



中华人民共和国通信行业标准

YD/T 2936-2015

扩展消息与表示协议 (XMPP) 即时消息与表示

Extensible messaging and presence protocol (XMPP)
— Instant messaging and presence

2015-07-14 发布

2015-10-01 实施

中华人民共和国工业和信息化部 发布

目 次

前 言.....III

1 范围.....1

2 规范性引用文件.....1

3 缩略语.....1

4 需求.....2

5 功能概述.....2

6 名册管理.....3

6.1 概述.....3

6.2 语法和语义.....3

6.3 登录时接收一个人的名册.....7

6.4 增加一个名册条目.....8

6.5 更新名册条目.....13

6.6 删除一个名册条目.....15

6.7 名册版本.....16

7 管理表示订阅.....17

7.1 概述.....17

7.2 请求一个订阅项.....17

7.3 取消一个订阅项.....26

7.4 取消订阅.....28

7.5 预批准请求订阅.....30

8 交换表示信息.....32

8.1 表示基本原理.....32

8.2 初始化表示.....32

8.3 表示探测.....34

8.4 后续的表示广播.....37

8.5 不可用的表示.....39

8.6 定向的表示.....41

8.7 表示语法.....43

9 交换消息.....45

9.1 概述.....45

9.2 一对一的聊天会话.....45

9.3 消息语法.....46

9.4 扩展的内容.....50

10 交换IQ节.....51

11 一个示例会话.....51

12 服务器处理XML节的规则.....57

 12.1 一般性事项.....57

 12.2 没有“to”地址.....58

 12.3 远程域.....58

 12.4 本地域.....58

 12.5 本地用户.....58

13 URIs的处理.....62

14 国际化事项.....62

15 安全性事项.....62

16 一致性要求.....63

附录A（规范性附录） 订阅状态.....67

附录B（资料性附录） 通信阻塞.....72

附录C（资料性附录） vCards.....73

附录D（资料性附录） ‘jabber:iq:roster’的XML规划.....74

前 言

本标准是XMPP系列标准之一，本系列标准的名称和结构预计如下：

- 扩展消息与表示协议 核心协议；
- 扩展消息与表示协议 即时消息与表示；
- 扩展消息与表示协议 地址格式；
- 扩展消息与表示协议 端到端的签名与对象加密。

本标准按照GB/T1.1-2009给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由中国通信标准化协会提出并归口。

本标准起草单位：重庆邮电大学、中国信息通信研究院、国家电网公司信息通信分公司。

本标准主要起草人：谢昊飞、王 浩、蔡林沁、唐晓铭、李瑞雪、罗志勇、魏 旻、王 恒、兰 飞、赵 锋、刘建明、陈 星、 张炎、谢金凤。

扩展消息与表示协议(XMPP)

即时消息与表示

1 范围

本标准规定了扩展消息与表示协议（XMPP）的即时消息与表示的通信规则，用于使任意两个或多个的网络实体间能够进行准实时结构的未扩展数据交换，以便于实体能够获得更详尽的结构信息。

本标准适用于即时通讯系统。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

YD/T 2935	扩展消息与表示协议（XMPP）核心协议
IETF RFC2779	即时消息协议需求（Instant Messaging/ Presence Protocol Requirements）
IETF RFC3861	即时消息与表示的地址解决方法（Address Resolution for Instant Messaging and Presence）
IETF RFC3987	国际化资源标识符（Internationalized Resource Identifiers (IRIs)）
IETF RFC4422	简单认证与安全层协议（Simple Authentication and Security Layer (SASL)）
IETF RFC5122	针对 XMPP 的 IRIs 和 URIs（Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)）
IETF RFC524	传输层控制协议（The Transport Layer Security (TLS) Protocol Version 1.2）
IETF RFC6121	扩展消息与表示协议(XMPP) 即时消息与表示（Extensible Messaging and Presence Protocol (XMPP):Instant Messaging and Presence）
XSF XEP-0016	隐私名单（Privacy Lists）
XSF XEP-0045	多用户聊天协议（Multi-User Chat）
XSF XEP-0071	XHTML即时消息（"XHTML-IM"）
XSF XEP-0115	实体能力（Entity Capabilities）
XSF XEP-0191	简单通信阻塞（Simple Communications Blocking）
XSF XEP-0237	登记版本（Roster Versioning）

3 缩略语

下列缩略语适用于本文件。

IQ	Info/Query	信息/查询
IRIs	Internationalized Resource Identifiers	国际化资源标识符
JID	Jabber Identifier	标识符
SASL	Simple Authentication And Security Layer	简单认证和安全层协议
TLS	Transport Layer Security	传输层安全协议

URIs	Uniform Resource Identifiers	统一资源标识符
XML	Extensible Markup Language	可扩展标记语言
XMPP	Extensible Messaging and Presence Protocol	扩展消息与表示协议

4 需求

从习惯上说，即时通信应用结合了以下几个因素。

- a) 聚集的中心是一个联系人或好友的列表（在XMPP中，这个列表被称为“名册”）；
- b) 使用这种应用程序的目的是实现准实时的与特定的联系人交换简短的文本消息，通常大部分这种消息是连续的，并以本标准5.1描述的一对一“聊天会话”的形式进行；
- c) 这种消息的交换是基于“表示”的，例如，关于某个特定联系人的网络可用性的消息（由此知道进行一对一聊天会话的联系人有哪些是在线或可用的）；
- d) 表示消息只提供给那些明确通过认证的联系人，叫做“表示订阅”。

因此本标准在较高的级别上假设用户必须能够完成以下使用案例：

- 管理联系人列表中的条目
- 与联系人交换信息
- 与联系人交换表示信息
- 管理发给联系人的或从联系人收到的表示订阅

这些领域的功能详细定义在IETF RFC2779中，感兴趣的用户可以直接阅读原文关于需求方面更深入的内容。虽然基于XMPP的即时消息和表示信息符合RFC2779的要求，但它不是特意为那种协议设计的，因为基础协议是在RFC2779成文之前通过Jabber开放源代码社区的一个开放的开发过程发展出来的。尽管XMPP标准委员会在“XEP”系列协议中定义了许多其他方面的功能（如XSF XEP-0045中描述的多用户文本聊天），但是这些协议不包含在本标准之中，因为它们不是RFC2779所规定的。

实现注意事项：IETF RFC2779规定表示服务应与即时通信分离。例如，它应尽可能地用这个协议来提供一个表示消息服务、一个即时消息服务，或同时提供两者。尽管本标准假定实现和部署希望提供统一的即时消息和表示消息服务，但没有要求一个服务应同时提供表示消息服务和即时消息服务，并且协议也提供了把表示消息服务和即时消息服务分离成为独立服务的可能性（如当一个客户端尝试发送一个<message/>节时，一个只提供表示的服务可以返回一个<service-unavailable/>的节错误）。

5 功能概述

本章提供给了基于XMPP的即时通信和表示特征的功能总结。

YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》规定了一个XMPP客户端如何连接到XMPP服务器。特别是，其指定了客户端给其他的XMPP网络实体发送XML节（XMPP中的基本单元）之前应完成的先决条件。这些先决条件由XML流协商组成，并包括XML流头部的交换，可选的通过传输层安全协议（TLS）保障的通道加密，通过SASL保障的强制认证，以及为客户端寻址的流绑定资源。

兼容性要求：规定了一个额外的先决条件：正式建立一个即时通信和表示会话。实现和部署经验表明这个先决条件是必不可少的。然而，为了向后兼容，一个实现仍然提供这种功能。这使得旧的软件在能连接上网络的同时也能使新软件节省交互过程。

当实现YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》中规定的先决条件时，一个XMPP客户端和XMPP服务器之间会有一个长久的XML流交互，这使得用户能够通过发送和接收流来控制客户端可能存在的无限数量的XML节。这种流能用来交换消息、分享表示信息以及准实时的方式进行结构化请求-响应交互。在XML流协商以后，即时通信和表示会话的通用流如下所述。

- a) 获取花名册；
- b) 发送初始化表示消息给服务器，服务器广播给所有订阅的联系人，这样就从XMPP通信的角度“上线”了；
- c) 交换消息、管理表示订阅、完成名册更新以及在会话期间的一般过程和用特殊的语法生成其他的XML节；
- d) 当发送不可用的表示信息时，终止会话并关闭XML流。

6 名册管理

6.1 概述

在XMPP中，用户的联系人列表称为名册，包括任意数量的特定名册条目。每个用户的名册存储在用户的服务器上，从而这个用户可以从任何资源访问该名册信息。当用户在名册中增加条目或者修改已存在的条目时，如果没有错误发生，那么服务器应存储修改的数据，并且当一个通过认证的客户端请求名册时应返回它存储的数据。

安全性注意事项：由于用户的名册可能包含机密资料，服务器应限制对这些数据的访问，因此，只有授权的实体（通常仅限为账户的所有者）才能获取、修改或删除这些数据。

假设只有存储用户名册的地方才是用户注册账户的服务器，用户只有通过这个服务器才能通过认证并访问网络。本标准中删除了名册存储、账户注册和网络认证之间必须严格匹配的规则，导致用户可以把它们的名册存储在另一个地点，或者能将多个名册存储在多个地点。然而，由于缺少实施和部署更加灵活的名册存储模型的经验，使用“客户端”和“服务器”（用“联系人名单”取代“某一个联系人名单”），而不是采用新的术语代替“用户存储名册的地点”。未来的规范中可能会为无服务器名册存储或多名册管理提供规范的规则，但这种规则不在本标准的范围之内。

6.2 语法和语义

6.2.1 版本属性

版本属性是一个用于识别特定版本的名册信息的字符串。它应由服务器生成，并被客户端视为不透明。服务器可以采用任何适当的方法生成版本号，比如说名册数据中的哈希表或者是严格递增的序列号。

本标准建议包含版本号属性。

对于版本属性的使用在6.6中有详细介绍。

兼容性要求：<query/>元素中的版本属性在本标准中是最新定义的。

6.2.2 名册条目

一个名册中的<query/>元素可以包含一个或多个<item/>子元素。每个<item/>子元素描述一个唯一的“名册条目”（有时候也称为“联系人”）。

<item/>条目子元素的语法在下面的章节中描述。

6.2.2.1 Approved属性

“Approved”属性是一个布尔对象，“真”值用在7.4节所描述的预先批准的订阅信息中（默认的值是“假”，这与[XML-DATATYPES]保持一致）。

服务器在给客户端发布预先批准的订阅应包含“Approved”属性。客户端在发给服务器的名册设置中不能包含“Approved”属性，但是应用类型为“subscribed”和“unsubscribed”的表示节来管理下面（7.4）提到的预先批准的信息作为替代。

兼容性要求：<item/>元素中的“apprved”属性在本标准中是最新定义的。

6.2.2.2 Ask属性

<item/>元素中包含“订阅”值的“ask”属性用在各种包含“等待”状态的订阅子状态中，见7.1.2。

服务器应该包含“ask”属性来告知客户端“等待”子状态。客户端在发送给服务器设置名册信息中不能包含“ask”属性，但是应用类型为“subscribed”和“unsubscribed”的表示节来管理下面（7.2.2）提到的预先批准的信息作为替代。

6.2.2.3 JID属性

<item/>元素中的“jid”属性指唯一标识名册条目的Jabber标识符（JID）。

“jid”属性在任何时候客户端或服务器添加、更新、删除和返回名册条目都是必要的。

6.2.2.4 Name属性

<item/>元素中的‘name’属性特指与JID有关的，由用户决定（不是指联系人）。尽管“name”属性可能意味着用户是人，这对服务器是不透明的。然而，在各种扩展的背景下“name”属性可能被服务器用来进行匹配。

当客户端增加或更新名册条目时“name”属性是可选的。

6.2.2.5 Subscription属性

表示的订阅状态在<item/>元素的“subscript”属性中获取。与订阅相关的值是：

none：用户对联系人的表示没有订阅，而且联系人也没有订阅用户的表示；这是默认的值，因此订阅属性不会被包含，这种状态就认为是“none”；

to：用户订阅了联系人的表示，但是联系人没有订阅用户的表示；

from：联系人订阅了用户的表示，但是用户没有订阅联系人的表示；

both：用户和联系人相互订阅了对方的表示（也称为“相互订阅”）。

在名册结果中，客户端应忽略“subscription”属性中除“none”、“to”、“from”和“both”以外的任何值。

在名册推送中，客户端应忽略“subscription”属性中除“none”、“to”、“from”和“both”以外的任何值。

在名册设置中，“subscription”属性可能会包含“remove”值，这表示将这个条目从名册中删除；在名册set中，服务器应忽略“remove”以外的任何值。

