| **访问链接** | http://maxuniform.quanshi.com/excel/callback/{$userId} |
| **请求方式** | GET |
| **数据类型** |  |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码 | | result |  | | content\_type |  | |
| **逻辑描述** | 1. 删除数据库contact 和group表中的记录   2. 将数据库表contact和group中的历史数据标记为有效数据 |
| **备注** |  |

### 导出通讯录

|  |  |
| --- | --- |
| **描述** | 客户端发送get请求，uniform提供通讯录的下载功能 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-EXCEL-003 |
| **访问链接** | http://maxuniform.quanshi.com/excel/export |
| **请求方式** | GET |
| **数据类型** |  |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $user\_id | 主持人帐号对应的用户ID | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码 | | result | 下载文件 | | content\_type | 返回数据格式 | |
| **逻辑描述** | 1. 客户端发出get请求，调用控制器类excel的export方法 2. 控制器类excel调用管理类excel\_manger的export方法   $this->load->library(‘excel\_manger’);  $this->excel\_manger->export();   1. 管理类excel\_manger调用模型类file\_model的get\_file方法   $this->load->model(‘file\_model’);  $this-> file\_model -> get\_file ();  3.1 数据库file表中有该用户的文件记录  3.1.1 判断服务器上是否存在该文件，和是否曾经更改  3.1.1.1 服务器上有该文件，且没有更改过，不做处理  3.1.1.2服务器上有该文件，显示通讯录被更改过，删除此文件，调用管理类contact\_manager的get\_all\_contacts()方法，获得所有联系人列表信息，调用自身的writer()方法，生成excel文件，调用模型类file\_model的update()更新数据库中文件信息。  $this->load-> library (‘contact\_manager’);  $this->contacter\_manager-> get\_all\_contacts ();  $this->writer($data\_arr);  $this->load-> model (‘file\_model’);  $this->file\_model ->update ();  3.1.1.3服务器上没有该文件，调用管理类contact\_manager的get\_all\_contacts()方法，获得所有联系人列表信息，调用自身的writer()方法，生成文件，调用file\_model的update()更新数据库  $this->load-> library (‘contact\_manager’);  $this-> contact\_manager -> get\_all\_contacts ();  $this->writer($data\_arr);  $this->load-> model (‘file\_model’);  $this->file\_model ->update ();  4. 提供下载 |
| **备注** | 对管理类excel\_manager的writer方法的说明：  Writer方法通过调用phpExcel库来实现生成excel文件的功能。  1．生成writer  //创建一个新的PHPExcel对象  $objPHPExcel = new PHPExcel();  //设置第一个sheet为活动的  $objPHPExcel->setActiveSheetIndex(0);  //获得当前活动的sheet  $objActSheet = $objPHPExcel->getActiveSheet();  //设置当前活动sheet的名称  $objActSheet->setTitle('导出联系人');  //社置Excel属性  $this->set\_excel\_attr($objPHPExcel);  //生成一个writer  $objWriter = IOFactory::createWriter($objPHPExcel, $new\_type);   1. 设置单元格的值   $objActSheet->setCellValue('A1’, ‘18911536706’);  或  $objActSheet->setCellValueExplicit(‘A1’, ‘zexuan.wu’, PHPExcel\_Cell\_DataType::TYPE\_STRING);   1. 标记单元格颜色   //把单元格A1标记为红色（把A行标记为红色只需要把getStyle的参数设为‘A1’）  $objFill = $objActSheet->getStyle(‘A1’)->getFill();  $objFill->setFillType(PHPExcel\_Style\_Fill::FILL\_SOLID);  $objFill->getStartColor()->setARGB(‘FF0000’); |

