企业应用接入手册

发布日期：2014 年 9 月

版本：V3.0.0

目　　录

1 介绍

敏行平台是基于社交网络理念的社交化平台，一个统一的人和人、人和企业、人和企业应用之间互通与互动的平台。平台的集成能力包括如下：

- 公众号接入方式，可以让不支持移动访问的企业应用快速实现移动接入，无需为每个企业应用单独开发移动 App，只需要基于敏行平台进行少量 HTML5 开发，即可实现类微信的公众号的接入以及企业社交网络
- Web 方式嵌入，可以在企业应用的 web 页面中嵌入平台提供的 javascript 代码即可快速为企业应用增加企业社交功能，
- Native 方式，提供原生（Native）打包方案可以直接整合已有的原生应用能够快速接入企业已有的移动客户端，提供 SSO 方案能够直接启动已有客户端，形成企业私有应用商店，应用客户端如果未安装，平台会引导用户自动安装该应用

2 公众号接入

2.1 快速入门

该章实现一个最简单的集成功能，实现敏行平台与 OA 系统进行初步集成，演示集成的概况。实现一个简单功能：查询 OA 的待办数量。

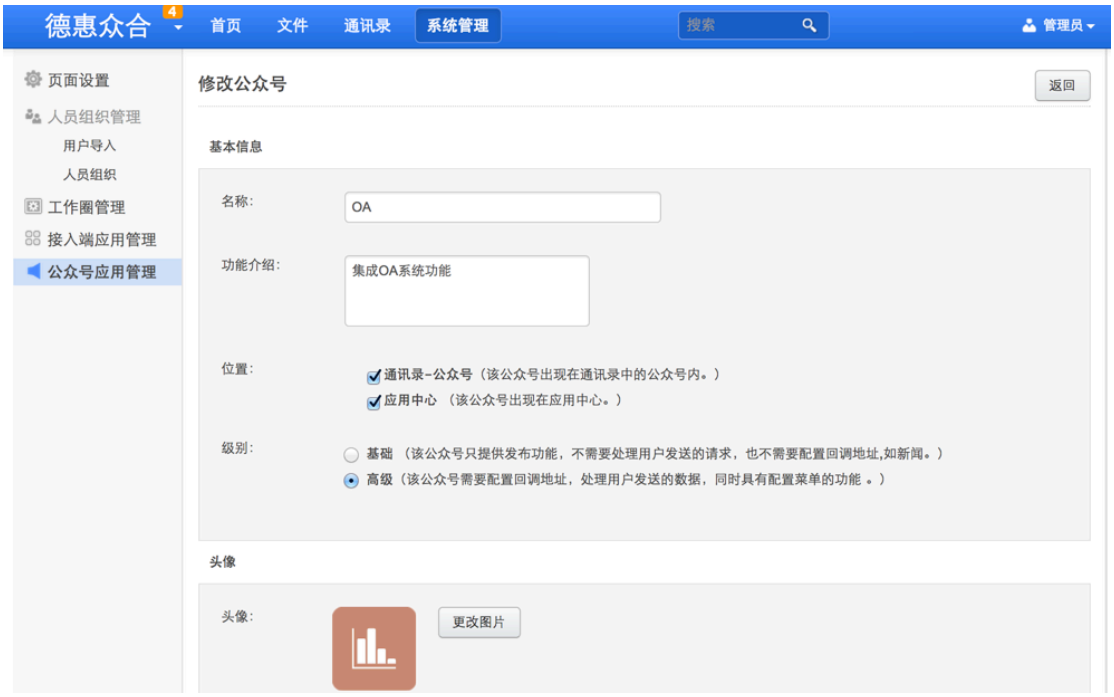