“subscription”这个属性是可选的。

6.2.2.6 组元素

<group/>元素表示客户端指定的联系人的分组或类别。一个<item/>元素可能包含不止一个<group/>元素，这意味着名册组不是独有的。尽管<group/>元素的XML特征数据可能意味着用户是人，但它对服务器是不透明的，然而，<group/>元素可能被服务器用来对各种XMPP扩展目标进行匹配。

当客户端添加或更新名册条目时，<group/>元素是可选的。如果名册设置不包括<group/>元素，那么条目就认为没有分组。

6.2.3 名册获取

“名册获取”是客户端向服务器请求返回名册；从语法上讲，它是一个从客户端发送给服务器的类型为“get”的IQ节，而且包含一个名字空间为“jabber:iq:roster”的<query/>元素，这个<query/>元素不能包含任何<item/>子元素。

```
C: <iq from='juliet@example.com/balcony'
    id='bv1bs71f'
    type='get'>
  <query xmlns='jabber:iq:roster'/>
</iq>
```

发送一个名册获取期望的结果是服务器返回一个名册结果。

6.2.4 名册结果

“名册结果”是服务器对“名册获取”的响应；从语法上来说它是一个服务器发给客户端的IQ节，它的类型为“result”，名字空间为“jabber:iq:roster”。

名册结果中的<query/>元素为每个联系人分配<item/>元素，因此可以包含多个<item/>元素。

```
S: <iq id='bv1bs71f'
    to='juliet@example.com/chamber'
    type='result'>
  <query xmlns='jabber:iq:roster' ver='ver7'>
    <item jid='nurse@example.com'/>
    <item jid='romeo@example.net'/>
  </query>
</iq>
```

如果名册存在但是没有联系人，那么服务器应返回一个包含<query/>子元素的IQ-result，这个<query/>子元素中不包含<item/>子元素（如服务器不能返回一个空的类型为“error”的<iq/>节）。

```
S: <iq id='bv1bs71f'
    to='juliet@example.com/chamber'
    type='result'>
  <query xmlns='jabber:iq:roster' ver='ver9'>
  </query>
</iq>
```

如果名册不存在，那么服务器应返回一个带有<item-not-found/>条件的节错误。

```
S: <iq id='bv1bs71f'
    to='juliet@example.com/chamber'
    type='error'>
  <error type='cancel'>
    <item-not-found
```



```

xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
</error>
</iq>

```

6.2.5 名册设置

“名册设置”是客户端向服务器请求修改（如创建、更新或删除）一个名册条目；从语法上讲它是一个从客户端发送给服务器的IQ节，这个IQ节类型为“set”，名字空间为‘jabber:iq:roster’，并包含<query/>子元素。

以下是名册设置的使用规则。

- (1) <query/>元素应包含并只能包含一个<item/>元素。
- (2) 服务器应忽略任何‘subscription’属性中除“remove”以外的任何值。

安全性注意事项：通常意义下，名册设置这种IQ节不包含‘to’地址，导致从一个名册更新过的账户的通过认证的资源（全JID）会发送所有的名册设置信息。此外，IETF RFC6121要求服务器执行特殊情况来检查名册设置信息从而忽略‘to’地址；但是本标准去掉了这种特殊情况，这意味着名册设置除了发送者的地址以外可能还包含了‘to’地址。因此，处理名册设置的实体应确保通过发送名册设置来更新名册的实体是经过授权的；如果不是的话，就返回一个<forbidden/>错误。

```

C: <iq from='juliet@example.com/balcony'
    id='rs1'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'/>
  </query>
</iq>

```

6.2.6 名册推送

“名册推送”是服务器发送给客户端的关于最近创建、更新或删除名册条目的信息；从语法上讲它是一个从服务器发送给客户端的IQ节，类型为“set”，名字空间为“jabber:iq:roster”，并包含一个<query/>元素。

以下是名册推送信息的使用规则：

- a) 名册推送中的<query/>元素应包含并只能包含一个<item/>元素；
- b) 除非没有包含“from”属性，否则接收的客户端应忽略这个节（如隐含在用户的纯JID账户中）或者它的“from”属性匹配用户的纯JID <user@domainpart>。

```

S: <iq id='a78b4q6ha463'
    to='juliet@example.com/chamber'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'/>
  </query>
</iq>

```


由于YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》规定了IQ节的语法，每个接收到从服务器发出的名册推送信息的资源都应该返回一个类型为“result”或“error”的IQ节（然而，很多现有的客户端不会对名册推送做出回复）。

```
C: <iq from='juliet@example.com/balcony'
      id='a78b4q6ha463'
      type='result'/>
```

```
C: <iq from='juliet@example.com/chamber'
      id='a78b4q6ha463'
      type='result'/>
```

安全性注意事项：通常意义上，名册推送不包括“from”地址，导致从所有账号的纯JID的名册推送信息都会被发送。但是，本标准批准除用户的服务器以外的实体均需维持名册信息，这意味着名册推送信息可能包含除了用户账户纯JID以外的“from”地址。因此，客户端应通过检查名册推送信息发送者的“from”地址来确保是授权的用户。如果客户端从未授权的实体收到一个名册推送信息，它不能处理这个名册推送的数据；另外这个客户端要返回一个<service-unavailable/>的节错误或者完全不返回任何错误。

实现注意事项：客户端在处理名册推送时没有定义错误的情况；如果服务器收到一个“error”类型的IQ作为对名册推送的响应，则应该忽略这个错误。

6.3 登录时接收一个人的名册

连接到服务器并绑定一个资源之后，客户端应该在发送初始化表示信息之前请求名册（然而，由于可能不是所有的资源都想接收名册，如一个带宽受限的连接，客户端对于名册的请求是可选的）。在一个连接的资源上发送一个初始化表示，则称为一个“可用的资源”。如果一个连接的资源或可用的资源请求名册，则称为一个“感兴趣的资源”。服务器应对所有感兴趣的资源发送名册推送信息。

实现注意事项：表示请求订阅发给可用的资源，但与订阅状态改变有关的名册推送信息发送给感兴趣的资源。因此，如果希望接收到请求订阅和名册推送信息，它应既要发送初始化表示也要发送名册请求信息。

客户端向服务器请求名册：

```
C: <iq from='juliet@example.com/balcony'
      id='hu2bac18'
      type='get'>
  <query xmlns='jabber:iq:roster'/>
</iq>
```

```
S: <iq id='hu2bac18'
      to='juliet@example.com/balcony'
      type='result'>
  <query xmlns='jabber:iq:roster' ver='ver11'>
```

```

    <item jid='romeo@example.net'
      name='Romeo'
      subscription='both'>
      <group>Friends</group>
    </item>
    <item jid='mercutio@example.com'
      name='Mercutio'
      subscription='from'>
    <item jid='benvolio@example.net'
      name='Benvolio'
      subscription='both'>
  </query>
</iq>

```

如果服务器不能处理名册获取，它应返回一个节错误（例如，如果名册的名字空间不支持就返回<service-unavailable/>，如果服务器在进行故障处理正在返回的名册，则返回<internal-server-error/>）。

6.4 增加一个名册条目

6.4.1 请求

任何时候，客户端可以在名册中增加一个条目。这通过发送一个包含新条目的名册设置来完成。

```

C: <iq from='juliet@example.com/balcony'
  id='ph1xaz53'
  type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'
      name='Nurse'>
      <group>Servants</group>
    </item>
  </query>
</iq>

```

6.4.2 成功示例

如果服务器能够成功的处理新条目的名册设置（例如，如果没有发生错误），它应在用户的名册中创建这个条目并做如下处理。

服务器应向发送名册设置的已连接的资源返回一个类型为“result”的IQ节。

```

S: <iq id='ph1xaz53'
  to='juliet@example.com/balcony'
  type='result'>

```

服务器也应向用户所有感兴趣的资源发送一个包含新名册条目的名册推送信息，包括生成名册设置的资源。

```
S: <iq to='juliet@example.com/balcony'
    id='a78b4q6ha463'
    type='set'>
  <query xmlns='jabber:iq:roster' ver='ver13'>
    <item jid='nurse@example.com'
      name='Nurse'
      subscription='none'>
      <group>Servants</group>
    </item>
  </query>
</iq>
```

```
S: <iq to='juliet@example.com/chamber'
    id='x81g3bdy4n19'
    type='set'>
  <query xmlns='jabber:iq:roster' ver='ver13'>
    <item jid='nurse@example.com'
      name='Nurse'
      subscription='none'>
      <group>Servants</group>
    </item>
  </query>
</iq>
```

由于YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》中定义的IQ节的语法是授权的，每个从服务器收到名册推送的资源应该返回一个类型为“result”或“error”的IQ节（然而，很多现有的客户端并不返回）。

```
C: <iq from='juliet@example.com/balcony'
    id='a78b4q6ha463'
    type='result'/>
```

```
C: <iq from='juliet@example.com/chamber'
    id='x81g3bdy4n19'
    type='result'/>
```

6.4.3 错误示例

如果服务器不能成功的处理“名册设置”节，它应返回一个节错误。以下是定义的节错误情形。明显地，其他的节错误也可能出现，比如，若服务器在处理名册获取时发生了内部错误问题就会返回

<internal-server-error/>, 甚至如果服务器只批准利用web接口等非XMPP方法进行名册修改时, 即返回<not-allowed/>错误。

如果名册设置的发送者没有权限更新名册, 那么服务器应返回<forbidden/>的节错误(一般来说只有账户本身授权资源才有这种权限)。

如果名册设置包含以下几个违规的项, 服务器应给客户端返回一个<bad-request/>的节错误:

- a) <query/>元素包含不止一个<item/>子元素;
- b) <item/>元素包含不止一个<group/>元素, 但是存在重复的组(一种可能的判断重复与否的方法被描述在[XMPP-ADDR]中)。

如果名册设置信息中包含任何以下的违规项, 那么服务器应向客户端返回一个<not-acceptable/>类型的节错误:

- “name” 属性的长度超过了服务器配置的限制。
- <group/>元素中的 XML 特征数据的长度为零(为了从所有组中删除一个条目, 客户端应拒绝名册设置中的任何组元素)。
- <group/>元素中的 XML 特征的数据长度超过了服务器配置的限制。

错误: 非法实体的名册设置初始化

```
C: <iq from='juliet@example.com/balcony'
    id='ix7s53v2'
    to='romeo@example.net'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'/>
  </query>
</iq>
```

```
S: <iq id='ix7s53v2'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='auth'>
    <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

错误: 名册设置具有不止一个条目

```
C: <iq from='juliet@example.com/balcony'
    id='nw83vcj4'
    type='set'>
  <query xmlns='jabber:iq:roster'>
```

```

<item jid='nurse@example.com'
      name='Nurse'>
  <group>Servants</group>
</item>
<item jid='mother@example.com'
      name='Mom'>
  <group>Family</group>
</item>
</query>
</iq>

```

```

S: <iq id='nw83vcj4'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>

```

错误：名册设置包含过量操作的条目

```

C: <iq from='juliet@example.com/balcony'
    id='yl491b3d'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'
          name='[ ... some-very-long-handle ... ]'>
      <group>Servants</group>
    </item>
  </query>
</iq>

```

```

S: <iq id='yl491b3d'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>

```


</iq>

错误：名册设置包含重复的组

```
C: <iq from='juliet@example.com/balcony'
    id='tk3va749'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'
      name='Nurse'>
      <group>Servants</group>
      <group>Servants</group>
    </item>
  </query>
</iq>
```

```
S: <iq id='tk3va749'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

错误：名册设置包含空的组

```
C: <iq from='juliet@example.com/balcony'
    id='fl3b486u'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'
      name='Nurse'>
      <group></group>
    </item>
  </query>
</iq>
```

```
S: <iq id='fl3b486u'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='modify'>
```



```

    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>