### 添加组

|  |  |
| --- | --- |
| **描述** | 在主持人通讯录中添加一个组 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-GROUP-001 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/group/add |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $group\_name | 组名 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(0：成功；其他：失败) | | result | group\_id：组ID；group\_name：组名；version：版本号 | | content\_type | 返回结果的数据类型json | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. MAX客户端向服务器端发出添加组的请求 2. 服务器根据请求链接，指定处理请求组控制器Group处理请求 3. 控制器调用数据校验辅助函数valid\_helper中的valid\_group\_name()函数校验组名是否合法   $this ->load->helper(‘valid’);  $bool = valid\_group\_name();   1. 组名如果合法，控制器调用逻辑处理类Group\_manager中的创建组方法add()   $this->load->library(‘group\_manager’);  $this->group\_manager->add();   1. 逻辑处理类Group\_manager调用自身统计组数量方法count\_by\_user\_id()，逻辑类调用数据模型Group\_model中count\_by\_uid()统计数量方法，检查组名数量是否已达到上限100   $count = $this->count\_by\_user\_id ();  $this->load->model(‘group\_model’);  $cnt = $this->group\_model->count\_by\_uid();   1. 组名如果未达到上限，逻辑处理类Group\_manager调用自身检查组名方法check\_group\_name()，组名不允许重复   $bool = $this-> check\_group\_name()   1. 逻辑类调用数据模型Group\_model中的get\_by\_gname()查询组名方法，查询数据库中是否已存在该组名   $res = $this->group\_model->get\_by\_gname();   1. 组名如果不存在，逻辑处理类调用模型类Group\_model中的save()方法向数据库插入数据   $this->group\_model->save();   1. 控制器分析逻辑类返回信息，返回给MAX客户端新添加的组ID及业务状态码 |
| **备注** |  |

### 删除组

|  |  |
| --- | --- |
| **描述** | 删除主持人通讯录中指定的组及其下的联系人 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-GROUP-002 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/group/del/{group\_id} |
| **请求方式** | GET |
| **数据类型** |  |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $group\_id | 组ID | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(0：成功；其他：失败) | | result | null | | content\_type | 返回结果的数据类型json | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. MAX客户端向服务器端发出删除组的请求  2. 服务器根据请求链接，指定处理请求组控制器Group处理请求   1. 控制器调用组逻辑处理类Group\_manager中的根据组ID删除组方法del\_by\_group\_id()进行删除   $this ->load->library(‘group\_manager’);  $bool = $this-> group\_manager->del\_by\_group\_id();   1. 删除组之前，应先删除此组下的所有联系人，逻辑类Group\_manager调用联系人逻辑处理类Contacter\_manager中的根据组ID删除联系人方法del\_by\_group\_id()   $this->load->library(‘contacter\_manager’);  $bool = $this-> contacter\_manager ->del\_by\_group\_id()   1. 逻辑类Contacter\_manager调用数据模型Contacter\_model中的del\_by\_gid()方法删除数据库中的联系人   $this ->load->model(‘contacter\_model’);  $res = $this->contacter\_model->del\_by\_gid();   1. 逻辑处理类Group\_manager调用数据模型Group\_model中的del\_by\_gid()方法删除数据库中的组   $this->load->model(‘group\_model’);  $res = $this->group\_model->del\_by\_gid();   1. 控制器分析逻辑类Group\_manager返回信息，返回给MAX客户端相应结果 |
| **备注** |  |

### 重命名组

|  |  |
| --- | --- |
| **描述** | 对主持人帐号通讯录中指定的组进行重命名操作 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-GROUP-003 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/group/edit |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $group\_id | 组ID | | $group\_name | 组的新名称 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(0：成功；其他：失败) | | result | group\_id：组ID；group\_name：组名；version：版本号 | | content\_type | 返回结果的数据类型json | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. MAX客户端向服务器端发出重命名组的请求  2. 服务器根据请求链接，指定处理请求组控制器Group的edit()方法处理请求  3. 控制器调用辅助函数valid\_helper 中的valid\_group\_name()校验组名是否合法  $this ->load->helper(‘valid’);  $bool = valid\_group\_name();   1. 组名如果合法，控制器调用逻辑处理类Group\_manager中的编辑组方法edit()进行组名的修改   $this ->load->library(‘group\_manager’);  $bool = $this-> group\_manager->edit();   1. 逻辑处理类Group\_manager调用自身check\_group\_name()方法检查组名是否存在，组名不可重复   $bool = $this-> check\_group\_name();   1. 逻辑类Group\_manager调用数据模型Group\_model中的get\_by\_gname()方法查询数据库中是否已存在该组名   $this->load->model(‘group\_model’);  $id = $this->group\_model->get\_by\_ gname ();   1. 组名如果不存在，逻辑处理类Group\_manager调用数据模型Group\_model中的方法update()向数据库中更新数据   $res = $this->group\_model->update();   1. 控制器分析逻辑类Group\_manager返回信息，返回给MAX客户端相应结果 |
| **备注** |  |