2.1.1 敏行平台添加公众号

通过社区管理员登录系统，在系统管理-公众号中添加公众号，如图。

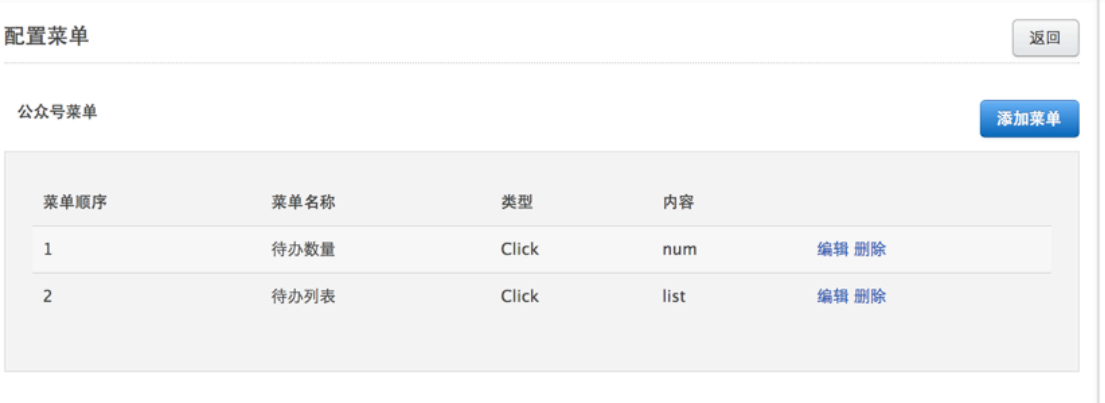
关键项说明：

- 级别：高级，才能配置菜单
- 回调地址：OA 系统端的地址，能接受公众号点击菜单时发送过去的请求。比如 OA 系统端的 servlet。
- 配置类型为 click 的菜单。

minxing365.com 敏行 移动互联时代的工作方式



minxing365.com 敏行 移动互联时代的工作方式



2.1.2 oa 系统二次开发

2.1.2.1 开发环境中引入敏行提供的 sdk

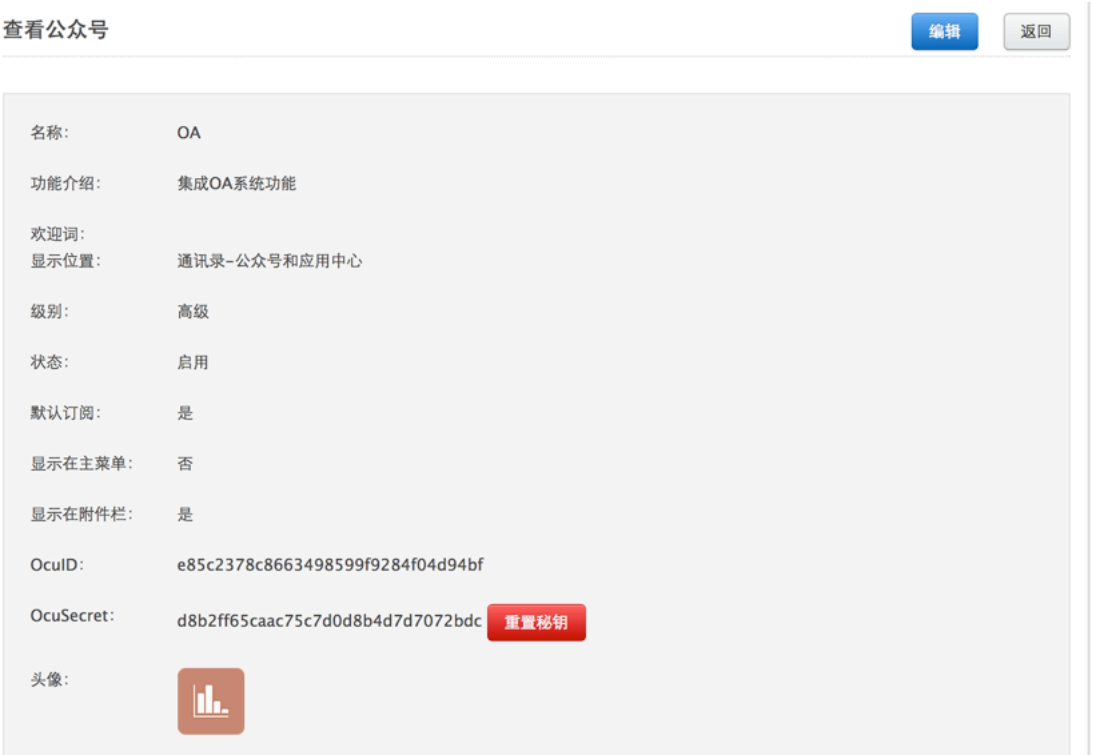
mx-java-sdk.jar
commons-codec.jar
commons-httpclient-3.1.jar
commons-logging-1.1.jar
dom4j-1.6.1.jar
jackson-all-1.8.5.jar
log4j-1.2.16.jar
servlet-api-2.3.jar