```

错误：名册设置包含过长的组名

```

C: <iq from='juliet@example.com/balcony'
    id='qh3b4v19'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'
        name='Nurse'>
      <group>[ ... some-very-long-group-name ... ]</group>
    </item>
  </query>
</iq>

```

```

S: <iq id='qh3b4v19'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>

```

兼容性要求：如果<item/>元素“jid”属性值与用户账户的纯JID，即<localpart@domainpart>的值相匹配，许多服务器会向客户端返回一个<not-allowed/>类型的节错误。

6.5 更新名册条目

6.5.1 请求

更新一个已有的名册的条目和增加一个新名册条目的方法是一样的，例如，通过发送一个名册设置的IQ节给服务器。

因为名册条目是不可分割的，所以条目向名册设置节时应被正确的更新。

以下是客户端需要更新名册条目的几个原因。

- a) 增加一个组；
- b) 删除一个组；
- c) 改变处理方式；
- d) 删除处理方式。

名册条目定义如下：

```

<item jid='romeo@example.net'
    name='Romeo'>

```

```

    <group>Friends</group>
  </item>

```

名册中具有这项条目的用户可能想要增加这个条目到其他的组中:

```

C: <iq from='juliet@example.com/balcony'
    id='di43b2x9'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@example.net'
      name='Romeo'>
      <group>Friends</group>
      <group>Lovers</group>
    </item>
  </query>
</iq>

```

将来的某时刻, 用户可能想要从原来的组中删除这个条目:

```

C: <iq from='juliet@example.com/balcony'
    id='lf72v157'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@example.net'
      name='Romeo'>
      <group>Lovers</group>
    </item>
  </query>
</iq>

```

用户可能想要从所有的组中删除条目:

```

C: <iq from='juliet@example.com/balcony'
    id='ju4b62a5'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@example.net'/>
  </query>
</iq>

```

用户可能试图改变某个条目的处理方式:

```

C: <iq from='juliet@example.com/balcony'
    id='gb3sv487'
    type='set'>

```

```

<query xmlns='jabber:iq:roster'>
  <item jid='romeo@example.net'
        name='MyRomeo'/>
</query>
</iq>

```

用户可能随后试图删除所有的处理方式:

```

C: <iq from='juliet@example.com/balcony'
    id='o3bx66s5'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@example.net'
          name=''/>
  </query>
</iq>

```

实现注意事项: 包含一个空“name”属性与不包含‘name’属性是等价的; 都是将名字设置为空字符。

6.5.2 成功示例

正如增加一个名册条目, 如果名册条目能够成功的被处理, 那么服务器应在用户的名册中更新这个条目, 向所有用户感兴趣的资源发送名册推送节, 并发送一个类型为“result”的IQ节给初始化的资源。

6.5.3 错误示例

6.4.3描述的错误的错误的情形同样适用于更新一个名册条目。

6.6 删除一个名册条目

6.6.1 请求

在任何时候, 客户端可以通过发送一个名册设置节并指定‘订阅’属性的值为“remove”, 以便从其名册中删除一个条目。

```

C: <iq from='juliet@example.com/balcony'
    id='hm4hs97y'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@example.com'
          subscription='remove'/>
  </query>
</iq>

```

6.6.2 成功示例

正如增加一个名册条目一样, 如果服务器能够成功的处理这个名册设置, 那么它应在用户的名册中更新这个条目, 然后向用户所有感兴趣的资源发送一个名册推送节(将‘订阅’属性的值设置为“remove”), 并试图初始化资源发送一个IQ结果; 详细说明见6.3。

此外，用户的服务器可能需要生成一个或更多的与订阅相关的表示节，如下：

如果联系人对用户有表示订阅，那么用户的服务器应发送一个类型为“unsubscribe”的表示节给联系人（为了从联系人的表示中取消订阅）。

如果用户对联系人有表示订阅，那么用户的服务器应发送一个类型为“unsubscribed”的细节给联系人（为了在用户中取消对联系人的订阅）。

如果表示订阅是相互的，那么用户应既发送一个类型为“unsubscribe”表示节又发送一个类型为“unsubscribed”的表示节给联系人。

```
S: <presence from='juliet@example.com'
      id='lm3ba81g'
      to='nurse@example.com'
      type='unsubscribe'/>
```

```
S: <presence from='juliet@example.com'
      id='xb2c1v4k'
      to='nurse@example.com'
      type='unsubscribed'/>
```

6.6.3 错误示例

如果“jid”属性的值指定一个不在名册中的条目，那么服务器应返回<item-not-found/>节错误。
错误：没有找到名册条目。

```
C: <iq from='juliet@example.com/balcony'
      id='uj4b1ca8'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='[ ... non-existent-jid ... ]'
          subscription='remove'/>
  </query>
</iq>

S: <iq id='uj4b1ca8'
    to='juliet@example.com/balcony'
    type='error'>
  <error type='modify'>
    <item-not-found
      xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.7 名册版本

6.7.1 流特征

如果服务器支持名册版本，那么它应在流协商的期间广播以下的流特征：

```
<ver xmlns='urn:xmpp:features:rosterver'/>。
```

名册版本的流特征仅仅是广播性的，因此不必在协商阶段授权。

6.7.2 请求

如上面部分所提到的，如果服务器支持名册版本并且这个连接上的服务器通告它支持名册版本，那么客户端应该在它的名册请求中包含“ver”元素；如果这个服务器通告它不支持名册版本，客户端不授权包含“ver”属性；如果客户端在名册获取中包含了“ver”属性，它将这个属性的值设置到名册的最后一个缓存的版本号中。

```
C: <iq from='romeo@example.net/home'
    id='r1h3vzp7'
    to='romeo@example.net'
    type='get'>
  <query xmlns='jabber:iq:roster' ver='ver14'/>
</iq>
```

如果客户端还没有名册的缓存，或者缓存被丢失或毁坏了，但是这个客户端希望引导名册版本的使用，它应试图将空字符串设置“ver”属性（如ver=...）。

明显地，如果客户端不支持名册版本或者不想要引导名册版本的使用，它就不用包含“ver”属性。

6.7.3 成功示例

从版本号被客户端枚举以后，不论名册有没有被修改，服务器应返回完整的名册，或者是返回一个空的IQ-result节（因此表示任何名册的修改都要通过名册推送节来发送，如下文所述）。总之，除非返回完整的名册，否则（1）使用比发送单个的名册推送节更少的带宽，或者（2）服务器不能关联具有先前的版本记录的版本号，服务器应该发送一个空的IQ-result节，然后通过名册推送发送修改条款。

```
S: <iq from='romeo@example.net'
    id='r1h3vzp7'
    to='romeo@example.net/home'
    type='result'/>
```

实现注意事项：空的IQ-result不同于<query/>元素，因此需明白一个空名册的用法。

如果名册版本有效，并且名册在客户端枚举了版本ID以后没有修改过，服务器不应该发送任何名册推送节给客户端（直到客户端会话期间一些相关的事件触发了名册推送节）。

在客户端枚举了版本ID以后，如果名册已经被修改了，那么服务器应为每个被修改的名册条目发送一个名册推送节给客户端（将名册版本同步发送的名册推送称为“临时名册推送”）。

定义：“名册修改”是指任何对名册数据的改变将会导致一个相关的客户端名册推送。因此，服务器里与名册处理相关的内部状态不会导致向已连接的客户端发送没有必要改变版本的名册推送。

```
S: <iq from='romeo@example.net'
    id='ah382g67'
    to='romeo@example.net/home'
```



```

    type='set'>
    <query xmlns='jabber:iq:roster' ver='ver34'>
      <item jid='tybalt@example.org' subscription='remove'/>
    </query>
  </iq>

```

```

S: <iq from='romeo@example.net'
    id='b2gs90j5'
    to='romeo@example.net/home'
    type='set'>
  <query xmlns='jabber:iq:roster' ver='ver42'>
    <item jid='bill@example.org' subscription='both'/>
  </query>
</iq>

```

```

S: <iq from='romeo@example.net'
    id='c73gs419'
    to='romeo@example.net/home'
    type='set'>
  <query xmlns='jabber:iq:roster' ver='ver72'>
    <item jid='nurse@example.org'
      name='Nurse'
      subscription='to'>
      <group>Servants</group>
    </item>
  </query>
</iq>

```

```

S: <iq from='romeo@example.net'
    id='dh361f35'
    to='romeo@example.net/home'
    type='set'>
  <query xmlns='jabber:iq:roster' ver='ver96'>
    <item jid='juliet@example.org'
      name='Juliet'
      subscription='both'>
      <group>VIPs</group>
    </item>
  </query>
</iq>

```



```

    </item>
  </query>
</iq>

```

这些“临时的名册推送”可以做如下理解。

试想，客户端从整个缓存名册版本（如“ver14”）到新的名册版本期间具有活跃的代表会话（表示“ver96”）。

在此段期间内，客户端可能已经收到了与各种名册版本（可能已经，称为“ver51”和“ver79”）相关的名册推送。然而，一些推送可能包含一些名册条目的中间的更新（如将账户bill@example.org的订阅状态从“none”修改为“to”或者从“to”修改为“both”）。

临时名册推送不会包含所有的中间步骤，只有最终的修改结果，这适用于客户端离线时的所有条目（这可能已经被称为：“ver34”、“ver42”、“ver72”和“ver96”）。

客户端在处理“临时名册推送”时，使用与其他任何名册推送相同的方法（事实上，从客户端的角度来说，它不能分辨“临时的”名册推送和“永久的”名册推送，因此它没有办法知道自己什么时候收到临时名册推送）。在完成重新连接并请求名册时，客户端应该请求与它先前的最后一次会话有关的版本，而不是先前第一次会话的版本。

当名册版本有效以后，服务器应在每个名册推送节中包含更新的名册版本。名册推送节应按照修改的次序出现，并且名册推送包含的版本应是唯一的。即使客户端在它的名册中没有包含获取或者设置的版本属性，服务器应该在所有的名册推送中包含“ver”属性，最后发送给客户端。

实现注意事项：关于名册版本的指导方针和更详细的例子参见XSF XEP-0237。

7 管理表示订阅

7.1 概述

为了保护XMPP用户的隐私，表示信息只对用户已授权的实体公开。当用户同意另一个实体看到它的表示信息时，这实体称为对用户的表示信息有一个“订阅”。一个对用户的表示信息有订阅的实体或者被用户订阅的实体称为“联系人”（在本标准中，“联系人”这个术语被认为是潜在的联系人或用户名册中的任何条目是不严格的）。

在XMPP中，订阅存在于整个表示会话之中；事实上，它一直存在直到联系人取消订阅或被取消订阅曾经授权的订阅为止。

在XMPP中，订阅通过发送包含特殊属性的表示节来管理（“subscribe”、“unsubscribe”、“subscribed”和“unsubscribed”）。

实现注意事项：当服务器处理或生成一个类型为“ubscribe”、“unsubscribe”、“bscribed”和“unsubscribed”的出站表示节时，服务器应用发送实体的纯JID <localpart@domainpart>来标记对外的表示节，而不是全JID <localpart@domainpart/resourcepart>。执行这个规则可以简化表示订阅模型并有利于阻止表示泄露；关于表示泄露的信息，见YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》。

表示状态也反映在用户和联系人的名册中。这个部分不包括与表示订阅相关的各种可能的情况，而且主要讲述用户和联系人之间相互订阅的协议流。关于订阅状态的完整细节见附录A。

7.2 请求一个订阅项

7.2.1 客户端生成出站的请求订阅

用户的客户端通过发送一个类型为“subscribe”并且指定“to”地址为潜在联系人的纯JID<contact@domainpart>的表示节来产生一个请求订阅。

