|  |
| --- |
| **XySoft Group** |
| **Tigase开发扩展文档** |
| v 0.1 |
|  |
| **XySoft Group** |
| **2018/4/19** |

密级：合作企业可见

|  |
| --- |
| [在此处键入文档的摘要。摘要通常是对文档内容的简短总结。在此处键入文档的摘要。摘要通常是对文档内容的简短总结。] |

目录

[统一开发规范约束 3](#_Toc513815207)

[1单聊 3](#_Toc513815208)

[1.1消息体 3](#_Toc513815209)

[1.3消息监听 3](#_Toc513815210)

[1.3扩展消息 5](#_Toc513815211)

[2群聊 5](#_Toc513815212)

[2.1建群 5](#_Toc513815213)

[2.2加入群 5](#_Toc513815214)

[2.3 邀请人进群 7](#_Toc513815215)

[2.4 群成员角色 7](#_Toc513815216)

[2.5 群消息 7](#_Toc513815217)

[2.6 离线消息处理 7](#_Toc513815218)

[2.7 删除群成员 7](#_Toc513815219)

[2.8 自己退群处理 7](#_Toc513815220)

[2.9 本地持久化需要注意的 7](#_Toc513815221)

[3文件传输 8](#_Toc513815222)

[3.1 si文件协商 8](#_Toc513815223)

[3.2 ibb支持 8](#_Toc513815224)

[3.3 Socket5支持 8](#_Toc513815225)

[3.4 HTTP门户文件存储（离线小文件） 8](#_Toc513815226)

[3.5 私有云分布式文件系统（离线文件） 8](#_Toc513815227)

[4组织架构 8](#_Toc513815228)

[4.1 提供同步接口供用户主动同步调用 8](#_Toc513815229)

[4.2 用户不能控制OA系统下的同步方案 9](#_Toc513815230)

[4.3 服务器到客户端的同步 9](#_Toc513815231)

[5认证 9](#_Toc513815232)

[5.1 XMPP认证 9](#_Toc513815233)

[5.2 用户自定义认证（需开发服务器插件） 9](#_Toc513815234)

[5.3 LDAP认证 9](#_Toc513815235)

[6 Web API接口 9](#_Toc513815236)

# 统一开发规范约束

## 1单聊

### 1.1消息体

<message

to="wangxin@win7-1803071731/pc"

from="test1@win7-1803071731/Spark"

id="qZFn5-8533"

**type="chat"**>

<body>ok，我收到了</body><request xmlns="urn:xmpp:receipts"/>

<active xmlns="http://jabber.org/protocol/chatstates"/>

</message>

**必须项：**

1. 发送消息类型必须为chat
2. 消息必须设定类型，不得为空

**可选项：**

1. 扩展属性可以酌情支持，比如保障消息可靠性确认的回执特性：<request xmlns="urn:xmpp:receipts"/>.
2. 消息状态，正在书写: <active xmlns="http://jabber.org/protocol/chatstates"/>

一般用于PC版。

1. 消息状态，停止书写: <paused xmlns="http://jabber.org/protocol/chatstates"/>

### 1.3消息监听

**必须项：**

1. 以下消息类型必须监听处理：
   * + 1. ***chat***
       2. ***normal***
       3. ***headline***
       4. ***groupchat***
       5. ***error***
2. 除特殊的情况，发送单聊消息不得使用类型为normal的消息类型
3. 消息在客户端持久化，消息id必须是来自xmpp消息体的id，如：id="qZFn5-8533"

例子：

StanzaListener listenerMsg = **new** StanzaListener() {

@Override

**public** **void** processStanza(Stanza stanza)

**throws** NotConnectedException, InterruptedException, NotLoggedInException {

**if** (stanza **instanceof** Message) {// 消息包

Message message = (Message) stanza;

**if** (message.getType() == Message.Type.***chat***) {

//单聊

**if** (message.getBody() == **null**) {

//抛弃该消息

} **else** {

ChatService.*recivePacket*(message);

}

}

**if** (message.getType() == Message.Type.***headline***) {

//重要消息

}

**if** (message.getType() == Message.Type.***normal***) {

// 可能是普通消息，也可能是回执消息，也可能是扩展消息

**if** (message.hasExtension("urn:xmpp:receipts")) {

// 处理回执消息

ChatService.*receiptArrived*(message);

} **else** **if** (message.hasExtension("urn:xmpp:attention:0")) {

// 震动提醒消息

} **else** **if** (message.hasExtension("http://jabber.org/protocol/muc#user")) {

// 被邀请加入群聊

//Smack中还可使用注册 InvitationListener进[XMPPConnection](eclipse-javadoc:%E2%98%82=XyTalk.pc/D:%5C/0NewIMRespo%5C/smack_4_2_4%5C/libs%5C/smack-extensions-4.2.4.jar%3Corg.jivesoftware.smackx.muc(MultiUserChatManager.class%E2%98%83MultiUserChatManager~getInstanceFor~Lorg.jivesoftware.smack.XMPPConnection;%E2%98%82org.jivesoftware.smack.XMPPConnection" \o "in org.jivesoftware.smack)来监听群聊邀请，其原理也是处理muc#user扩展消息

} **else** **if** (message.hasExtension("jabber:x:conference")) {

// 被邀请加入群聊的另一种形式

} **else** {

**if** (message.getBody() == **null**) {

//抛弃该消息

} **else** {

//等同于chat类型消息处理

ChatService.*recivePacket*(message);

}

}

}

**if** (message.getType() == Message.Type.***groupchat***) {

// 群聊

MucChatService.*recivePacket*(message);

