Atitit xmpp协议概要

目录

[1. Xep XMPP扩展协议 1](#_Toc19215)

[1.1. XEP-0295：XMPP的JSON编码 1](#_Toc7533)

[2. XMPP节： 1](#_Toc28337)

[2.1. 核心节：<message><presene><iq> 1](#_Toc31996)

[2.2. 节属性： 2](#_Toc31297)

[2.3. 支持通用属性：from，to，type，id。 2](#_Toc16052)

[3. 核心节： 3](#_Toc24203)

[3.1. 1）、presence 3](#_Toc4514)

[3.2. 2）、message 4](#_Toc29172)

[3.3. ）、iq info/query（信息与查询），他给xmpp童心提供请求和响应机制 4](#_Toc4074)

[3.4. 、error在错误提示节中， 5](#_Toc13880)

[4. xmpp连接： 6](#_Toc19438)

[4.1. 连接，流的建立，身份验证以及断开连接。 6](#_Toc5473)

# Xep [XMPP](https://en.wikipedia.org/wiki/XMPP" \o ")扩展协议

* [XMPP](https://en.wikipedia.org/wiki/XMPP" \o "XMPP) Extension Protocol, in the XMPP Standards Foundation
* [XMPP](https://en.wikipedia.org/wiki/XMPP" \o ")扩展协议，在XMPP标准基金会

## XEP-0295：XMPP的JSON编码

# ****XMPP节：****

## 核心节：<message><presene><iq>

1、<stream>  xmpp以其标记开始。  
2、<presence>用来表明用户状态。可以来广播或“发布-订阅”；<status>网络状态、<show>可用状态、<priority>资源优先级。  
3、<iq>一种请求/响应机制。  
4、<message>用于两个jabber用户发送消息。<body>是消息内容；他包含<subject/>（消息标题）<body/>（消息内容）<thread/>(跟踪会话线索)子标签。

简单会话：  
<stream:stream>    创建一个xmpp流  
<iq type=’get’>    请求elizabeth的花名册，也就是她的所有已存储的联系人列表。  
<query xmlns=’jabber:iq:roster’/>  
</iq>  
<presence/>             通知服务器她已经在线并可以访问。  当他注意到darcy在线的时候，就会发个message节，  
<message to=’darcy@pemberley.lib’  
from=’ellizabaeth@longbourn.lit/ballroom’  
type=’chat’>  
<body>sdf</body>  
</message>  
<presence type=’unavailable’/>  告诉服务器，这是不可访问的要关闭  
</stream:stream>

## 节属性：

## 支持通用属性：from，to，type，id。

1）、from  
识别此节的起始jid。这里不建议在输出的节上手工设置from属性，服务器会在这些节通过时添加正确的from属性，而如果错误地设置from属性会导致服务器拒绝整个节。  
客户端-服务器中，接收到的节上没有from属性，意味着该节来源于服务器自身，而在服务器-服务器中，缺少from被视为是错误。  
2）、to  
把xml节发送到to属性指定的jid。他和from相似。  
客户端-服务器流中没有to属性，那么服务器将假设它是有意发给服务器自身的消息，建议在向服务器自身发消息时忽略to。  
如果to属性中指定的jid是一个用户，那么服务器有可能代表用户来处理该节。  
如果目的地是一个裸jid，那么服务器将处理这个节。  
如果目的地是一个完整的jid，那么服务器将直接把该节路由到该用户。  
3）、type  
这个属性指定了三个节的具体类型。  
三个节都可以把type设置为error，表示这个节是对已接收到的同一类型的节的错误影响，不要响应类型为error的节，避免在网络上出线反馈环节。  
4）、id  
给节指定id来辅助识别响应。对于iq节，id是必须的，但是对于其他的节id是可选的。如果某个节是为了响应一个携带id属性的节而产生的，那么这个应答必须包含携带相同值得id值。  
id必须具有唯一性，这样节的发送者就可以使用它来甄别响应了。最简单的做法就是让id属性值在给定的流中保持唯一性，以免歧义。  
在message和presence节的应答节一般仅限于报告错误，iq的应答节可以用来通知成功操作、确认命令或返回请求的数据。无论如何客户端都可以使用应答节的id属性来识别与该节相关联的请求节。   在短时间内发送大量同类型的节，此时这个功能就非常关键了。因为这些节的应答可能会以乱序形式到达。