```
UC: <presence id='xk3h1v69'
      to='juliet@example.com'
      type='subscribe'/>
```

当用户给一个潜在的即时通信和表示联系人发送表示请求订阅时，“to”属性的值应是纯JID<contact@domainpart>而不是全JID<contact@domainpart/resourcepart>，因为期望的结果是让用户从所有联系人的资源接收表示信息，而不仅仅是“to”属性里某个特定的资源。使用纯JID也会是用户的服务器或联系人的服务器简化订阅、表示探测和表示通告的处理过程。

为了达到追踪的目的，客户端应该在表示请求订阅中包括“id”属性。

实现注意事项：在生成并发送一个出站的表示请求订阅之前，需要先发送一个名册设置，很多XMPP客户端都会提示用户关于潜在联系人的信息（如“处理的”和“期望得到的”名册组）。这种行为是可选的，因为服务器会将上传用户提供的关于联系人的名册设置信息替换成等待接收名册推送的行为。服务器应以任何顺序来处理名册设置和出站的表示请求订阅（如不论客户端是用哪种顺序生成的）。

7.2.2 服务器处理出站的请求订阅

一旦收到出站的表示请求订阅，用户的服务器应做如下处理。

a) 在处理这个请求之前，用户的服务器应先检查“to”属性中JID的语法（然而，一些现有的实现不执行这个检查）。如果JID是<contact@domainpart/resourcepart>这种形式，而不是<contact@domainpart>，那么用户的服务器应该认为这个请求是直接发送给联系人的纯JID，并且相应修改“to”地址。服务器也要检查JID的格式是否符合[XMPP-ADDR]的定义，也可能返回一个<jid-malformed/>的节错误；

b) 如果潜在的联系人和用户都在同一个服务器上，那么当处理请求订阅并转发给这个（本地）联系人的时候，服务器应符合7.1.3中的规则；

c) 如果潜在的联系人在远程服务器上，受制于本地服务器的规则，这个用户的服务器应按照核心XMPP节处理规则将这个节路由到远程的域（这可能会导致返回一个合适的节错误给用户，比如<remote-server-timeout/>）。

在上文中提到，在本地转发或远程路由表示请求订阅之前，用户的服务器应先使用用户的纯JID<user@domainpart>标记这个出站的请求订阅。

```
US: <presence from='romeo@example.net'
      id='xk3h1v69'
      to='juliet@example.com'
      type='subscribe'/>
```

如果表示请求订阅不能被本地转发或远程路由（如由于这个请求是错误的，本地联系人不存在，远程服务器不存在，尝试连接的远程服务器超时，或者是用户服务器定义的任何其他的错误），然后用户服务器应返回一个合适的节错误给用户。以下是一个例子：

```
US: <presence from='juliet@example.com'
      id='xk3h1v69'
      to='romeo@example.net'
```

```

        type='error'>
    <error type='modify'>
        <remote-server-not-found
            xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
</presence>

```

在本地转发或者远程路由这个表示请求订阅以后，用户服务器应发送一个名册推送节给用户感兴趣的所有资源，包括订阅状态为“none”的潜在的联系人和订阅标记为待定的（通过值为“subscribe”的“ask”属性）。

```

US: <iq id='b89c5r7ib574'
    to='romeo@example.net/foo'
    type='set'>
    <query xmlns='jabber:iq:roster'>
        <item ask='subscribe'
            jid='juliet@example.com'
            subscription='none'/>
    </query>
</iq>

```

```

US: <iq id='b89c5r7ib575'
    to='romeo@example.net/bar'
    type='set'>
    <query xmlns='jabber:iq:roster'>
        <item ask='subscribe'
            jid='juliet@example.com'
            subscription='none'/>
    </query>
</iq>

```

如果远程联系人在配置时间内不授权或拒绝请求订阅，用户服务器应该采用特定的实现算法重新发送请求订阅给联系人（如一个合适的新的资源对用户变为可用，或者在一段时间过去以后）；这对回复短暂的、不易察觉的错误有帮助，这可能在原始的请求订阅被路由到远程域的时候发生。若执行此操作，建议服务器包含一个“id”属性，这样就能跟踪重新发送的请求订阅的响应。

7.2.3 服务器处理入站请求订阅

在处理入站请求订阅之前，联系人的服务器应该检查包含在“to”属性中的JID的语法。如果JID是<contact@domainpart/resourcepart>形式而不是<contact@domainpart>形式，那么联系人的服务器应该将它视为直接发给联系人的纯JID，并相应的修改“to”地址。服务器可能也会检查JID地址是否符合YD/T

2935《扩展消息与表示协议（XMPP） 核心协议》中定义的格式，并可能返回一个<jid-malformed/>的节错误。

当处理入站的表示请求订阅的时候，联系人的服务器应遵守规则。

首先，联系人服务器不能以联系人的名义自动的批准请求订阅，除非（a）联系人预先批准了用户的请求订阅，参见7.4；（b）将账户配置为自动批准请求订阅；或者（c）与自动批准的服务提供者达成共识（如通过使用企业调度中的协议）。相反，如果请求订阅需要批准，那么联系人服务器应将这个请求投递给联系人可用的资源，让联系人来批准或者是拒绝。

如果联系人存在并且用户已经订阅了这个联系人的表示，那么联系人的服务器应以联系人的名义自动的回复，回复的方法是从联系人的纯JID向用户的纯JID发送一个类型为“subscribed”的表示节，而且联系人的服务器将这种情况视为对用户的表示订阅“预批准（pre-approval）”，然后联系人的服务器也应以联系人的名义自动返回。

```
CS: <presence from='juliet@example.com'
      id='xk3h1v69'
      to='romeo@example.net'
      type='subscribed'/>
```

另外，如果在请求订阅被联系人服务器接收到的时候至少有一个与联系人相关的可用的资源，那么联系人服务入站器应将这个请求订阅信息按照第12章的规则发送给所有的可用的资源。作为一种确认接收到表示请求订阅的方法，联系人服务器可能从联系人的纯JID向用户的JID发送一个类型为“unavailable”表示节（用户客户端不能假设这个确认提供与联系人相关的表示信息，因为它来自联系人的纯JID，而且它在请求订阅信息被批准以前被接收）。

此外，如果在联系人服务器收到请求订阅的时候没有可用的资源，那么联系人服务器应记录包含表示请求的完整表示节，其中包括所有扩展的内容，然后当联系人有可用的资源时将这个请求转发。当联系人创建了一个可用的资源以后，联系人服务器应继续转发这个请求订阅，直到联系人批准或拒绝这个请求（当联系人出现可用的资源以后服务器不能转发超过一个特定用户的请求订阅；例如：如果用户在某个联系人离线的时候向其发送了多个请求订阅，联系人的服务器值保存其中的一个，比如第一个或最后一个，然后再联系人上线只发送给一个请求；这有利于阻止“请求订阅泛洪”）。

安全性注意事项：联系人服务器存储整个表示请求订阅的授权可能会引入应用程序资源耗尽攻击，例如，通过服务器欺诈或调整用户组来对抗联系人的服务器或某个特定的联系人。对于服务器的实现，建议考虑这种可能的攻击并提供方法来抵消它，比如让服务器管理者设置进入的将要被记录的表示请求订阅的长度。

7.2.4 客户端处理入站的请求订阅

当一个交互式客户端接收到一个请求订阅时，它应将这个请求呈现给操作这个客户端的自然人（如“联系人”）批准，除非联系人明确的将客户端配置为自动批准或拒绝某些或所有的请求订阅，如上所述。一个自动处理的客户端有自己接收或拒绝请求订阅的规则。

客户端通过发送一个类型为“subscribed”的表示节来批准一个请求订阅，联系人服务器如7.1.5中表述的方式进行处理，用户服务器如7.1.6中表述的方式进行处理。其中，CC表示客户端到客户端。

```
CC: <presence id='h4v1c4kj'
```

```
to='romeo@example.net'
type='subscribed'/>
```

客户端通过发送一个类型为“unsubscribed”的表示节来拒绝一个请求订阅，联系人服务器和用户服务器都如第7.2节中表述的方式进行处理。

```
CC: <presence id='tb2m1b59'
to='romeo@example.net'
type='unsubscribed'/>
```

7.2.5 服务器处理出站的订阅批准

当联系人服务器发送订阅批准时，联系人服务器应用联系人的纯JID<contact@domainpart>来标记出站的节，并且将这个节本地转发或远程路由给用户。其中CS表示客户端到服务器。

```
CS: <presence from='juliet@example.com'
id='h4v1c4kj'
to='romeo@example.net'
type='subscribed'/>
```

联系人服务器应发送一个更新名册推送给联系人所有感兴趣的资源，其中‘subscription’属性的值设置为“from”。（这里假设联系人还没有订阅用户；若是此类情况，‘subscription’属性值要设置成“both”，根据附录A的解释。）

```
CS: <iq id='a78b4q6ha463'
to='juliet@example.com/balcony'
type='set'>
<query xmlns='jabber:iq:roster'>
<item jid='romeo@example.net'
subscription='from'/>
</query>
</iq>
```

```
CS: <iq id='x81g3bdy4n19'
to='juliet@example.com/chamber'
type='set'>
<query xmlns='jabber:iq:roster'>
<item jid='romeo@example.net'
subscription='from'/>
</query>
</iq>
```

从联系人的角度来看，现在存在一个来自用户的订阅，这就是“subscription”属性值设置为“from”的原因（这里假设联系人对用户还没有订阅；若类似此种情况，‘subscription’属性值要设置为“both”，按照附录A的解释）。

联系人的服务器应随后从联系人可用的资源中发送一个当前的表示给用户。

```
CS: <presence from='juliet@example.com/balcony'
      id='pw72bc5j'
      to='romeo@example.net'/>
```

```
CS: <presence from='juliet@example.com/chamber'
      id='ux31da4q'
      to='romeo@example.net'/>
```

为了订阅用户的表示，联系人需要发送一个请求订阅给用户。（XMPP客户端会经常自动发送请求订阅而不是需要联系人自己初始化请求订阅，从假设期望的目标状态是共同的订阅的时候开始。）当然，当联系人发送表示给用户的时候，订阅状态也会不同于前面那些例子（见附录A），而且角色也会颠倒过来。

7.2.6 服务器处理入站的订阅批准

当用户服务器接收到一个订阅批准的时候，它应先检测联系人是否在用户的名册中，其中 subscription='none' 或 subscription='from'，并且“ask”标志设置为“subscribe”（如订阅状态为“None + Pending Out”、“None + Pending Out+In”或“From + Pending Out”。见附录A）如果这个检查是成功的，那么用户服务器应：

a) 将入站的订阅批准传递给用户所有感兴趣的资源。这应发生在发送下面步骤表述的名册推送之前。其中，US表示用户到服务器。

```
US: <presence from='juliet@example.com'
      id='h4v1c4kj'
      to='romeo@example.net'
      type='subscribed'/>
```

b) 向所有用户感兴趣的资源取消初始化名册推送，包含对某个联系人的更新条目，其中‘subscription’属性值设置为“to”（如果订阅状态为“None + Pending Out”或者“None + PendingOut+In”）或者“both”（如果订阅状态为“From + PendingOut”）。

```
US: <iq id='b89c5r7ib576'
      to='romeo@example.net/foo'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@example.com'
          subscription='to'/>
  </query>
</iq>
```

```
US: <iq id='b89c5r7ib577'
      to='romeo@example.net/bar'
```



```

        type='set'>
      <query xmlns='jabber:iq:roster'>
        <item jid='juliet@example.com'
          subscription='to'/>
      </query>
    </iq>

```

c) 用户的服务器也应向用户的每个可用的资源转发从联系人的每个可用的资源收到的可用的表示节。

[... 给资源1 ...]

```

US: <presence from='juliet@example.com/balcony'
      id='pw72bc5j'
      to='romeo@example.net'/>

```

[... 给资源2 ...]

```

US: <presence from='juliet@example.com/balcony'
      id='pw72bc5j'
      to='romeo@example.net'/>

```

[... 给资源1 ...]

```

US: <presence from='juliet@example.com/chamber'
      id='ux31da4q'
      to='romeo@example.net'/>

```

[... 给资源2 ...]