2.1.2.2 引入敏行提供的配置文件

将 mx_ocu.properties 放到 source 目录下，修改配置文件 mx_ocu.properties

api_prefix = /api/v1
root_url = http://**敏行平台的地址 ip:port**/
OcuID = e85c2378c8663498599f9284f04d94bf
OcuSecret = d8b2ff65caac75c7d0d8b4d7d7072bdc

OcuID 和 OcuSecret 参考截图



2.1.2.3 编写回调地址的 servlet

点击菜单“查询待办数量“会请求到公众号中配置的回调地址，回调地址对应的是个 servlet，直接返回待办数量到公众号对话中展示。

```
package com;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Search extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String content= "您的待办数量有15条！";
        response.setContentType("application/html");
        response.setCharacterEncoding("utf-8");
        response.getWriter().write(content);
        response.getWriter().flush();
        response.getWriter().close();
    }
}
```

2.2 公众号详述

通过社区系统管理员，在平台中注册公众号，需要提供以下基本信息：

参数	描述
名称	公众号显示的名称
功能介绍	公众号的功能介绍信息
欢迎词	订阅公众号的时候给用户推送的欢迎词消息
级别	基础 （该公众号只提供发布功能，不需要处理用户发送的请求，也不

	需要配置回调地址,如新闻。） 高级（该公众号需要配置回调地址，处理用户发送的数据，同时具有配置菜单的功能 。）
状态	禁用后就不显示公众号
默认订阅	是 （系统强制订阅公众号，用户不能取消订阅。） 否 （该公众号需要用户主动订阅，并且用户可以随时取消订阅。）
显示在附件栏	是 （该公众号对应的应用将出现在聊天的附件栏。） 否 （该公众号对应的应用不出现在聊天的附件栏。）
显示在主菜单	是 （该公众号出现在主菜单。） 否 （该公众号出现在应用中心。）
强制置顶顺序	数字类型，按照升序排列，只有强制订阅的公众号并且要在手机上强制置顶才需要填写该字段
回调地址	用户在公众号里发送消息和事件（包括：菜单点击、订阅、取消订阅）回请求该地址。
SSOKey	设置公众号对应的系统与平台中用户通过哪个字段进行 SSO
启动地址	(如果启动 Native 应用，启动地址以 native://launchApp 开头，同时需要填写以下对应 Native 配置)
android_launcher	(android 应用启动的入口)。举例： url 必须先配置完成。 1. 菜单方式：当需要将第三方系统集成到敏行中，同时第三方入口需要显示在敏行首页的底部菜单上，且第三方系统要求嵌入在敏行首页框架中显示时，需要将第三方系统和敏行打包成一个工程，并且配置嵌入 view 的类的 classpath ，eg: com.minxing.demo.widget.CRMView。 2. 当第三方入口为应用中心或者附件的地方，不需要嵌入显示的其他位置时，则不要求必须与敏行打包，可以是独立的工程，由敏行调用第三方程序启动。需要传递的参数是

	packageName://launchClassName 3. 例如：com.minxing.demo://com.minxing.demo.CRMActivity
Android 安装地址	如果需要启动第三方 app,同时手机端没有安装这个第三方 app,则引导用户到该地址去下载安装
iOS Scheme	(iOS 应用启动需要的 Scheme URL)
iOS 安装地址	iOS 应用启动需要的 Scheme URL url 必须先配置完成。 1. 嵌到我们应用内部的，这种情况需要在这个位置填写这个第三方库的启动类的类名，举例：FirstViewController 启动第三方的独立应用，那么这个时候 ios_scheme 就写那个第三方的应用的 scheme， 举例，如果你想调用微信，那么这个地方就需要写 weixin://xxxxx