# 核心节：

## 1）、presence

他控制并报告实体的可访问性。(包括：在线，离线；  离开，请勿打扰等。  ) 他还可以用来建立和终止向其他实体发布出席订阅。  
有了出席订阅通知，我们在即时通信系统中就可以在发送消息之前知道接收者是否在线。  
（1）、普通的presence  
此时他是不包含type属性的，如果有的话他的属性值为unavailable活着error。  在这里，type属性是没有available值得，因为我们可以通过缺省的type来设定。  
用户可以通过发送不携带to属性，并直接发往服务器的presence节来操纵出席状态。如下：  
<presence/>      前两个节将用户的出息状态设置为在线和离线  
<presence type=‘unavailable’/>   离线  
<presence>  
<show>away</show>      只能用在presence中，值away离开，chat聊天，end不希望被打扰，xa长期离开，  传达用户可访问性的性质，它请求接收者的客户端使用这个消息来更新发送者出息状态的可视化指示器。  
<status>sf</status>   一个人们可读的字符串，用户可以将其设置为能够传达出席信息的任何值。  聊天时，接收者的客户端中这个字符串一般紧挨着联系人名字显示。  
</presence>  
<presence>  
<status>sfdf</status>  
<priority>22</priority>    优先级  同时具有多个回话的用户可以使用优先级来指出哪一个资源应该接收到那些发往该用户裸jid的聊天消息。  
</presence>  
<presence>  
<prioriry>11</pritority>  
</presence>

（2）、扩展presence  
比如将你正在听歌，心情等信息广播给其他联系人。这里用流量较大。

（3）、出席订阅  
server会自动将出席信息广播给联系人，用户也会从所有他已经进行出席订阅的联系人那里接收到出席的状态更新。  
对于好友状态的订阅：A订阅了B得出席信息，这里并不是说B也订阅了A得出席信息。  
订阅状态：subscribe（建立订阅），unsubscribe（取消订阅），subscribed（应答建立），unsubscribed（应答取消）。  
AB订阅：  
<presence from=‘a@longbourn.lit/outside’ to=‘b@pemberley.lit’ type=‘subscribe’/>  
<presence from=‘b@pemberley.lit/library’ to=‘a@longbourn.lit/outside’ type=‘subscribed’/>  
<presence from=‘b@pemberley.lit/library’ to=‘a@longbourn.lit’ type=‘subscribe’/>  
<presence from=‘a@longbourn.lit/outside’ to=‘b@pemberley.lit/library’ type=‘subscribed’/>

（4）、定向出席  
他是一种直接发给另一个用户或其他某个实体的普通presence节。这里可以用来向那些没有进行出席订阅的实体传送出席状态信息。也就是临时出席。  
这里当发送者变成不可访问的状态的时候，出席信息的接收者将会自动得到通知，即使发送者忘记显示的通知接收者。这里可以使用定向出席来临时地掌握某个用户的可访问性。

## 2）、message

从一端到另一端发送消息。这个节属于发出后不再过问，可靠性不能保证。一旦消息发出去了，发送者就不会知道他是不是传送出去了，也不会知道到达的时间。  
message示例：  
单人聊天  
<message from=‘a@netherfield.lit/frawing\_room’ to=‘b@pemberley.lit’ type=‘chat’>  
<body>adf</body>  
</message>  
多人聊天  
benlent夫人向聊天室bennets@chat.merython.lit发送消息，bennet接收消息  
<message from=’bennets@chat.merython.lit/mrs.bennet’ to=‘mr.bennet@longbourn.lit/study’ type=‘groupchat’>  
<body>df</body>  
</message>