```

US: <presence from='juliet@example.com/chamber'
      id='ux31da4q'
      to='romeo@example.net'/>

```

实现注意事项：如果在接收到入站的订阅批准通告时用户的账户没有可用的资源，用户的服务器可能会记录下通告（理想情况下是完整的表示节），并且在这个账户以后出现一个可用的资源时传递这个通告。这种行为向用户提供了不止一种发生在用户离线时的关于名册变更的完整信号。

另外——相反，如果用户不存在，如果联系人不在用户的名册中，或者如果联系人在用户的名册中，订阅状态不是上文描述的；

检查——然后用户服务器应无声的忽略订阅批准通告，而不转发给用户，不更改用户的名册，并且不向用户感兴趣的资源生成名册推送。

在用户看来，现在存在一个发送给联系人的表示（即是‘subscription’属性要设置为“to”的原因）。

7.3 取消一个订阅项

7.3.1 客户端生成取消订阅项

如果一个联系人希望取消一个先前对用户批准的订阅项，取消一个先前批准的订阅项（见7.4），或者拒绝一个请求订阅项，它会发送一个类型为“unsubscribed”的表示节。

```
CC: <presence id='ij5b1v7g'
      to='romeo@example.net'
      type='unsubscribed'/>
```

7.3.2 服务器处理发出的取消订阅项

一旦接收到出站的取消订阅项，联系人的服务器应作如下处理。

a) 如果用户的纯JID没有在联系人的名册中或者在用户的名册中状态为“None”、“None + PendingOut”或者“To”，那么联系人的服务器不能向用户路由或者转发这个类型为“unsubscribed”表示节，并且不能向用户发送类型为“unavailable”的表示通告；

b) 如果用户的纯JID是在联系人名册中，状态为“None”、“None + Pending Out”或“To”，并且‘批准’标志设置为“真”（因此像7.4描述的那样发送一个预先批准的订阅项信号），那么联系人服务器应删除这个预批准并且不能将类型为“unsubscribed”的表示节路由给用户；

c) 另外，如以下例子所示，联系人服务器应路由或者转发类型为“unavailable”的表示通告以及类型为“unsubscribed”的表示节给用户，而且应发送一个名册推送给联系人。

当用户仍然对联系人表示订阅时（如在联系人的服务器路由或转发这个类型为“unsubscribed”的表示节给用户之前），联系人的服务器应从联系人的所有在线资源发送一个类型为“unavailable”的表示节给用户。其中CS表示客户端到服务器。

```
CS: <presence from='juliet@example.com/balcony'
      id='i8bsg3h3'
      type='unavailable'/>
```

```
CS: <presence from='juliet@example.com/chamber'
      id='bvx2c9mk'
      type='unavailable'/>
```

然后联系人的服务器应路由或传递这个类型为“unsubscribed”的表示节给用户，确保将出站的取消订阅项标记为联系人的纯JID <contact@domainpart>。

```
CS: <presence from='juliet@example.com'
      id='ij5b1v7g'
      to='romeo@example.net'
      type='unsubscribed'/>
```

联系人的服务器应向所有联系人感兴趣的资源发送一个包含名册条目更新的名册推送，此类资源的订阅状态是现有的“none”或“to”见附录A）。

```
CS: <iq id='pw3f2v175b34'
      to='juliet@example.com/balcony'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@example.net'
          subscription='none'/>
  </query>
</iq>
```

```
CS: <iq id='zu2y3f571v35'
      to='juliet@example.com/chamber'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@example.net'
          subscription='none'/>
  </query>
</iq>
```

7.3.3 服务器处理入站的取消订阅项

当用户服务器接收到入站的取消订阅项时，它应先检测看联系人是否在用户的名册中，并且 subscription= ‘to’ 或者 subscription= ‘both’ （见附录A）。如果这个检测是成功，那么用户的服务器应：

- a) 向用户所有感兴趣的资源传送取消订阅项消息。这应发生在下一步的发送名册推送之前；

```
US: <presence from='juliet@example.com'
      id='ij5b1v7g'
      to='romeo@example.net'
      type='unsubscribed'/>
```

- b) 向用户所有感兴趣的资源发起名册推送，包含联系人名册条目更新，“subscription”属性值设置为“none”（如果subscription状态为“To”或“To + Pending In”）或者“from”（如果subscription状态为”Both”）；

```
US: <iq id='h37h3u1bv400'
      to='romeo@example.net/foo'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@example.com'
          subscription='none'/>
  </query>
```

```

    </query>
  </iq>

```

```

US: <iq id='h37h3u1bv401'
    to='romeo@example.net/bar'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@example.com'
        subscription='none'/>
  </query>
</iq>

```

用户服务器应要转发进入的类型为“unavailable”的表示节。

实现注意事项：如果在接收到进入的退订通告的时候用户账户没有可用的资源，那么用户服务器应记录下这个通告（理想的是记录完整的表示节），然后在以后出现用户时可用资源转发出去。这种行为能够为用户提供在离线期间与名册改变有关的更完整的信息。

另外——如果用户不存在，如果联系人不在用户的名册里，或者如果联系人在用户的名册中但订阅状态不是如上文描述的；

检查——那么用户服务器应无声地忽略这个退订通告节，不转发这个通告给用户，不修改用户的名册，并且不向用户感兴趣的资源生成名册推送。

7.4 取消订阅

7.4.1 客户端生成取消订阅

如果用户想要取消对一个联系人的订阅，那么它要发送一个类型为“unsubscribe”的表示节。

```

UC: <presence id='ul4bs71n'
    to='juliet@example.com'
    type='unsubscribe'/>

```

7.4.2 服务器处理出站的取消订阅

一旦接收到出站的取消订阅，用户服务器应作如下处理：

a) 如果联系人与用户在同一个服务器上，那么服务器应按照7.3.3中指定的类似于处理请求订阅规则来处理；

b) 如果联系人在远程服务器上，那么要按照XMPP节处理规则，将这个节路由到远程域。（这可能会导致产生一个合适的节错误给用户，比如<remote-server-timeout/>。）

如上所述，在本地转发或远程路由这个取消订阅之前，用户服务器应使用用户的纯JID<user@domainpart>来标记这个节。

```

US: <presence from='romeo@example.net'
    id='ul4bs71n'
    to='juliet@example.com'
    type='unsubscribe'/>

```


用户服务器应发送一个包含名册条目更新的名册推送给用户所有感兴趣的资源，其订阅状态是现有的“none”或者“from”（见附录A）。

```
US: <iq id='h37h3u1bv402'
      to='romeo@example.net/foo'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@example.com'
          subscription='none'/>
  </query>
</iq>
```

```
US: <iq to='romeo@example.net/bar'
      type='set'
      id='h37h3u1bv403'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@example.com'
          subscription='none'/>
  </query>
</iq>
```

7.4.3 服务器处理入站的取消订阅

当联系人服务器接收到取消订阅通告时，它应先检查用户的纯JID是否在联系人名册之中，并且订阅状态为subscription=‘from’或subscription=‘Both’（如订阅状态“from”、“From + Pending Out”或“Both”）。

a) 向联系人所有感兴趣的资源转发入站的取消订阅（这有助于给联系人的服务器提供更多关于取消订阅的情况，这样它们能够区分联系人另一个资源发出的原始的名册推送和来自用户的取消订阅的不同）。这应发生在下面步骤描述的发送名册推送之前；

```
CS: <presence from='romeo@example.net'
      id='ul4bs7ln'
      to='juliet@example.com'
      type='unsubscribe'/>
```

b) 向联系人所有感兴趣的资源发送名册推送，包括用户更新的名册条目，其中“subscription”属性值设置为“none”（如果订阅状态为“From”或“From + Pending Out”）或者“to”（如果订阅状态为“Both”）；

```
CS: <iq id='tn2b5893gls4'
      to='juliet@example.com/balcony'
      type='set'>
  <query xmlns='jabber:iq:roster'>
```

```

        <item jid='romeo@example.net'
            subscription='none'/>
    </query>
</iq>

```

```

CS: <iq id='sp3b56n27hrp'
    to='juliet@example.com/chamber'
    type='set'>
    <query xmlns='jabber:iq:roster'>
        <item jid='romeo@example.net'
            subscription='none'/>
    </query>
</iq>

```

- c) 从联系人每个可用的资源生成一个出站的类型为“unavailable”的表示节给用户。

```

CS: <presence from='juliet@example.com/balcony'
    id='o5v91w49'
    to='romeo@example.net'
    type='unavailable'/>

```

```

CS: <presence from='juliet@example.com/chamber'
    id='n6b1c37k'
    to='romeo@example.net'
    type='unavailable'/>

```

实现注意事项：如果联系人在收到入站的取消订阅通告时账户没有可用的资源，那么联系人服务器应记录下这个通告（最好是完整的表示节），并在用户出现可用的资源以后转发过去。这种行为为用户提供了更完整的关于用户离线时产生的名册改变信号。

与此相反，如果联系人不存在，或用户不在联系人名册中，或者如果用户的纯JID在联系人名册中，但是订阅状态不是前面提到的类型，那么用户服务器应无声地忽略这个退订通告，不转发这个通告给用户，不修改用户的名册，并且不向用户感兴趣的资源生成名册推送。然而，如果联系人服务器记录了从用户发往联系人的入站表示请求订阅，但是用户不在联系人的名册中（功能上等同于订阅状态“None+Pending In”，联系人从来没有将用户添加到联系人名册中），那么联系人服务器应仅仅只是删除入站表示请求订阅的所有记录（但不能从联系人的名册中删除用户，因为用户从来没有被添加到联系人名册中）。

实现注意事项：用户客户端不能依赖来自联系人的取消订阅通告，因为它应考虑它自己对于联系人的表示订阅，而且它关于联系人的表示信息，当它发送类型为“unsubscribe”的表示节或者当它接收来自取消请求订阅产生的名册推送触发的期间，是无效的或空的。

7.5 预批准请求订阅

7.5.1 概述

如果用户没有收到来自联系人的请求订阅，用户可以预批准此类请求，因此它以后可以通过用户服务器自动的批准请求订阅

在部分客户端和服务端对订阅的预批准的支持是可选的。如果一个服务器支持订阅的预先批准，那么它应在流协商期间广告如下的流特征：

```
<sub xmlns='urn:xmpp:features:pre-approval'/>
```

预先批准订阅的流特征仅仅是通告性的，因此不用强制协商。

7.5.2 客户端生成订阅的预批准

如果一个与客户端连接的服务器通告支持订阅的预批准，那么客户端可以通过发送一个类型为“subscribed”的表示节给联系人生成一个订阅预批准。

```
UC: <presence id='pg81vx64'
      to='juliet@example.com'
      type='subscribed'/>
```

如果服务器通告它不支持预批准，那么客户端不能尝试对潜在或实际的联系人的请求订阅进行预批准。

7.5.3 服务器处理订阅预批准

一旦接收到类型为“subscribed”的表示节，用户服务器如果支持订阅预处理的话，应作如下处理。

如果联系人在用户的名册中的状态为“Both”、“From”或“From + Pending Out”，那么用户服务器应无声地忽略这个节。

如果联系人在用户名册中的状态为“To + Pending In”、“None + Pending In”或“None + Pending Out+In”，用户服务器应像正常的订阅批准一样处理节，将现有的名册条目的状态更新为“Both”、“From”或“From + Pending Out”，向用户所有感兴趣的资源推送名册条目，并将类型为“subscribed”的表示节发送给联系人。

如果联系人在用户名册中状态为“To”、“None”或“None + Pending Out”，用户服务器要注意，订阅批准是通过将“approved”标志置为“真”，然后向用户所有感兴趣的资源推送修改后的名册条目。然而，用户服务器不能路由类型为“subscribed”的节给联系人。

如果联系人并不在用户名册中，用户服务器应为联系人创建一个名册条目，状态为“none”并设置“approved”标志为“真”，然后向用户感兴趣的资源推送名册条目。但是用户服务器不能将类型为“subscribed”的表示节路由给联系人。

名册推送的例子如下。

```
US: <iq id='h3bs81vs763f'
      to='romeo@example.net/bar'
      type='set'>
  <query xmlns='jabber:iq:roster'>
    <item approved='true'
          jid='juliet@example.com'
          subscription='none'/>
```