注册成功以后，浏览查看该公众号，可以看到生成的 OcuID 和 OcuSecret，这两个参数在后续调用 API 的时候需要。

如果注册为高级公众号，还可以配置用户菜单，该菜单会出现在公众号，菜单的基本信息如下：

参数	**描述**
类型	当前支持 URL 和 Click 两种类型
内容	如果是 URL 类型，填写 URL 地址 如果是 Click 类型，填写参数代码，如 001，这个代码回头会发送到应用系统的回调地址中，用于进行业务查询。

2.3 接受消息

2.3.1 验证消息真实性

每次开发者接收用户消息的时候，平台都会带上前面五个参数(timestamp、

nonce、token、open_id、sso_key）访问开发者设置的 URL，开发者通过接口对签名的效验判断此条消息的真实性。

同时，开发者可以获取到 sso_key，调用 api 获取到用户在平台中的详细信息

举例：回调地址是个 servlet，servlet 中如何验证用户推送过来的消息的合法性

```
/*
http://192.168.100.23:8080/oa/callbackServlet
?timestamp=1395059758
&nonce=919754
&token=517ffec7dd1d957a265c58fc08a9875f%3At82Dy1WsSK27FssSG0FWmfAHz9M
%3D
&open_id=1d3961ac6c9a0166223fbc0abae6c35bce98d5c0c6519a672387058d2411
642cf44c6172068b623408b6da01b3883509
&sso_key=zhangsan@partner.com

举例：request.getInputStream 可以获取到 post 来的流信息，其中
FromUserName=open_id
提醒：request.getParameter 获取数据后，再使用 request.getInputStream 就不能获
取到数据了。
<xml><fromusername><![CDATA[b76ce9b53edbb93fc7beee163aa1e6390a74cbc9e
9a429e7865d9298921d2184f941d8dba69e0cf74f88aa69d5d73e8a]]></fromusern
ame><ssokeyvalue>demo81@dehui.com</ssokeyvalue><createtime>1406532781
</createtime><msgtype><![CDATA[event]]></msgtype><event><![CDATA[unsu
bscribe]]></event></xml>

*/
public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
    String xml_open_id="";
    InputStream in=request.getInputStream();
    try {
        String s=Util.InputStreamTOString(in, "utf-8");
        //解析 xml 数据，获取到 FromUserName 就是 open_id，具体操作自行实现
        xml_open_id=***FromUserName***;
    } catch (Exception e) {
        e.printStackTrace();
```

```
    }
    OcuAccount oa=new OcuAccount();
    String open_id = oa.checkSignature(request);
    if(open_id.equals(xml_open_id)){
        //请求合法,并能获取到用户的信息
        User user=oa.getUserInfo(open_id);
        System.out.println(user);
    }else{
        //请求非法
    }
}
```

2.3.2 接受普通消息

当普通敏行用户向公众账号发消息时，平台服务器将 POST 消息的 XML 数据包到填写的公众号的回调地址 URL 上。各消息类型的推送 XML 数据包结构如下。

```
<xml>
<fromusername><![CDATA[63d675a2438861387578ab2f2939528357181473ff
f741e9c74f14bcda92fa74b0b5b09ba51698e77d0372188e73ff5b]]></fromuserna
me>
<ssokeyvalue>zhangsan@partner.com</ssokeyvalue>
<createtime>1396517316</createtime>
<msgtype><![CDATA[text]]></msgtype>
<content><![CDATA[this is a test]]></content>
<msgid>8822</msgid>
</xml>
```