（1）、消息类型  
type指出：chat(一对一聊天)，error，normal，group chat(多人聊天)，headline(不支持或者不方便应答的自动化服务使用)。  
type是可以省略的，默认为normal，但是我们应该提供一个type值，

（2）、消息内容  
<body>元素包含着该消息中人们的可读的内容。只要有不同的xml:lang属性就可以使用多个<body>。  
<thread>元素用来创建线索，他的内容是一个用来区分不同线索的唯一标识符。应答节应该包含与所应答的节相同的<thread>元素。 （线索：向电子邮件一样）  
在消息中，可以使用XHTML-IM来给消息提供格式化，超链接，以及富媒体。也可以使用Chat State Notifications来允许用户通告对方自己正在撰写消息或有空。（qq的正在输入）

3

## ）、iq info/query（信息与查询），他给xmpp童心提供请求和响应机制

这个节表示的时info/query（信息与查询），他给xmpp童心提供请求和响应机制。和http的基本工作原理想死，允许获取和设置查询。（get 和 post）  
每个节都需要有一个响应，但是这个节的必须的id将用来把响应与导致该响应的请求关联起来。  
type：get，set请求节； result，error为响应节。  
下面看示例，每个iq节必须匹配id属性  
a向服务器发送了一个格式错误的花名册请求  
<iq from=a@longbourn.lit/garden’ type=‘get’ id=‘roster1’>  
<query cmlns=‘jabber:iq:roster’/>  
</iq>  
<iq to=‘a@longbourn.lit/garden’ type=‘error’ id=‘roster1’>  
<query xmlns=‘jabber:iq:roster’/>  
<error type=‘cancel’>  
<feature-not-implemented xmlns=‘urn:ietf:params:xml:ns:xmpp-stanzas’/>  
</error>  
</iq>

重新发送  
<iq from=a@longbourn.lit/garden’ type=‘get’ id=‘roster2’>  
<query cmlns=‘jabber:iq:roster’/>  
</iq>  
<iq to=‘a@longbourn.lit/garden’ type=‘result’ id=‘roster2’>  
<query xmlns=‘jabber:iq:roster’>  
<item jid=‘b@longbourn.lit’ name=‘b’/>  
<item jid=‘c@netherfield.lit’ name=‘c’/>  
</query>  
</iq>

a将b添加到花名册,回复空白iq来应答成功。  
<iq from=‘a@longbourn.lit/garden’ type=‘set’ id=‘roster3’>  
<query xmlns=‘jabber:iq:roster’>  
<item jid=“b@pemberley.lit’ name=‘b’/>  
</query>  
</iq>  
<iq to=‘a@longbourn.lit/garden’ type=‘result’ id=‘roster3’/>

## 、error在错误提示节中，

1. 必须将type设置为error，并且需要携带一个<error>子元素。  
   在error中，他的type类型为：cancel（不应该重试），continue（警告信息），modify（发送的数据需要一些修改才可以被接受），auth（通知实体在以某种方式进行身份验证之后重试），wait（报告server临时遇到问题，稍后重发）。  
   在<error>中，必须包含一个错误条件作为他的子元素，也可以在<error>元素的子元素中指定应用程序的特定错误条件。还可以包含一个<text>元素来进一步指出有关该错误的详细信息。

要注意的是每个错误条件元素都必须位于urn:ietf:params:xml:ns:xmpp-stanzas命名空间里。

<error type=‘cancel’>  
<not-allowed xmlns=‘urn:ietf:params:xml:ns:xmpp-stanzas’/>  这里指出了一般性故障  
<closed-node xmlns=‘http:/jabber.org/protocol/pubsub#errors’/>  给出了请缺德应用程序错误提示信息  
<text xmlns=‘urn:ietf:params:xml:ns:xmpp-stanzas>dfs</text>   包含了问题描述  
</error>