```

    </query>
  </iq>

```

当“approved”标志设置为“真”时，用户服务器不能将类型为“subscribe”的表示节从联系人发给用户，相反地，应自动返回代表用户响应的表示节，其中节类型为“subscribed”，从用户的纯JID发送给联系人的纯JID。

实现注意事项：对于在收到来自联系人的表示请求订阅以后，服务器是否还维持订阅批准记录和实现本地服务策略的问题。如果服务器不维持此类记录，那么一旦接收到请求订阅，它不会在联系人的名册条目中包含“approved”属性（如在随后的名册推送和名册结果中）；如果服务器维持此类记录，它会一直为联系人的名册条目包含“approved”属性（设置为“真”），直到用户给联系人发送一个类型为“unsubscribe”的表示节（或者从名册实体中删除联系人）。

实现注意事项：一个客户端可以通过发送一个类型为“unsubscribed”的表示节来取消预批准，这在7.2中描述得更加详细。在这种情况下，用户服务器需向用户所有感兴趣的资源发送一个删除了“approved”属性的名册推送（另一种方法，客户端可以只删除名册实体）。

8 交换表示信息

8.1 表示基本原理

表示的概念是指一个实体在网络中的通信可用性。在最基本的层次中，表示是指明一个实体在通信中是否可用的布尔“on/off”变量。在XMPP中，一个实体的可用性在它的客户端生成一个没有“type”属性的<presence/>节的时候被标识，而一个实体不可用性是在它的客户端生成一个“type”属性为“unavailable”的<presence/>节的时候被标识。

XMPP表示通常按照“发布—订阅”或“观察者”的模式，实体发送表示给它的服务器，然后服务器广播这个信息给所有订阅了这个实体的表示的联系人，一个生成表示的实体是“呈现者”，而一个接收表示的实体是“订阅者”。客户端通过发送一个没有“to”地址的表示节，这个表示节或者没有“type”属性或者“type”属性为“unavailable”，并将这个表示节发送给它的服务器，服务器广播给所有订阅的实体从而生成表示。这种类型的表示称为“广播表示”。（客户端也能发送“直接表示”，例如，带有“to”地址的表示节；这使用不多但是有时用户发送表示给那些没有订阅用户表示信息的实体，见8.6）。

在客户端完成了YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》规定的先决条件以后，它可以通过发送一个初始化表示（8.2）在服务器中建立一个“表示会话”，表示会话通过发送不可用表示终止。在表示会话阶段，一个连接的资源被称为“可用的资源”。

在XMPP中，连接消息和表示的应用程序，从功能上讲，传递表示信号可用性的通信的默认类型是消息；然而，XMPP应用程序没有必要将消息和表示功能相结合，而且它们能够提供没有消息的单独的表示特征（另外，XMPP服务器不需要网络可用性来保证消息和IQ节路由成功）。

消息要求：在下面的例子中，用户是<juliet@example.com>，包含两个可用的资源（“balcony”和“chamber”），而且其名册中有三个订阅状态为“from”或“both”的联系人：<romeo@example.net>、<mercutio@example.com>和benvolio@example.net。

8.2 初始化表示

8.2.1 客户端生成初始化表示

在完成YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》规定先决条件之后，客户端通过给它的服务器发送“初始化表示”来标记通信的可用性，例如，一个没有“to”地址（表示要被服务器代表客户端广播）并且没有“type”属性的表示节（表示用户的可用性）。

UC: <presence/>

除了扩展内容以外，初始化表示节可能包含<priority/>元素、<show/>元素和一个或多个<status/>元素；细节在8.7中描述。

8.2.2 服务器处理初始化表示

一旦从客户端接收到初始化表示，用户服务器应从用户的全JID<user@domainpart/resourcepart>发送初始化表示信息；这些都是JID存在于用户名册中的联系人，并且‘subscription’属性值被设置为“from”或“both”。

US: <presence from='juliet@example.com/balcony'
to='romeo@example.net'/>

US: <presence from='juliet@example.com/balcony'
to='mercutio@example.com'/>

US: <presence from='juliet@example.com/balcony'
to='benvolio@example.net'/>

用户的服务器也应从用户最新可用的资源中广播初始化表示给用户所有的可用的资源，包括起初生成表示广告的资源（如一个实体暗中订阅了它自己的表示）。

[...对于“公开的”资源 ...]

US: <presence from='juliet@example.com/balcony'
to='juliet@example.com'/>

[... 对于“隐藏的”资源 ...]

US: <presence from='juliet@example.com/balcony'
to='juliet@example.com'/>

如果没有关于用户的联系人的表示信息，用户服务器应也代表用户向用户的联系人发送表示探测，如8.4所述。

8.2.3 服务器处理入站的初始化表示

一旦接收到来自用户的表示，联系人的服务器应转发用户的表示节给联系人所有可用的资源。

[... 向资源1 ...]

CS: <presence from='juliet@example.com/balcony'
to='romeo@example.net'/>

[... 向资源2 ...]

```
CS: <presence from='juliet@example.com/balcony'
      to='romeo@example.net'/>
```

8.2.4 客户端处理初始化表示

当联系人客户端接收到来自用户的表示时，建议交互的客户端进行如下行为：

如果用户的纯JID在联系人的名册中，在合适的名册界面中显示表示信息。

如果用户不在联系人的名册中，但是联系人和用户都在积极地交换消息或IQ节，为通信会话在用户界面中显示表示信息。

否则，就忽略表示信息并不将它显示给联系人。

8.3 表示探测

8.3.1 概述

“表示探测”是对一个联系人当前的表示信息的请求，由用户服务器代表用户发送；从语法上来讲，它是“type”属性为“probe”的表示节。在表示订阅的背景下，“from”地址的值应为订阅用户的纯JID，而“to”地址应为用户订阅的联系人纯JID，因为表示订阅是基于纯JID的。

```
US: <presence from='juliet@example.com'
      id='ign291v5'
      to='romeo@example.net'
      type='probe'/>
```

兼容性要求：IETF RFC6121中规定检测是针对全JID的，而不是纯JID。

已接收到表示订阅背景之外的表示信息的实体也可以发送表示探测，通常在联系人已经发送了第8.6节中描述的直接表示的情况；在这种情况下“from”或“to”地址的值可以是全JID而不是纯JID。完整的了解请见8.6节。

表示探测不能由客户端发送，由于通常客户端没有必要发送它们，因此从用户的联系人收集表示信息是用户服务器的工作。然而，如果一个用户客户端产生了一个出站的表示探测，那么用户服务器应该路由这个探测（如果联系人在另一个服务器）或是处理这个探测（如果联系人在同一个服务器上），而且不能接收来自自己连接客户端的表示来探测作为客户端返回一个节或流错误的唯一原因。

8.3.2 服务器生成出站的表示探测

当一个服务器需要发现一个用户的联系人的可用性的时候，它从用户的纯JID <user@domainpart>向联系人的纯JID <user@domainpart>发送一个表示探测。

实现注意事项：尽管表示探测是为发送给联系人做准备的（如一个用户订阅的实体），但是服务器可能也要向一个实体的全JID发送表示探测，这个实体在当前的会话期间已经收到过表示信息。

当用户通过发送初始化表示而开始一个新的表示会话时，用户的服务器应发送一个表示探测；但是，如果服务器认为它已有关于用户联系人的最新的可靠的表示信息，那么服务器可能选择不发送探测（如用户有另一个可用的资源或者用户在新的表示会话开始前暂时掉线）。此外，如果服务器在一段配置的时间里还没有收到表示信息或其他流量，那么它可能周期性地向联系人发送表示探测；这有助于阻止那些好像是在线其实不在线的“异常的”联系人。

```
US: <presence from='juliet@example.com'
```

```

id='ign291v5'
to='romeo@example.net'
type='probe'/>

```

```

US: <presence from='juliet@example.com'
id='xv291f38'
to='mercutio@example.com'
type='probe'/>

```

当然，如果联系人账号与用户在同一个服务器上，那么用户服务器不用发送表示探测，因为服务器是本地处理用户信息。

8.3.3 服务器处理入站的表示探测

一旦从代表用户的服务器接收到发往联系人纯JID的表示探测，联系人服务器应做如下回复：

a) 如果联系人账户不存在，或者用户的纯JID在联系人名册中并且订阅状态不为“From”、“From + Pending Out”或“Both”（如附件A所解释的），然后联系人服务器应该返回一个类型为“unsubscribed”的表示节作为对表示探测的响应（这将会引起一个流向联系人的取消用户的订阅的协议。但是这并不会导致取消如第7.4节所述的订阅的预批准）。这里“from”地址应是联系人的纯JID，因为指定一个全JID将会构成表示泄露。

```

CS: <presence from='mercutio@example.com'
id='xv291f38'
to='juliet@example.com'
type='unsubscribed'/>

```

但是，如果服务器收到一个来自服务器自己配置域或其他的类似的可信的服务的表示探测，那么它可以提供关于用户的表示信息给那个实体。

b) 否则，如果联系人临时或永久的移动到了另一个地址，那么服务器应该返回一个类型为“error”的表示节，并且节错误条件为<redirect/>（临时的）或<gone/>（永久的），其中包含联系人的新地址。

```

CS: <presence from='mercutio@example.com'
id='xv291f38'
to='juliet@example.com'
type='error'>
  <error type='modify'>
    <gone xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
      xmpp:la-mer@example.com
    </gone>
  </error>
</presence>

```

c) 否则，如果联系人没有可用的资源，那么服务器应该通过向用户发送一个类型为“unavailable”的表示节来作为对表示探测的回复。因为没有与联系人相关的可用资源，这里的“from”地址是纯JID。如果适当的按照本地安全策略，这个表示通告可能包含最后一个不可用的表示节的完整的XML流，这是

服务器接收到的来自联系人表示节，否则表示通告应该指出没有提供最初的详细联系人是不可用的。在任何情况下，返回给探测实体的表示通告应该包含关于在生成最后不可用的表示节的时候的信息（用XMPP延迟转发扩展格式化的）。

```
CS: <presence from='mercutio@example.com'
      id='xv291f38'
      to='juliet@example.com'
      type='unavailable'>
  <delay xmlns='urn:xmpp:delay'
        stamp='2002-09-10T23:41:07Z'/>
</presence>
```

d) 否则，如果联系人最后有一个可用的资源，那么服务器应通过给用户发送最后一个没有“to”属性的表示节的完整的XML作为对表示探测的回复，这个表示节是服务器从联系人的每个可用的资源接收到的。此处‘from’地址是每个可用资源的全JID。

```
CS: <presence from='romeo@example.net/foo'
      id='hzf1v27k'
      to='juliet@example.com'/>
```

```
CS: <presence from='romeo@example.net/bar'
      id='ps6t1fu3'
      to='juliet@example.com'>
  <show>away</show>
</presence>
```

实现注意事项：经过“完整的XML”表示完整的节从开始标签<presence>到结束标签</presence>，包括所有的元素和属性，不论是否被内容名字空间或扩展名字空间所限制；然而，如果在一个服务器到服务器上发送完整节，那么联系人服务器需要将内容名字空间从‘jabber:client’转到‘jabber:server’。

如果联系人服务器接收到地址为联系人全JID的表示探测，那么服务器不能返回任何关于资源表示信息，除非是检测中指定‘to’地址的资源。针对纯JID检测的规则#1和#2同样适用于全JID检测的情况。如果有一个使全JID和检测实体相匹配的资源来查看联系人的表示，这个资源已经通过了表示订阅认证，那么服务器应返回一个可用的表示通告，这个表示通告应该只传达这个资源是可用的信息（而不是<show/>、<status/>、<priority/>或表示扩展这些细节信息）。

```
CS: <presence from='romeo@example.net/bar'
      to='lobby@chat.example.com'/>
```

实现注意事项：参见第8.6节关于处理直接表示的规则。

下面介绍“id”属性的处理

关于表示探测的‘id’属性的处理在IETF RFC6121中没有指出。尽管“发送一个检测并接收一个回复”的模式看起来像与XMPP<iq/>节相似的请求—响应协议，但实际上不是，因为对检测的响应可能由多个表示节构成（一个对用户当前有效的每一个可用的资源）。由于这个原因，如果联系人当前有可用的资源，那么服务器在发送这些表示通告给检测的实体时，应该保存联系人最初的表示节的‘id’属性。

相比之下，如果联系人当前没有可用的资源，那么检测的实体不能被授权（通过表示订阅）看到联系人的表示，或者出现一个域检测相关的错误，然后联系人服务器应该在恢复检测实体的时候反映用户的表示探测的“id”。

下面的例子说明了此处的不同点。

在第一个场景中，Juliet从它“隐藏”的资源发送表示。

```
CC: <presence from='juliet@example.com/chamber' id='pres1'>
    <show>dnd</show>
    <status>busy!</status>
</presence>
```

它也从“公开的（balcony）”资源发送。

```
CC: <presence from='juliet@example.com/balcony' id='pres2'>
    <show>away</show>
    <status>stepped away</status>
</presence>
```

远程的服务器然后给Juliet发送一个检测。