}

**if** (message.getType() == Message.Type.***error***) {

// 错误信息

ErrorMsgService.*recivePacket*(message);

}

}

}

};

### 1.3扩展消息

## 2群聊

### 2.1建群

### 2.2加入群

Xmpp体系下的MUC Room，默认必须由用户主动进入，才开始群通讯，用户在进入前，Room里的任何群聊消息是不可监听的，其原理是MUC实际是一个发布订阅体系，一个Room就是一个Node，用户加入群（或者说进入群），就代表订阅监听了Node上（即该Room）的消息（注意：订阅后，不仅仅是其他群成员发送的消息，自己发送的消息，服务器一样也会发回，需要客户端去重）。

Xmpp体系下的MUC有以下几个问题：

1. 登陆时无未读新消息感知。用户下线后，再次登陆，也不能感知曾经进过的群里任何离线消息（因为没有进入群，即没有订阅）；
2. 在线时也可能没有未读新消息感知（除非进入群）。比如原来群里有A、B、C，用户A退出群（离开）之后，这个Room里的成员就没有A了，BC之间能收到群消息，只有A再次主动进入群，或者被BC邀请进入群，才能收到离开这段时间的离线消息；

所以如果要实现类似微信的群聊，即随时可感知未读消息，可以有3种方式可以选择

1. 登陆的时候就订阅（进入）所有自己相关的群，不再退出，除非被群主剔除；
2. 群里有人发送消息时，才向群成员发起进群邀请。比如群成员有A、B、C，
   1. 第一轮，A进入群，发了一条Groupchat Message1消息，则接着查询群成员中谁没加入，查询发现B、C没有进入，则向B、C各发送一条邀请消息Invate Message1，假如这时B在线，C关机离线，则B受邀进群，发给C的Invate Message1则被服务器离线存储。
   2. 第二轮，A又发了一条Groupchat Message2消息，按设计逻辑，又要再次查询谁没进入群，发现C没有进入，又发现几分钟前曾经发起过对C的邀请，这次就不发邀请了。
   3. 第三轮，B退出了群，A又发了一条Groupchat Message3消息，又要再次查询谁没进入群，发现B、C没有进入，发现其中B曾经进过群，后来又离开了，则对B发出邀请消息Invate Message2，发现几分钟前曾经发起过对C的邀请，这次也不发邀请给C了
   4. ……无穷尽
3. 由服务器来处理收到群消息后的邀请加群，逻辑将更加复杂棘手

综上所述，方案2、3异常复杂，处理逻辑容易出错。

所以得出，方案1是最优选。

#### 必须项：

1. 用户登陆后必须进入自己相关的群；
2. 除非被群主剔除，不得离开群；
3. 以“username-真实姓名”为nickname进入群

#### 需要延伸考虑的项：

1. 如果用户有几百个群，登陆时都订阅需要占用多少堆内存？需要用jvm监控工具来测试每个群的内存占用。
2. 如果不全部订阅，只选择最常用的群订阅，那其余的群不能即时获取最新消息，怎么处理？

#### 登陆后订阅所有群的例子：

*connection*.login();

…..

// **TODO** 登陆后订阅全部房间

//在sqlite库中查询MUC群组

List<DbRoom> dbMucRooms = *roomDbService*.findMucRooms();

**for** (DbRoom dbRoom : dbMucRooms) {

**if** (dbRoom!=**null** &&

! dbRoom.getRoomId().isEmpty() &&

dbRoom.getRoomId().contains("@") ) {

MultiUserChat mucRoom = MultiUserChatManager.*getInstanceFor*(*connection*).getMultiUserChat(JidCreate.*entityBareFrom*(dbRoom.getRoomId()));

//将room加入堆缓存，以便重复利用

*roomCacheService*.put(dbRoom.getRoomId(),mucRoom);

//以“username-真实姓名”为nickname进入群，完成订阅

mucRoom.join(Resourcepart.*from*(UserCache.*CurrentUserName* + "-" + UserCache.*CurrentUserRealName*));

}

}

#### 进群并定制接收离线消息的例子：

requestMaxStanzasHistory可以用来设置进入群之后获取最近的几条离线消息。

MucEnterConfiguration.Builder builder = room.getEnterConfigurationBuilder(Resourcepart.*from*(UserCache.*CurrentUserName* + "-" + UserCache.*CurrentUserRealName*));

**//只获取最后10条历史记录**

builder.requestMaxStanzasHistory(10);

//只获取最后1000秒以前的离线消息

builder.requestHistorySince(1000);

//只获取2018-5-1以来的历史记录

builder.requestHistorySince(**new** Date(2018,5,1));

MucEnterConfiguration mucEnterConfiguration = builder.build();

room.join(mucEnterConfiguration);

### 2.3 邀请人进群

### 2.4 群成员角色

### 2.5 群消息

### 2.6 离线消息处理

### 2.7 删除群成员

### 2.8 自己退群处理

### 2.9 本地持久化需要注意的

## 3文件传输

### 3.1 si文件协商

### 3.2 ibb支持

### 3.3 Socket5支持

### 3.4 HTTP门户文件存储（离线小文件）

### 3.5 私有云分布式文件系统（离线文件）

## 4组织架构

### 4.1 提供同步接口供用户主动同步调用

### 4.2 用户不能控制OA系统下的同步方案

### 4.3 服务器到客户端的同步

## 5认证

### 5.1 XMPP认证

### 5.2 用户自定义认证（需开发服务器插件）

### 5.3 LDAP认证

## 6 Web API接口

# Tigase XMPP插件开发