### 添加联系人

|  |  |
| --- | --- |
| **描述** | 为主持人通讯录中添加联系人信息 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-CONTACTER-001 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/contacter/add.html |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $request\_data | 请求数据，包含以下字段：  group\_name：组名； name：联系人姓名；  email：邮箱地址； mobile\_cnum：手机号对应的国家码；  moblie：手机号； tel\_cnum：固话对应的国家码；  area\_num：区号； tel：固定电话；  exten\_num：分机号； | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 0：添加失败  1：添加成功时返回联系人信息(其中包含组ID、联系人ID)以及新的通讯录版本号； | | content\_type | 返回结果的数据类型(json) | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. 客户端向接口发送添加联系人操作的请求； 2. 系统联系人控制器(contacter类)接收到请求后对请求数据进行有效性验证；   $this->load->helper(‘valid);  contacter\_valid($request\_data);   1. 联系人辅助函数(valid\_helper类)将验证结果返回给控制器contacter类； 2. contacter类验证请求数据，如果验证不通过，则直接返回给客户端请求结果错误；如果验证通则开始调用业务逻辑层的联系人类(contacter\_manager类)的添加联系人方法；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->add($request\_data);   1. contacter\_manager类在添加联系人操作前，验证主持人的通讯录中联系人总人数是否已经达到上限；   $this->count\_by\_user\_id($user\_id);   1. contacter\_manager类调用联系人模型类(contacter\_model类)统计主持人通讯录中已有联系人总数；   $this->load->model(‘contacter\_model’);  $this->contacter\_model->count\_by\_uid($user\_id);   1. contacter\_model类将统计的结果返回给contacter\_manager类； 2. contacter\_manager类获取返回统计的总人数后与允许添加联系人最大值进行比对，如果超出最大值则不允许进行添加联系的人操作并直接返回结果；如果没有超出联系人上限值，则进行邮箱是否重复的验证操作；   $this->get\_by\_email($email);   1. contacter\_manager类调用contacter\_model类中的get\_by\_email()方法来判断请求数据中的邮箱信息是否有已存在的记录；   $this->load->model(‘contacter\_model’);  $this->contacter\_model->get\_by\_email($email);   1. contacter\_model类获取查询的结果并返回给contacter\_manager类； 2. contacter\_manager类通过返回的结果是否为空判断邮箱是否已存在；如果返回结果不为空则表示该邮箱已被使用，系统则不进行添加联系人的操作；如果返回结果为空，则表示该邮箱可用，系统则再进行手机号是否重复的验证操作；   $this->get\_by\_mobile($mobile);   1. contacter\_manager类调用contacter\_model类中的get\_by\_mobile()方法判断请求数据中的手机号是否有已存在的记录；   $this->contacter\_model->get\_by\_mobile($mobile);   1. contacter\_model类将查询的结果返回给contacter\_manager类； 2. contacter\_manager类通过返回的结果是否为空判断手机号是否已存在；如果存在则表示该手机号已被使用，系统则不进行添加联系人的操作；如果不存则根据请求数据中的组名称来获取组ID信息；   $this->load->library(‘group\_manager’);  $this->group\_manager->get\_by\_group\_name($user\_id, $group\_name);   1. group\_manager类将组的查询结果返回给contacter\_manager类； 2. contacter\_manager类根据group\_manager类返回的结果判断待查询的组是否存在，如果存在则直接返回组ID信息；如果不存在则需要进行添加组的操作；   $this->load->library(‘group\_manager’);  $this->group\_manager->add($user\_id, $group\_name);   1. group\_manager类返回给contacter\_manager类查询出来的组ID信息； 2. contacter\_manager类获取到组的ID信息后开始进行联系人添加操作； 3. contacter\_manager类调用联系人模型类(contatcter\_model类)中的添加操作；   $this->contacter\_model->save($request\_data);   1. 联系人模型类将操作的结果返回给contacter\_manager类； 2. contacter\_manager类接收返回的添加操作的结果并返回给控制器； 3. 联系人控制器将添加操作的请求结果返回给客户端； |
| **备注** |  |