```
US: <presence from='romeo@example.net' id='probe1' type='probe' />
```

Juliet’s的服务器接着向Romeo发送它所有的表示通告，保留其客户端已经发送的节的‘id’属性。

```
CS: <presence from='juliet@example.com/chamber' id='pres1'>
    <show>dnd</show>
    <status>busy!</status>
</presence>
```

```
CS: <presence from='juliet@example.com/balcony' id='pres2'>
    <show>away</show>
    <status>stepped away</status>
</presence>
```

在第二个场景中，Juliet在Romeo’s的服务器上发送检测的时候是离线的

```
US: <presence from='romeo@example.net'
    id='probe2'
    type='probe' />
```

Juliet’s的服务器用一个不可用的通告回复，来反映出Romeo’s的表示探测的‘id’，由于从客户端发出的可用的通告中没有预留的‘id’。

```
CS: <presence from='juliet@example.com'
    id='probe2'
    type='unavailable' />
```

8.4 后续的表达广播

8.4.1 客户端生成后续的表达广播

在发送初始化表示以后，在会话期间的任何时候，用户客户端可以通过发送广播的表示节来更新它的可用性，在这个表示节中不能由“to”地址和“type”属性。

```
UC: <presence>
    <show>away</show>
</presence>
```

表示广播可以包含<priority/>元素、<show/>元素和一个或多个<status/>元素的实例，以及扩展的内容；细节在第8.7节中描述。

然而，用户应该通过发送一个表示更新来广播与用户通信可用性或资源通信能力相关的信息，而不是与可能对这种相关的信息用户的联系人感兴趣，但是应该应用其他方式进行发送，比如说XEP-0163中描述的“发布-订阅”方式。

8.4.2 服务器处理后续的出站表示

一旦接收到一个表示更新的可用性的表示节时，用户的服务器应广播类型为“from”或“both”表示节的完整的XML给用户名册中的联系人。

兼容性要求：IETF RFC6121规定用户的服务器在发送后续的表示通告之前应检测确保没有接收到来自联系人的表示错误。这条规则已经删除了，由于本标准使用类型为“unsubscribe”的表示节来解决订阅同步的问题，在某种程度上由于这种节将用户名册中的联系人的订阅状态改为“none”或“to”，因此消除了检查错误的需要。

兼容性要求：如果订阅状态为“both”，一些现有的服务器实现只在用户服务器上的联系人上线是才发送随后的表示通告给联系人（如果用户服务器从来没收到作为对它发给联系人的表示探测的联系人在线的有效的指示，那么用户服务器不向用户发送后续的表示通告给联系人）。这种行为认为是节省带宽，因为大多数表示订阅是双向的，而且很多联系人在一些特定的时间里不是在线的状态。

```
US: <presence from='juliet@example.com/balcony'
    to='romeo@example.net'>
    <show>away</show>
</presence>
```

```
US: <presence from='juliet@example.com/balcony'
    to='benvolio@example.net'>
    <show>away</show>
</presence>
```

```
US: <presence from='juliet@example.com/balcony'
    to='mercutio@example.com'>
    <show>away</show>
</presence>
```

实现注意事项：见8.6节关于处理直接表示的规则。

用户服务器也应向所有用户可用的资源发送表示节（包括先前生成表示通告的资源）。

```
US: <presence from='juliet@example.com/balcony'
      to='juliet@example.com/chamber'>
  <show>away</show>
</presence>
```

```
US: <presence from='juliet@example.com/balcony'
      to='juliet@example.com/balcony'>
  <show>away</show>
</presence>
```

8.4.3 服务器处理后续的入站表示

当接收到来自用户的表示时，联系人服务器应转发用户的表示节给联系人的所有的可用的资源。

[...向资源1...]

```
CS: <presence from='juliet@example.com/balcony'
      to='romeo@example.net'>
  <show>away</show>
</presence>
```

[...向资源2...]

```
CS: <presence from='juliet@example.com/balcony'
      to='romeo@example.net'>
  <show>away</show>
</presence>
```

8.4.4 客户端处理后续的表示

从联系人客户端的角度来看，初始化表示广播与随后的表示之间并没有多少不同，因此联系人客户端按照8.4.3中处理处理入站的表示的规则来处理。

8.5 不可用的表示

8.5.1 客户端生成不可用表示

在结束与服务器表示会话之前，用户的客户端应该通过发送“不可用表示”节来使自己不可用，例如，一个没有“to”属性和“type”属性为“unavailable”的表示节。

```
UC: <presence type='unavailable'>
```

可选的，不可用的表示节可以包含一个或更多的指定了用户不再可见的<status/>元素。

```
UC: <presence type='unavailable'>
  <status>going on vacation</status>
</presence>
```

然而，不可用的表示节不能包含<priority/>元素或<show/>元素，因为这些元素只适用于可用的资源中。

8.5.2 服务器处理出站的不可用的表示

用户的服务器不能依赖从一个可用的资源接收不可用的表示，因为这个资源可能突然变得不可用了（如这个资源的XML流可能在有或没有任何流错误的情况下关闭）。

如果一个可用的资源由于任何原因变得不可用，用户服务器应向用户名册中所有联系人广播订阅类型为“from”或“both”的不可用表示。

兼容性要求：IETF RFC6121规定，用户服务器在发送不可用表示通告之前要检查确保它没有从联系人接收到一个表示错误。这条规则在本标准中去掉了，因为本标准中使用类型为“unsubscribe”（而不是“error”）的表示节来解决订阅同步的问题，在某种程度上是由于节将用户名册中的联系人的订阅状态改变为“none”或“to”，因此取消检查错误的需要。

实现注意事项：即使用户的服务器不向离线的联系人广播用户后续的表示通告，它也应广播用户的不可用表示通告；如果它不这样做，那么联系人从服务器接收到的最后的表示将会是用户在表示会话中的初始化表示，从而导致联系人认为用户是在线的。

实现注意事项：见8.6中关于处理直接表示的规则。

如果来自客户端的不可用通告被无声的接收到了，那么服务器应广播表示节的完整XML。

```
US: <presence from='juliet@example.com/balcony'
      to='romeo@example.net'
      type='unavailable'>
  <status>going on vacation</status>
</presence>
```

```
US: <presence from='juliet@example.com/balcony'
      to='benvolio@example.net'
      type='unavailable'>
  <status>going on vacation</status>
</presence>
```

```
US: <presence from='juliet@example.com/balcony'
      to='mercutio@example.com'
      type='unavailable'>
  <status>going on vacation</status>
</presence>
```

用户服务器也应向用户所有可用的资源发送不可用通告（和最初生成不可用表示的资源一样）。

```
US: <presence from='juliet@example.com/balcony'
      to='juliet@example.com/chamber'
      type='unavailable'>
  <status>going on vacation</status>
</presence>
```

如果服务器检测到用户突然离线了，那么服务器应代表用户生成一个不可用的表示广播。

实现注意事项：任何没有‘type’属‘to’属性，而且是在服务器广播或生产不可用表示通告之后客户端发送的表示节，应通过用户服务器向所有的订阅者转发或路由（如应将其作为一个新的表示会话的初始化表示来看待）。

8.5.3 服务器处理入站的不可用表示

当接收到一个来自用户的不可用通告，联系人的服务器应向所有联系人的可用的资源转发用户的表示节。

[...向资源1 ...]

```
CS: <presence from='juliet@example.com/balcony'
      to='romeo@example.net'
      type='unavailable'>
    <status>going on vacation</status>
  </presence>
```

[... 向资源2 ...]

```
CS: <presence from='juliet@example.com/balcony'
      to='romeo@example.net'
      type='unavailable'>
    <status>going on vacation</status>
  </presence>
```

实现注意事项：如果联系人的服务器向离线的用户广播以后的表示信息，那么它也应更新其内部代表性的在线的实体，尽管用户是不可用的。

8.5.4 客户端处理不可用的表示

从联系人客户端的角度来看，可用的表示广播和不可用的表示广播之间并没有多大的不同，因此通常联系人服务器按照4.4.3描述的处理入站的表示的规则来处理。

然而，如果联系人从用户的纯JID（而不是某个特定的可用的资源的全JID）接收到一个不可用的通告，联系人客户端应该像所有资源那样对待这个不可用的通告。

8.6 定向的表示

8.6.1 总体考虑

通常，客户端在希望与一个没有订阅它的表示的资源分享它的表示信息的时候发送定向表示，这通常在临时的基础上。定向表示的普通用户包括如9.1所描述的临时的一对一聊天会话以及如XEP-0045描述的多人聊天室。

临时关系通过与另一个实体分享定向表示建立，这个实体相比通过表示订阅建立的永久关系要次要一些。因此，对表示订阅的创建、修改或删除要优先于以下部分介绍的规则。举例说明：如果一个用户与一个联系人分享表示，但是后来要通过表示订阅“握手”来增加此联系人，比如，通过向联系人发送以后的表示广播。另一个例子，如果用户后续需要取消联系人对用户表示的订阅，用户服务器应像7.2所描述的那样正常的操作，这包括发送不可用表示给联系人，即使用户已经向联系人发送过定向表示。

XMPP服务器通常通过维护一个实体（纯JID或全JID）列表来实现定向表示，列表中的实体在与用户的某个特定的资源（全JID）的当前会话期间已经发送过定向表示，然后当用户离线（比如通过发送一个类型为“unavailable”的表示节广播）时清空列表。服务器应从表示列表中删除任何给用户发送不可用表示或被用户发送不可用表示的实体。

8.6.2 客户端生成定向表示

定向表示是客户端生成的带有‘to’属性且是另外实体的全JID或纯JID的表示节，而且这个表示节没有“type”属性（表示可用性）或“type”属性值为“unavailable”。

8.6.3 服务器处理出站的定向表示

当用户服务器接收到定向表示节的时候，它应按照如下规则来处理：

如果用户在发送初始化表示之后和发送不可用表示广播之前向名册中的联系人发送可用或不可用的定向表示，并且这个表示节订阅类型为“from”或“both”，（如在用户表示会话阶段），用户服务器应本地转发或远程路由这个表示节的整个XML，但是不能另外修改关于表示广播的联系人状态。（如其应该在任何后续的表示广播中包含联系人的经用户初始化的JID。）

在发送初始化表示之后且发送不可用表示广播之前（比如在用户表示会话期间），如果用户向不在用户名册中的实体发送订阅类型为“from”或“both”的定向表示，那么用户服务器应向实体本地转发或远程路由这个表示节的完整XML流，但是不能修改联系人关于可用的表示广播的状态（如它不能在任何后续的广播中包含实体的JID）；然而，如果从用户发送的可用资源中发送的定向表示变得不可用，那么用户的服务器应将这个不可用的表示路由到实体（如果用户还没有向实体发送定向的不可用表示）。

如果用户在没有先发送初始化表示时或在已经发送过不可用的表示广播之后发送定向表示（如资源是连接的但不可用），那么用户服务器应像情况#2中用户发送定向表示那样对待这个实体。

8.6.4 服务器处理入站的定向表示

从联系人服务器的角度来看，表示广播和定向表示之间没有大的差别，因此联系人服务器按照第8.3.3、8.4.3和8.5.3中定义的处理入站表示的规则那样来处理。

8.6.5 客户端处理入站定向表示

从联系人客户端的角度看，表示广播和定向表示之间没有太大的差别，因此联系人的客户端按照8.4.3订阅的处理出站表示的规则来处理。

8.6.6 服务器处理表示探测

如果用户客户端已经向另一个实体发送了一个定向表示（如一个一对一的聊天用户或一个多用户聊天室），一段时间后，这个实体或它的服务器可能想要知道这个客户端是否仍然在线。这个场景在多用户聊天室中尤为普遍，其中用户可能很长一段时期作为参与者。如果用户的客户端在没有告知聊天室的情况下（或者是通过客户端或者是服务器端）离线了，那么用户在聊天室中的带就成为了“异常”状态，其看起来在参与其实已经不在聊天室中了。为了检测那种“异常的”状态，一些多用户聊天室向已经加入聊天室的用户发送表示探测。

在定向表示的情况下，正在检测的实体应该从那些接收到定向表示的JID发送检测（不论是全JID或纯JID）。检测应该被发送到用户的全JID，而不是没有资源部分的纯JID，由于与定向表示有关的临时“认证”是基于全JID的，这个全JID来自于用户想检测实体发送的定向表示。当用户服务器接收到一个检测到时候，它应先申请任意一个与表示订阅相关的逻辑，如8.3.2描述的那样。如果检测的实体对用户的表

示没有订阅，那么服务器应监测用户是否已经给它当前会话的实体发送了定向表示；如果是，服务器应只用类型为“available”或“unavailable”的表示答复检测（如不包含子元素），而且只针对全JID。（如不针对任何其他的当前可能与用户纯JID相关的资源）。

8.7 表示语法

8.7.1 类型属性

没有“type”属性表示相关的实体对通信是可见的。

值为“unavailable”的“type”属性表示相关的实体对通信是不可用的。

XMPP表示节也被用于针对其他实体的表示的协商和管理订阅。这些任务都通过类型为“subscribe”、“unsubscribe”、“subscribed”和“unsubscribed”的表示节来完成。

如果用户和联系人与不同的XMPP服务器相连，那么那些服务器也是用类型为“probe”的特殊表示节来判断对方服务器上实体的可用性。客户端不能发送类型为“type”的表示节。

“type”属性的值总结如下：

- error——在处理一个以前发送的表示节是发生了错误；如果表示节的类型为“error”，那么它应包含一个<error/>子元素。
- probe——对实体当前表示的请求；应该只能由代表用户的服务器生成。
- subscribe——发送者希望订阅接收者的表示。
- subscribed——发送者批准接收者接收他们的表示。
- unavailable——发送者不再对通信可用。
- unsubscribe——发送者对来自接收者的表示取消订阅。
- unsubscribed——请求订阅已经被拒绝了或者先前的订阅已经被取消了。

如果‘type’属性的值不在以上范围内，接收者或中间路由器应该返回一个<request/>的节错误。

实现注意事项：<presence/>元素中的“type”属性没有默认值。

实现注意事项：<presence/>元素中的“type”属性没有“available”值。

8.7.2 子元素

按照声明的默认名字空间，表示节的名字空间限制为‘jabber:client’或‘jabber:server’，这些表示节定义了一些子元素，尤其是<show/>、<status/>和<priority/>元素。这些子元素被用于提供关于实体的可用性的更多的细节。通常这些子元素只有在表示节没有‘type’属性时被包含，尽管在下文中会包含例外的情况。

8.7.2.1 Show元素

可选的<show/>元素指定了一个实体或一个特定的资源的可用性子状态。一个表示节不能包含多于一个的<show/>元素。没有为<show/>元素定义属性。<show/>元素不能包含混合的内容（如[XML]第3.2.2节所定义）。<show/>元素中的XML特征的数据并不意味着针对人类用户表达。XML特征的数据应是以下的一个（额外的可用性状态可以通过扩展的内容元素定义）。

away——实体或资源暂时的离开。

chat——实体或资源对聊天感兴趣。

dnd——实体或资源正忙（dnd=“Do Not Disturb”）。

xa——实体或资源离开一段时间（xa=“Extended Away”）。

如果没有<show/> 元素，那么就假设实体是在线而且可见的。

接收者和中间路由器对可用性状态的任何特殊处理由实现负责（如将可用性状态编入节中进行逻辑的转发或路由）。

8.7.2.2 Status元素

可选的<status/>元素包含人类可读的XML字符的数据，这种数据是用自然语言描述的指定实体的可用性。通常在表示节没有“type”属性时，联合show元素用户提供对一个可用性状态的描述（如“在一个会议中”）。