内容说明

参数	描述
FromUserName	发送方帐号（一个 OpenID）
SSOKeyValue	平台与系统对应的用于 sso 验证的用户信息

CreateTime	消息创建时间（整型）
MsgType	text
Content	文本消息内容
MsgId	消息 id

2.3.3 接受事件推送

1. 自定义菜单事件

```
<xml>
<fromusername><![CDATA[aaba7680cf55a42aab18142ec9c8a612a225e8d6b020a486efd1ef884ab7e3f5d699caca3018461b5f41fa5cd4332bc8]]></fromusername>
<ssokeyvalue>zhangsan@partner.com</ssokeyvalue>
<createtime>1396426778</createtime>
<msgtype><![CDATA[event]]></msgtype>
<event><![CDATA[click]]></event>
<eventkey><![CDATA[menu1]]></eventkey>
</xml>
```

参数	描述
FromUserName	发送方帐号（一个 OpenID）
SSOKeyValue	平台与系统对应的用于 sso 验证的用户信息
CreateTime	消息创建时间（整型）
MsgType	消息类型，event
Event	事件类型，click
EventKey	事件 KEY 值，与自定义菜单中菜单项对应

2. 订阅/取消订阅事件

用户在订阅与取消订阅公众号时，平台会把这个事件推送到公众号中填写的回调地址 URL。方便开发者给用户下发欢迎消息或者做公众号的解绑。

```
<xml>
<fromusername><![CDATA[fa2f227beee0821d4e239308936af32df6cc9e3c1c753daba3622304a9ed774c013339047ad7e332ba3e0b3ecef928fb]]></fromusername>
<ssokeyvalue>zhangsan@partner.com</ssokeyvalue>
<createtime>1396528573</createtime>
<msgtype><![CDATA[event]]></msgtype>
<event><![CDATA[subscribe]]></event>
</xml>
```

参数	描述
FromUserName	发送方帐号（一个 OpenID）
SSOKeyValue	平台与系统对应的用于 sso 验证的用户信息
CreateTime	消息创建时间（整型）
MsgType	消息类型，event
Event	事件类型，subscribe(订阅)、unsubscribe(取消订阅)

2.3.4 获取接受消息

举例说明获取用户的推送消息或者事件的 xml 内容

回调地址是个 servlet，如下获取到用户的推送消息或者事件推送的 xml 内

容

```
   public void doGet(HttpServletRequest request, HttpServletResponse
response)
                    throws ServletException, IOException {
            InputStream in=request.getInputStream();
            try {
                    String s=Util.InputStreamTOString(in,  "utf-8");
                    System.out.println("获取到的用户推送消息或事件推
送的具体内容："+s);
            }catch(Exception e){
            }
   }
   public static String InputStreamTOString(InputStream in,String
encoding)  throws Exception{
            ByteArrayOutputStream outStream =  new
ByteArrayOutputStream();
            byte[] data =  new byte[1024];
            int count = -1;
            while((count = in.read(data,0,1024)) != -1)
                    outStream.write(data,  0,  count);
            data =  null;
            return new String(outStream.toByteArray(),"utf-8");
   }
```

2.4 发送消息

2.4.1 发送被动响应消息

对于每一个 POST 请求，开发者在响应包中返回特定内容，对该消息进行响应（现支持回复文本、图文）。

1. 回复订阅消息

业务系统根据实际情况返回订阅是否成功。

```
   public void doGet(HttpServletRequest request, HttpServletResponse
response)
               throws ServletException, IOException {
           InputStream in=request.getInputStream();
           FromPostBean frompostbean=Util.getFromPostBean(in);
           String eventkey=frompostbean.Event;
           String message="";
           try {
               if ("subscribe".equals(eventkey)){
                   boolean success = true;
                   if(success)
                       message="{\"errcode\":0,\"errmsg\":\"订阅成功
\"}";
                   else
                       message="{\"errcode\":1,\"errmsg\":\"系统维护中
\"}";
                   response.setContentType("application/json");
                   response.setCharacterEncoding("utf-8");
                   response.getWriter().write(message);
                   response.getWriter().flush();
```

```
                response.getWriter().close();
            }
        }catch(Exception e){

        }

    }
```