### 编辑联系人

|  |  |
| --- | --- |
| **描述** | 对主持人通讯录中指定的联系人信息进行修改操作 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK- CONTACTER -002 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/contacter/edit.html |
| **请求方式** | POST |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $contacter\_id | 联系人ID | | $request\_data | 请求数据，包含以下字段：  group\_name：组名； name：联系人姓名；  email：邮箱地址； mobile\_cnum：手机号对应的国家码；  moblie：手机号； tel\_cnum：固话对应的国家码；  area\_num：区号； tel：固定电话；  exten\_num：分机号； | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 如果编辑成功则返回新的通讯录版本号； | | content\_type | 返回结果的数据类型(json) | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. 客户端向接口发送编辑联系人请求； 2. 联系人控制器类(contacter类)接收到请求后调用验证辅助函数(valid\_helper)类进行数据有效性验证；   $this->load->helper(‘valid’)；  contacter\_valid($request\_data);   1. 系统调用辅助函数类(valid\_helper类)将验证结果返回给控制器contacter类； 2. contacter类根据返回的结果判断是否进行下一步的操作，如果请求数据验证无效，contacter类将直接返回失败的结果；如果请求数据验证有效，contacter类则开始调用联系人业务逻辑类(contacter\_manager类)中的edit()方法对联系人信息进行编辑操作；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->edit($request\_data);   1. contacter\_manager类在进行编辑操作前需要通过调用本类中的根据邮箱获取联系人信息的方法get\_by\_email()来判断当前请求数据中的邮箱地址是否已被使用； 2. contacter\_manager类调用联系人模型类(contacter\_model类)中的get\_by\_email()方法查询请求数据中的邮箱地址是否已被其他人使用；   $this->load->model(‘Contacter\_model’);  $this->contacter\_model->get\_by\_email($email);   1. contacter\_model类将查询的结果返回给contacter\_manager类； 2. contacter\_manager类根据返回结果是否为空来判断邮箱是否已被使用，如果邮箱被使用则不进行联系人的编辑操作并返回操作失败的结果；如果邮箱没有被使用，contacter\_manager类则调用来类中的根据手机号获取联系人信息的方法get\_by\_mobile()再验证请求数据中的手机号是否存在；   $this->get\_by\_mobile($mobile);   1. contacter\_manager类调用联系人模型类(contacter\_model类)中的get\_by\_moblie()方法查询请求数据中的手机号是否已被其他人使用；   $this->contacter\_model->get\_by\_mobile($mobile);   1. contacter\_model类将查询的结果返回给contacter\_manager类； 2. contacter\_manager类根据返回是否为空来判断手机号是否已被使用，如果已被使用则不进行联系人的编辑操作并返回操作失败的结果；如果未被使用，contacter\_manager类则根据请求数据的组名称获取组的ID信息；   $this->load->library(‘group\_manager’);  $this->group\_manager-> get\_by\_group\_name ($user\_id, $group\_name);   1. group\_manager类返回查询的组信息给contacter\_manager类，contacter\_manager类判断是否需要进行新建组操作，如果返回结果为空，则需要进行添加组的操作； 2. contacter\_manager类调用contacter\_model类进行添加组操作；   $this->load->library(‘group\_manager’);  $this->group\_manager->add($user\_id, $group\_name);   1. group\_manager类将获取出来的组ID信息返回给contacter\_manager类； 2. contacter\_manager类进行编辑联系人的操作； 3. contacter\_manager类调用contacter\_model类中的更新方法update()进行数据更新操作；   $this->contacter\_model->update($request\_data);   1. contacter\_model类将更新操作的结果返回给contacter\_manager类； 2. contacter\_manager类将更新操作的结果返回到contacter类； 3. contacter类将请求结果返回给客户端； |
| **备注** |  |