# ****xmpp连接：****

## 连接，流的建立，身份验证以及断开连接。

1）.连接  
发送节前要建立xmpp流，在流存在之前要建立通往xmpp服务器的连接。  
客户端和服务器利用域名系统将服务器的域名解析成一个能够连接的地址。电子邮件服务特别的使用邮件交换台记录来提供处理特定域邮件的服务器列表。这样一个知名服务器地址就不用处理每一项服务请求了。电子邮件在dns中有特殊的对待。现在服务记录srv可用来为任意服务提供类似的功能。  
当xmpp客户端或服务器连接到另一个xmpp服务器的时候，首先要在服务器域中查询适当的srv记录。查询应答中可能包含多条srv记录，这样就可以在多台服务器间建立负载均衡链接。  
当没有找到合适的srv记录的时候，应用程序试着直接连接到给定域。大多数库也可以显式制定要连接的服务器。  
2）.流的建立  
向服务器发送起始元素<stream:stream>，就可以打开xmpp流。服务器通过发送响应流的起始标记<stream:stream>进行响应。  
服务器先发送<stream:features>元素，详细列举xmpp流中支持的所有功能。这些功能大多数与可用的加密和身份验证选项有关。  
客户端发给服务器  
<?xml version=’1.0′?>  
<stream:stream xmlns=’jabber:client’  
xmlns:straem=’http://ethrx.jabber.org/streams’  
version=’1.0′  
to=’pemberley.lit’>  
服务器应答  
<?xml version=’1.0′?>  
<stream:stream xmlns=’jabber:client’  
xmlns:stream=’http://etherx.jabber.org/streams’  
version=’1.0′  
from=’pemberley.lit’  
id=’dddd’  
xml:lang=’en’>  
<stream:features>  
<stream:freatures>  
<starttls xmlns=’urn:ietf:params:xml:ns:xmpp-tls’/>  
<compression xmlns=’http://jabber.org/features/compress’>  
<method>zlib</method>  
</compression>  
<mechanisms xmlns=’urn:ietf:params:xml:ns:xmpp-sasl’>                     <mechanism>DIGEST-MD5</mechanism>  
<mechanism>PLAIN</mechanism>  
</mechanisms>  
</stream:features>

根据数据交换信息指导pemberley.org服务器支持TLS，流压缩zlib等。  
在两台服务器间建立xmpp流的过程相同，但是 jabber：client换成 jabber：server才可以。

3）.身份验证  
xmpp可以进行tls传输层安全加密，并且大多数客户端默认使用。服务器支持tls加密后，客户端就会启动tls链接，并将当前套接字升级为一个加密套接字。创建一对新的xmpp流。  
xmpp身份验证使用sasl 简单身份验证与安全层协议，并支持多种身份验证机制。服务器通提供明文验证和基于md5摘要的验证，但是某些服务器还支持通过kerberos或特殊令牌来验证。  
完成身份验证后，客户端必选为该链接绑定一个资源并启动一个会话。如果用户正在线路上查看xmpp流量，就会看到<bind><session>元素被发送出去。（在<iq>中）  如果客户端没有提供绑定的资源，那么服务器将为其选定一个。  
当两台服务器相互连接的时候，服务器交换并验证tls证书，或者接受服务器使用某种回拨协议通过dns来验证发送者身份。

4）.连接断开  
当用户结束xmpp会话的时候，他们终止会话并断开连接。最有礼貌的终止会话的方式是：先发送无效出席信息，然后关闭<stream:stream>元素。  
通过发送最后的无效出席信息，用户的联系人能够得知该用户离开的原因。  
<presence type=’unavailable’/>  
</stream:stream>  
然后，服务器终止发往客户端的流。