2. 回复文本消息

```
   public void doGet(HttpServletRequest request, HttpServletResponse
response)
                    throws ServletException, IOException {
            InputStream in=request.getInputStream();
            try {
                    String s=Util.InputStreamTOString(in,  "utf-8");
                    System.out.println("获取到的用户推送消息或事件推
送的具体内容："+s);
            }catch(Exception e){
            }
            //根据用户推送来的信息和实际的业务，回复用户文本消息
            String content=  "测试用的，回复的文本消息内容";
            response.setContentType("application/html");
            response.getWriter().write(content);
            response.getWriter().flush();
            response.getWriter().close();
   }
```

3. 回复单图文消息

单图文消息结构

```
{
  "article_count": 1,
  "articles": [
    {
      "title": "title1",
      "description": "description1",
      "pic_url": "http://www.test.com/img/logo.png",
      "app_url": "",
      "url": "http://www.baidu.com"
    }
  ]
}
```

servlet 返回响应单图文消息

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    InputStream in=request.getInputStream();
    try {
        String s=Util.InputStreamTOString(in,"utf-8");
        System.out.println("获取到的用户推送消息或事件推送的具体内容："+s);
    }catch(Exception e){
    }
    //根据用户推送来的信息和实际的业务，回复用户文本消息。Content 内容必须是正确格式的 JSON 字符串。样例是单图文，也支持多图文。修改 JSON 字符串即可。
    String content="{\"article_count\":1,\"articles\":[{\"title\":\"title\",\"description\":\"description\",\"pic_url\":\"http://www.test.com/img/logo.png\",\"app_url\":\"\",\"url\":\"http://www.baidu.com\"}]}";
    response.setContentType("application/json");
    response.getWriter().write(content);
    response.getWriter().flush();
    response.getWriter().close();
}
```

4. 回复多图文消息

多图文消息结构，servlet 代码参考单图文的。

```
{
  "article_count": 4,
  "articles": [
    {
      "title": "title1",
      "description": "description1",
      "pic_url": "http://www.test.com/img/logo.png",
      "app_url": "",
      "url": "http://www.baidu.com"
    },{
      "title": "title2",
      "description": "description2",
      "pic_url": "http://www.test.com/img/logo.png",
      "app_url": "",
      "url": "http://www.baidu.com"
    },{
      "title": "title3",
      "description": "description3",
      "pic_url": "http://www.test.com/img/logo.png",
      "app_url": "",
      "url": "http://www.baidu.com"
    },{
      "title": "title4",
      "description": "description4",
      "pic_url": "http://www.test.com/img/logo.png",
      "app_url": "",
      "url": "http://www.baidu.com"
    }
  ]
}
```

2.4.2 发送主动消息

需要主动给平台推送消息的时候，可以直接调用 api 发送消息（文本、图文）到平台，把消息推送给订阅了该公众号的用户。

比如 OA 系统需要主动发送待办信息给敏行用户时，OA 系统可以调用消息接口发送消息给特定用户。

比如新闻系统需要主动发送新闻给订阅新闻公众号的用户时，新闻系统可以调用消息接口发送消息给订阅的用户。

1. 发送文本消息

```
public void textToUser(String content,boolean ispub){
                    TextMessage tm =  new TextMessage(content);
                    List<String> ids =  new ArrayList<String>();
                    ids.add(user);
                    OcuAccount   oa   =    new OcuAccount();
                    oa.setRootUrl("http://minxing.com"); //敏行平台
地址
                    oa.setApiPrefix("/api/v1");
                    oa.setOcuId("6d803e3e020e956c263af17298bf0903")
;
                    oa.setOcuSecret("884a7f22f2d2b9f68bed557a8cbd17
d1");
                    if(ispub){  //给订阅该公众号的所有人发送消息
                            oa.sendMessageToPublic(tm);
                    }else{          //给订阅该公众号的某个人发送消息
                            oa.sendMessageToUsers(tm,    ids);
                    }
            }
```

2. 发送图文消息

```
public void multiarticleToUsers(Article[] pt,boolean ispub){
                ArticleMessage am=new ArticleMessage();
```