### 移动联系人

|  |  |
| --- | --- |
| **描述** | 对主持人通讯录中指定的联系人移动到其他组下 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK- CONTACTER -003 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/contacter/move/{contacter\_id}/{new\_group\_id}.html |
| **请求方式** | GET |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $contacter\_id | 联系人ID | | $new\_group\_id | 新组ID | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码（成功：0；失败：非0） | | result | 移动成功时返回新的通讯录版本号； | | content\_type | 返回结果的数据类型(json) | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. 客户端向接口发送移动联系人请求； 2. 系统联系人控制器(contacter类)调用联系人业务逻辑类(即contacter\_manager类)中的移动联系人方法move()；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->move($request\_data);   1. contacter\_manager类调用组业务逻辑类(group\_manager类)中的get\_by\_group\_id()方法进行移动操作前先检查待移动的目标组是否有效，如果有效则进行移动操作，如果无效，则不进行移动操作；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager ->get\_by\_group\_id($group\_id);   1. group\_manager类将查询的结果返回给contacter\_manager类； 2. contacter\_manager类调用联系人业务逻辑类(contacter\_model类)中的update()方法更新联系人的组信息；   $this->contacter\_model->update($data);   1. contacter\_model类将更新操作的结果返回给contacter\_manager类； 2. contacter\_manager类对更新操作的结果返回到contacter类； 3. contacter类将请求结果返回给客户端； |
| **备注** |  |

### 删除联系人

|  |  |
| --- | --- |
| **描述** | 删除主持人通讯录中指定的联系人信息 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK- CONTACTER -004 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/contacter/del/{contacter\_id} |
| **请求方式** | GET |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $contacter\_id | 联系人ID | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 当删除成功时返回新的通讯录版本号； | | content\_type | 返回结果的数据类型(jsonl) | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. 客户端向接口发送删除指定联系人的请求； 2. 系统联系人控制器(contacter类)接收到请求后则调用系统内部联系人业务逻辑类(contacter\_manager类)中的删除联系人del\_by\_contacter\_id()方法进行删除操作；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->del\_by\_contacter\_id($contacter\_id);   1. contacter\_manager类在进行删除操作时需要对数据库进行操作，则系统会调用contacter\_model类进行处理；   $this->load->model(‘contacter\_model’);  $this->contacter\_model->del\_by\_cid($contacter\_id);   1. contacter\_model类将删除操作的结果返回给联系人类contacter\_manager类； 2. contacter\_manager类接收到返回结果并返回给contacter类控制器； 3. contacter类控制器将客户端的请求结果返回； |
| **备注** | 删除联系人为物理操作 |

### 查检邮箱/手机号是否存在

|  |  |
| --- | --- |
| **描述** | 查询主持人通讯录中是否存在待提供的邮箱地址 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK- CONTACTER-005 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/contacter/checkExist/{key}/{value} |
| **请求方式** | GET |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $key | 待验证的字段(可供选择字段：email、mobile) | | $value | 待验证字段对应的值 | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 有结果则返回当前字段值对应的联系人信息；  无结果则返回为空； | | content\_type | 返回结果的数据类型(json) | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. 客户端向接口发送验证邮箱/手机号是否已存在的请求； 2. 系统联系人控制器(cotacter类)接收到请求后判断是进行邮箱的重复性验证还是手机号的重复性验证。当进行邮箱的验证时调用系统内部的辅助函数valid\_helper类对请求数据进行有效性验证；   $this->load->helper(‘valid\_helper’);  valid\_email();   1. valid\_helper类将邮箱数据有效性验证结果返回给contacter类控制器； 2. 邮箱验证通过后contacter类调用系统内部的联系人业务逻辑类(contacter\_manager类)验证请求邮箱数据信息是否存在；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->get\_by\_email($email);   1. contacter\_manager类调用系统内部的联系人模型类(contacter\_model类)查询数据库中是否有符合该邮箱的记录；   $this->load->model(‘contacter\_model’);  $this->contacter\_model->get\_by\_email($email);   1. contacter\_model类将查询的结果返回给管理层的联系人contacter\_manager类； 2. contacter\_manager类将接收到的返回结果返回给contacter类； 3. 系统进行手机号码的验证操作，则联系人控制器contacter类调用系统内部的辅助函数valid\_helper类对请求数据进行有效性验证；   $this->load->helper(‘valid’);  valid\_mobile();   1. valid\_helper类将手机号数据有效性验证结果返回给contacter类控制器； 2. 手机号验证通过后contacter类调用系统内部的联系人业务逻辑类(contacter\_manager类)验证请求手机号数据信息是否存在；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->get\_by\_mobile($mobile);   1. contacter\_manager类调用系统内部的联系人模型类(contacter\_model类)查询数据库中是否有符合该手机号的联系人记录； 2. contacter\_model类将查询的结果返回给联系人业务逻辑类(contacter\_manager类)； 3. contacter\_manager类将查询的结果返回给控制器； 4. contacter控制器将客户端的请求结果返回； |
| **备注** |  |