```
                for(int i=0;i<4;i++){
                        if(pt[i]!=null){
                                am.addArticle(pt[i]);
                        }
                }
                List<String> ids =  new ArrayList<String>();
                ids.add(user);
                OcuAccount oa=new OcuAccount();
                oa.setRootUrl("http://minxing.com"); //敏行平台地址
                oa.setApiPrefix("/api/v1");
                oa.setOcuId("6d803e3e020e956c263af17298bf0903");
                oa.setOcuSecret("884a7f22f2d2b9f68bed557a8cbd17d1");
                if(ispub){  //给订阅该公众号的所有人发送消息
                        oa.sendMessageToPublic(am);
                }else{          //给订阅该公众号的某个人发送消息
                        oa.sendMessageToUsers(am, ids);
                }
        }
```

图文消息分类

图片建议尺寸：720 像素 * 400 像素

图文消息分两类，分别对应不同的构建方式:

1. 点击消息打开的时候，直接打开 url 对应的链接地址

参数	描述
title	消息标题
imageUrl	http://**图片地址**
url	http://**点击消息打开的地址**

description	消息的描述
appUrl	默认为空，需要打开对方 app 时才填写，格式为 para1=name&para2=title，本质就是打开第三方 APP，同时传参数过去

```
private Article[] parseArticle(HttpServletRequest request){
        Article[] articles = new Article[4];
        for(int i=1;i<5;i++){
            if(request.getParameter("title"+i)!=null){
                String title = request.getParameter("title"+i);
                String description =
request.getParameter("description"+i);
                String picUrl = request.getParameter("picUrl"+i);
                String url = request.getParameter("url"+i);
                String appUrl = request.getParameter("appUrl"+i);
                Article a = new
Article(title,description,picUrl,url,appUrl);
                articles[i-1]=a;
            }
        }
        return articles;

    }
```

2. 该类消息内容存储在敏行中，点击直接打开该消息

参数	描述
title	消息标题
subtitle	副标题
author	作者
createTime	创建时间
picUrl	http://**图片地址**
content	消息内容

3 Web 方式嵌入

在企业应用的 web 页面中嵌入平台提供的 javascript 代码即可快速为企业应用增加企业社交功能。

3.1 代码引入

在需要整合的网页中嵌入以下代码，推荐将此 JS 在页面<head>区域引用

<script src="http://敏行地址 ip:port/connect/mx_loader.js"
data-network="yuyi-info.com"></script>

`data-network` 是网络地址，可以从网站首页的 URL 中获取

在需要显示消息流的位置粘贴以下代码

```
<div id="feed-container"></div>
```

`feed-container` 这是消息流在页面中显示的容器，可以随意指定 id，需要与 API 调用指定的 ID 一致。

3.2 悬浮 widget 方式

在应用系统中得页面中嵌入如图的悬浮框，点击悬浮框弹出层可以实现即时通讯功能


```
//所有关于 openGrap 的请求，都需要在该回调中运行
MX.onGraphLoad({
    objectProperties: {
        url: "https://www.news.com/file/abc123",
        type: "file",
        title: "news.pdf",
        image: "https://www.news.com/file/abc123"
    },
    ocuId: "xxxxxxxxx"
},
function(mc) {
    //悬浮敏行 widget
```

```
    mc.embedGraph({
        container: {
            placement: "right",
            style: "/connect/images/chat.png"
        },
        height: "400px",
        type: "conversations_graph",
        config: {
            promptText: "跟你同事说点啥"
        }
    });
});
```

属性名	类型	默认值	描述
objectProperties	Object	空	这个属性如果不填写，会默认从网页上抓取信息，发送到敏行服务器，如果配置了信息，会按照配置的信息进行发送
ocuId	Object	空	这是在敏行平台注册的公众号的开发 ID，用于链接访问时进行签名校验
container	ID 或者 Widget 配置	feed_container	这个是用来指定消息流显示的位置,必须是页面中唯一的 ID，如果是做成悬浮的 widget，需要配置 placement 和 style，placement 默认是 right，表示右漂浮，style 用来设置悬浮的图片
height	int	auto	这个是用来指定消息流容器的高度,默认是自动
width	int	100%	这个是用来指定消息流容器的宽度,默认是与父容器（feed_container）的宽度一致
type	string	'graph'	设定 graph 的的类型，只是两个值 graph 和 conversation_graph
config.promptText	string	'想说点什么呢'	用来指定输入框中的提示文字
conf.miniApps	array	ALL	指定默认显示哪些 miniApp 的输入器，默认是显示所有，如果指定["update"]就只显示普通消息输入框
conf.showFeedFilter	boolean	false	是否显示消息流类型的过滤，详见下面图解
conf.showAppFilters	boolean	false	是否显示 miniApp 的类型过滤

3.3 与某人即时通讯

在应用的页面上直接跟人员进行即时通讯，及时讨论相关事宜。


```
<span data-mxim="liuj@mx.com"></span>
    MX.onGraphLoad(function(mc){
         mc.initIM({
            ssoKey:"email",
            width:"300px",
            height:"400px"
         })
     })
```

属性名	类型	默认值	描述
data-mxim	String	无	这个是用来指定即时通讯中的聊天对象，一般会根据 ssoKey 决定使用什么类型的属性，默认是对方的邮箱
height	px	400px	这个是用来指定容器的高度
width	px	300px	这个是用来指定容器的宽度
ssoKey	string	email	指定当前应用与敏行平台 sso 的字段类型，如邮箱，Ldap 中的 loginName

3.4 嵌入 iframe 方式

3.5 显示消息流

可以在应用的页面上显示消息流

嵌入以下 JS 代码即可

```
MX.embedFeed({
    container:"feed_container",
    type:"following",
    //feedId:"1", 当 type 为 group、user、topic 的时候需要指定 feedId，具体获取方法见上图示
    config:{
            promptText: "跟这个说点什么吧",
            miniApps:["update"],
            showFeedFilter:true,
            showAppFilter:true
    }
});
```

属性名	类型	默认值	描述
container	ID	feed_container	这个是用来指定消息流显示的位置,必须是页面中唯一的 ID
height	int	auto	这个是用来指定消息流容器的高度,默认是自动
width	int	100%	这个是用来指定消息流容器的高度,默认是自动 width int 100% 这个是用来指定消息流容器的宽度,默认是与父容器（feed_container）的宽度一致
type	string	following	指定消息流的类型，候选值有following、my_all、created_by_me、all、conversations、group、user、topic，当为 group、user、topic 时需要指定对应的 feedId
config.promptText	string	'想说点什么呢'	用来指定输入框中的提示文字
conf.miniApps	array	ALL	指定默认显示哪些 miniApp 的输入器，默认是显示所有，如果指定["update"]就只显示普通消息输入框
conf.showFeedFilter	boolean	false	是否显示消息流类型的过滤，详见下面图解
conf.showAppFilters	boolean	false	是否显示 miniApp 的类型过滤

3.6 事件

类型	描述

fresh_message	显示消息流时，如果该消息流有新的消息到达，会触发该事件，并传递当前消息流中的未读消息数

```
MX.on("fresh_message",function(response){
   response=={
        data:{
            following_feed_unseen_count: 0 //我的关注未读数量
            unseen_private_messages_count: 23 //即时通讯未读数量
        },
        feedId: 1    //feedId 用于回调判断使用
        type: "graph" //嵌入的类型,用于回调判断使用
    }

})
```

3.7 页面信息抓取的规则

如果调用 onGraphLoad 时没有传递 objectProperties,会自动从网页抓取信息，优先从网页的以下字段进行抓取，如果抓取不到再获取网页的 url 和标题。

```
〈head〉
...
〈meta property="og:type" content="file"〉
〈meta property="og:title" content="news.pdf"〉
〈meta property="og:image" content="https://www.news.com/file/abc123.png"〉
〈meta property="og:description" content="这是这个文件的描述，你要看看哟。"〉
...
〈/head〉
```

参看 http://ogp.me/了解详细 Open Graph 的信息.

4 native 方式集成

提供多种方式与已有 Native 应用整合。

详见文档《敏行平台 Native 方式集成企业应用手册.doc》