### 获取通讯录信息

|  |  |
| --- | --- |
| **描述** | 获取主持人的所有通讯录信息 |
| **HLD ID** | MAXUniform-HLD-ADDRESSBOOK-CONTACTER-006 |
| **访问链接** | https://maxuniform.quanshi.com/addressbook/contacter/getByUserId/{version\_time}/{is\_force} |
| **请求方式** | GET |
| **数据类型** | json/xml |
| **参数列表** | |  |  | | --- | --- | | 字段参数 | 说明 | | $version\_time | 版本时间戳(默认值：当前时间戳) | | $is\_force | 是否强制获取(默认值FALSE：表示不获取；TRUE：表示强制获取通讯录信息，该情况下不比对版本时间戳参数，返回结果中会带上新的时间戳) | |
| **返回结果** | |  |  | | --- | --- | | 字段参数 | 说明 | | status | 业务状态码(例如50001) | | result | 有结果则返回当主持人对应的所有联系人信息并带上新的时间戳；  无结果则返回为空； | | content\_type | 返回结果的数据类型(json) | |
| **依赖外部系统** |  |
| **逻辑描述** | 1. 客户端向接口发送获取通讯录信息的请求； 2. 系统联系人控制器(contacter类)判断请求过来的通讯录版本号是否与查询出来的版本号一致，如果一致则表示客户端的通讯录信息是最新的，接口则返回空值；如果版本号不一致，contacter类则接口调用系统内部的联系人业务逻辑类(contacter\_manager类)中的get\_by\_user\_id()方法获取主持人的所有联系人信息；   $this->load->library(‘contacter\_manager’);  $this->contacter\_manager->get\_by\_user\_id($user\_id);   1. contacter\_manager类调用系统内部的联系人模型类(contacter\_model类) 中的get\_by\_uid()方法通过查询数据库获取联系人信息；   $this->load->model(‘contacter\_model’);  $this->contacter\_model->get\_by\_uid($user\_id);   1. contacter\_model类将从数据库中查询出来的结果返回给contacter\_manager类； 2. contacter\_manager类调用验证辅助函数类(valid\_helper类)将返回的数据进行排序处理，先对组进行排序操作；   $this->load->helper(‘valid’);  sort\_group($data);   1. valid\_helper类将组排序好后的结果返回给contacter\_manager类； 2. contacter\_manager类对返回的数据再进行组内联系人排序操作；   sort\_contacter($data);   1. valid\_helper类将联系人信息排序完成后的结果返回给contacter\_manager类； 2. contacter\_manager类将获取的通讯录信息返回给控制器； 3. contacter类控制器将客户端的请求结果返回； |
| **备注** |  |

# 数据结构设计

## 数据结构图

## 数据字典表

### 参会人临时表（max\_conference\_invitee\_temp）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | bigint(20) | 主键，自增长字段 |
| user\_id | bigint(20) | 帐号id 索引类型：normal ；  索引方式：btree |
| group\_name | varchar(255) | 组名 |
| email | varchar(255) | 参会人邮箱 |
| mobile | varchar(100) | 参会人手机号 |
| name | varchar(100) | 参会人姓名 |
| email\_status | tinyint(1) | 0:合法；1：不合法；2：重复 |
| mobile\_status | tinyint(1) | 0:合法；1：不合法；2：重复 |
| name\_status | tinyint(1) | 0:合法；1：不合法；2：重复 |

### 参会人设置表（max\_conference\_attendee\_setting）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | int(11) | 主键，自增长字段 |
| conference\_id | varchar(20) | 预约会议id 索引类型：normal ；索引方式：btree |
| key | varchar(100) | 键 |
| value | varchar(100) | 值 |
| user\_id | bigint(20) | 帐号id索引类型：normal ；索引方式：btree |

### 参会人表（max\_conference\_invitee）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | bigint(20) | 主键，自增长字段 |
| conference\_id | varchar(20) | 帐号id索引类型：normal ；索引方式：btree |
| group\_name | varchar(255) | 组名 |
| email | varchar(255) | 参会人邮箱 |
| mobile | varchar(100) | 参会人手机号 |
| name | varchar(100) | 参会人姓名 |
| is\_during\_meet | tinyint(1) | 是否会中邀请 0：会前邀请；1：会中邀请 |

### 帐号信息表（max\_conference）添加2个字段

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| pwdtype | tinyint(1) | 密码类型：0:固定密码；1:安全密码 |
| informtype | tinyint(1) | 通知方式：1：邮件；2：短信；3：邮件+短信 |

### 点名时间表（max \_rollcall）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | bigint(20) | 主键，自增长字段 |
| userId | bigint(20) | 主持人用户id |
| tempConferenceId | varchar(20) | 临时会议id |
| rollcallId | varchar(30) | 每次点名标示(时间戳+临时会议id),索引 |
| startTime | datetime | 点名开始时间 |
| endTime | datetime | 点名结束时间 |
| status | tiyint(1) | 点名状态（1点名中 2 结束点名3：取消点名） |
| createTime | datetime | 创建时间 |
| lastmodifyTime | datetime | 最后修改时间 |

### 签到表（max\_rollcall\_sign）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | bigint(11) | 主键，自增长字段 |
| rollcallId | varchar(30) | 每次点名标示(时间戳+临时会议id)，索引 |
| tempUserId | varchar(50) | 临时用户id，允许为空 |
| email | varchar(255) | 用户邮箱，允许为空 |
| createTime | datetime | 创建时间 |

### 联系人（max\_ addressbook \_contact）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **类型** | **字段说明** | **备注** |
| id | bigint(20) | 主键，自增长字段 | 主键 |
| user\_id | bigint(20) | 用户id(来自UMS) | 索引 |
| product\_id | bigint(20) | 产品id |  |
| name | varchar(60) | 联系人姓名 |  |
| group\_id | bigint(20) | 组id | 索引 |
| email | varchar(120) | 联系人邮箱 |  |
| mobile\_country\_code | varchar(10) | 手机国家码 |  |
| mobile | varchar(30) | 手机 |  |
| tel\_country\_code | Varchar(10) | 固话国家码 |  |
| tel\_area\_code | varchar(10) | 固话区号 |  |
| telephone | varchar(30) | 固话 |  |
| tel\_ext\_code | varchar(10) | 固话分机号 |  |
| name\_status | TINYINT(1) | 姓名状态 |  |
| email\_ status | TINYINT(1) | 邮箱状态 |  |
| mobile\_country\_code\_status | TINYINT(1) | 手机号国家码状态 |  |
| mobile\_status | TINYINT(1) | 手机状态 |  |
| tel\_country\_code\_status | TINYINT(1) | 固话国家码状态 |  |
| area\_code\_status | TINYINT(1) | 固话区号状态 |  |
| tel\_status | TINYINT(1) | 固话状态 |  |
| ext\_code\_status | TINYINT(1) | 分机号状态 |  |
| group\_name\_status | TINYINT(1) | 组名状态 |  |
| is\_valid | TINYINT(1) | 是否有效 |  |
| is\_history | TINYINT(1) | 是否是历史数据 |  |
| start\_time | BIGINT(20) | 成为历史数据的时间 |  |
| name\_first\_character\_type | Tinyint(1) | 1,首字符数字，2、首字符英文，3、首字符汉字，0，空 |  |
| name\_quanpin | Varchar(100) | 联系人姓名全拼 |  |

### 联系人历史表（max\_ addressbook\_contact\_history）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | bigint(20) | 主键，自增长字段 |
| user\_id | bigint(20) | 用户id(来自UMS) |
| product\_id | bigint(20) | 产品id |
| name | varchar(60) | 联系人姓名 |
| group\_id | bigint(20) | 组id |
| email | varchar(120) | 联系人邮箱 |
| mobile\_country\_code | varchar(10) | 手机国家码 |
| mobile | varchar(30) | 手机 |
| tel\_country\_code | Varchar(10) | 固话国家码 |
| tel\_area\_code | varchar(10) | 固话区号 |
| telephone | varchar(30) | 固话 |
| tel\_ext\_code | varchar(10) | 固话分机号 |
| name\_status | TINYINT(1) | 姓名状态 |
| email\_ status | TINYINT(1) | 邮箱状态 |
| mobile\_country\_code\_status | TINYINT(1) | 手机号国家码状态 |
| mobile\_status | TINYINT(1) | 手机状态 |
| tel\_country\_code\_status | TINYINT(1) | 固话国家码状态 |
| area\_code\_status | TINYINT(1) | 固话区号状态 |
| tel\_status | TINYINT(1) | 固话状态 |
| ext\_code\_status | TINYINT(1) | 分机号状态 |
| group\_name\_status | TINYINT(1) | 组名状态 |
| is\_valid | TINYINT(1) | 是否有效 |
| is\_history | TINYINT(1) | 是否是历史数据 |
| start\_time | BIGINT(20) | 成为历史数据的时间 |
| name\_first\_character\_type | Tinyint(1) | 1,首字符数字，2、首字符英文，3、首字符汉字，0，空 |
| name\_quanpin | Varchar(100) | 联系人姓名全拼 |

### 组（max\_ addressbook \_group）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **类型** | **字段说明** | **备注** |
| id | bigint(20) | 主键，自增长字段 |  |
| user\_id | bigint(20) | 用户id(来自UMS) | 索引 |
| product\_id | bigint(20) | 产品id |  |
| group\_name | varchar(60) | 组名 |  |
| Is\_history | TINYINT(1) | 是否是历史数据 |  |
| start\_time | BIGINT(20) | 成为历史数据的时间 |  |
| Group\_name\_first\_character\_type | Tinyint(1) | 1,首字符数字，2、首字符英文，3、首字符汉字，0空 |  |
| Group\_name\_quanpin | Varchar(100) | 组名全拼 |  |

### 组历史表（max\_ addressbook \_group\_history）

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **字段说明** |
| id | bigint(20) | 主键，自增长字段 |
| user\_id | bigint(20) | 用户id(来自UMS) |
| product\_id | bigint(20) | 产品id |
| group\_name | varchar(255) | 组名 |
| start\_time | BIGINT | 导入历史表的时间 |
| name\_first\_character\_type | Tinyint(1) | 1,首字符数字，2、首字符英文，3、首字符汉字 |
| name\_quanpin | Varchar(100) | 组名全拼 |

### 文件表（max\_addressbook\_file）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **类型** | **字段说明** | **备注** |
| id | bigint(20) | 主键，自增长字段 |  |
| user\_id | bigint(20) | 用户id(来自UMS) | 索引 |
| product\_id | bigint(20) | 产品id |  |
| path | varchar(100) | 通讯录文件存放地址 |  |
| modify\_time | BIGINT(20) | 最后被更改的时间 |  |

### 版本号表（max\_addressbook\_version）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **类型** | **字段说明** | **备注** |
| id | bigint(20) | 主键，自增长字段 |  |
| user\_id | bigint(20) | 用户id(来自UMS) | 索引 |
| product\_id | bigint(20) | 产品id |  |
| version | Bigint(20) | 版本号（时间戳） |  |
| Type | Tinyint(1) | 版本号类型，1.通讯录 |  |

# 系统出错处理设计

## 出错信息

|  |  |  |  |
| --- | --- | --- | --- |
| 输出信息 | 输出形式 | 含义 | 处理方法 |
| apache服务器异常 | Linux服务器记录系统日志 | 客户端访问apache服务器时，所有apache停止工作 | Linux服务器记录系统崩溃错误日志并报警，运维人员在5分钟内解决问题，接口返回404错误 |
| 数据库访问失败 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | 访问mysql时失败 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，php返回相关错误码 |
| 服务器空间不足 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | 用户上传文件时，服务器空间不足，导致上传失败 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，接口返回（（503错误）） |
| PC2MIS服务器访问失败 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | 访问PC2MIS服务器异常 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，接口返回（（503错误）） |
| UMS服务器访问失败 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | 访问UMS服务器异常 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，接口返回（（503错误）） |
| MSS链接失败 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | apache服务器在访问MSS服务器发送邮件时，MSS数据库链接失败 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，接口返回（（503错误）） |
| CMS访问失败 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | 访问TANG的CMS进行会议的操作时，访问失败或调用接口失败 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，接口返回（（503错误）） |
| 短信服务器连接失败 | PHP记录错误日志，并将错误信息写入linux服务器系统日志Syslog | 调用短信服务器发送短信时，连接短信服务器失败 | 运维监控系统发现Syslog出现异常日志，报警，运维人员在5分钟内介入处理，接口返回（（503错误）） |