```
<presence from='romeo@example.net/orchard'
          xml:lang='en'>
  <show>dnd</show>
  <status>Wooing Juliet</status>
</presence>
```

没有为<status/>元素定义属性，除了从XML中继承的‘xml:lang’属性除外。<status/>元素不能包含混合的内容。<status/>元素的多个实例可能会被包括，但是每个实例拥有一个不同语言值的‘xml:lang’属性（或者是明确的或者是从更高的XML层次中的‘xml:lang’值继承来的，从发送者的角度来看更高的XML层次可以包含XML流头部）。

```
<presence from='romeo@example.net/orchard'
          id='jx62vs97'
          xml:lang='en'>
  <show>dnd</show>
  <status>Wooing Juliet</status>
  <status xml:lang='cs'>Dvořák se Julii</status>
</presence>
```

类型为“unavailable”的表示节可能也包含<status/>元素，这个元素能够提供关于实体离线的详细原因。

```
<presence from='romeo@example.net/orchard'
          id='oy6sb241'
          type='unavailable'
          xml:lang='en'>
  <status>Busy IRL</status>
</presence>
```

<status/>子元素可能在订阅相关的表示节中被发送（如类型为“subscribe”、“subscribed”、“unsubscribe”或“unsubscribed”），用来对行为的描述。交互式的客户端将这种<status/>信息呈现给人类用户。

```
<presence from='romeo@example.net'
          id='uc51xs63'
          to='nurse@example.com'
```



```

        type='subscribe'>
        <status>Hi, Juliet told me to add you to my buddy list.</status>
    </presence>

```

8.7.2.3 Priority元素

可选的<priority/>元素包含非人类可读的并指示资源优先级的XML字符数据。这个值应是-128到127之间的整数。一个表示节不能包含超过一个<priority/>元素。<priority/>元素没有属性定义。<priority/>元素不能包含混合的内容（如[XML]第3.2.2节所定义的）。

```

<presence xml:lang='en'>
    <show>dnd</show>
    <status>Wooing Juliet</status>
    <status xml:lang='cs'>Dvořák se Julii</status>
    <priority>1</priority>
</presence>

```

如果没有提供优先级，处理服务器或客户端应把优先级当成零。

客户端服务器可能通过客户端覆盖提供的优先级的值（如为了利用传递给账户所有可用资源的纯JID信息的处理规则）。如果服务器这样做的话，在它向自己回应客户端的表示以及向用户联系人发送表示通告的时候，它就应传达修改的优先级值。（由于这个修改的优先级值通常是默认为零，传递修改的优先级值可以通过不包含<priority/>子元素来实现。）

更多关于即时通信和表示应用中的节处理的语法信息。

8.7.3 扩展内容

一个XML节可能包含被名字空间所限定的任意子元素；这也适用于表示节。

（在以下的例子中，表示节包含XSF XEP-0115中定义的实体权限。）

```

<presence from='romeo@example.net'>
    <c xmlns='http://jabber.org/protocol/caps'
        hash='sha-1'
        node='http://psi-im.org'
        ver='q07IKJEyJvHSyhy//CH0CxmKi8w='/>
</presence>

```

任何在表示节中包含的扩展内容应该代表实体的通信可用性或提供关于通信相关功能的信息。

9 交换消息

9.1 概述

一旦一个客户端已经通过服务器认证并绑定了一个到XML流的资源，一个XMPP服务器将要“从”或“向”客户端路由XML节。一种能够被交换的节是<message/>（如果服务器的消息传递功能有效）。消息交换是XMPP的一个基本应用，并在用户生成发往其他实体的消息节时出现。如第12章定义的，发送者服务器对转发给接收者的信息（如果接收者在同一个服务器上）或路由给接收者服务器的信息（如果接收者在一个远程服务器能够）负责。因此，一个消息节用于将信息“推入”另一个实体。

9.2 一对一的聊天会话

实际上，用户之间的即时通信活动常常以突发对话的形式出现，这种对话我们称为“聊天会话”：在一个相对短的时间内两个会话参与者之间迅速的、多重的信息交换。

当一个人想要和一个联系人从事聊天会话时（而不是发送单一的信息给不会回复的实体），由用户的客户端生成的消息类型应该是“chat”并且联系人服务器应该保存后面回复的消息类型。用户的客户端应该在它的初始化信息中包含一个<thread/>元素，联系人客户端也应该在会话期间保存这个初始化信息。

用户客户端应该将聊天期间的初始化消息的地址写为联系人的纯JID <contact@domainpart>（而不是尝试通过任何已经从联系人接收到的表示通告的<show/>，<status/>或<priority/>的值来猜测合适的全JID<contact@domainpart/resourcepart>）。直到用户的客户端接收到来自联系人的回复，它才向联系人的纯JID发送更进一步的信息。联系人客户端应该向用户的全JID<user@domainpart/resourcepart>回复，这个全JID在联系人的回复的‘from’地址中获取并“锁定”。客户端应该在接收到来自任何其他的被对方控制的资源（或来自锁定的资源的表示节）的<message/>或<presence/>节的时候“解锁”；最后，在会话期间应该将它下一个信息发送给对方的纯JID（需将先前的“锁定”和“解锁”），直到它接收到一个来自对方全JID的信息。

当两个当事人在进行聊天会话但是没有相互共享基于订阅的表示时，他们应该发送定向表示给对方，以至于双方当事人都能轻易的发现在聊天会话期间对方是否离线。然而，客户端应为用户提供一种方式来使此类全局表示共享无效，或者只对特定的实体有效。此外，一个当事人应该在他有理由相信聊天会话已经结束的时候向对方发送定向的不可用表示（如在一段合理的时间以后，两个当前用户之间没有后续的消息交互）。

在下面第11章中提供了一个会话的例子。

9.3 消息语法

9.3.1 To 属性

即时通信客户端通过在<message/>节的“to”属性中提供预定接收者的JID来为消息指定一个预定的接收者。

如果消息被发送到任何现有的聊天会话或接收的信息之外，那么“to”地址值的形式应该是<localpart@domainpart>，而不是<localpart@domainpart/resourcepart>。

```
<message
  from='juliet@example.com/balcony'
  id='ktx72v49'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

如果消息是作为对一个以前接收到的地址形式为<localpart@domainpart/resourcepart>的信息的回复（如在9.2中描述的一对一聊天会话的场景下），“to”地址值的形式应该是

<localpart@domainpart/resourcepart>而不是<localpart@domainpart>,除非发送者已经知道预定接收者的资源不再可用。

```
<message
  from='romeo@example.net/orchard'
  id='sl3nx51f'
  to='juliet@example.com/balcony'
  type='chat'
  xml:lang='en'>
  <body>Neither, fair saint, if either thee dislike.</body>
</message>
```

9.3.2 Type 属性

在即时通信应用中的消息节的一般用途是：单条消息；在一对一聊天会话中的消息‘在多人聊天室中的消息发送’警报、通知或其他没有期望回复的信息；以及错误。这些用户通过“type”属性进行区分。建议包含“type”属性。如果包含，“type”属性值应为以下值中的一个。

- chat——这种消息在一对一聊天会话中发送。通常一个交互式客户端要在两个当前用户之间的一对一聊天界面中呈现一个类型为“chat”的信息，包括一个合适的会话历史。详细的关于一对一聊天会话的推荐规定见 9.1。

- error——这种消息在处理从另一个实体接收到的消息是发送错误的时候，由实体自己产生。一个接收到类型为“error”的消息的客户端应该呈现出一个合适的界面来通知源发送者错误的性质。

- groupchat——这种信息在多用户聊天的情况下发送（与 RFC1459 里面的相似）。通常的一个接收客户端要在一个支持多对多聊天的界面中呈现类型为“groupchat”的信息，包括一个聊天室中当事人的名册和一个合适的会话历史。关于 XMPP 的群组聊天的细节，请参考 XEP-0045。

- 标题——这种信息向不期望回复的对象提供警报、通告或其他暂时的信息（如新闻标题、体育更新、准实时的市场数据或企业内容）。由于没有期待对这些信息进行回复，通常一个接收客户端要在一个能合适区分单独消息、聊天消息和群组消息的界面中呈现类型为“headline”的信息（如不赋予接收者回复的能力）。如果‘to’地址是纯 JID，接收服务器应该向接收者所有具有高优先级的可用资源转发这种信息，而且应向至少一个这种资源转发；如果‘to’地址是全 JID 而且有一个相匹配的资源，服务器应向那个资源转发那个消息；否则服务器应无声的忽略这条消息或返回一个错误（见第 12 章）。

- 正规的——这是一个发送一对一会话或群组聊天之外的单独的消息，而且这种消息的接收者需要回复。通常接收客户端要在一个接收回复有效的界面中呈现出类型为“normal”的消息，但是没有会话历史。“type”属性的默认值为“normal”。

一个即时通信应用程序应支持所有上文提到的消息类型。如果一个接收到一条没有“type”属性的消息，或者应用程序不理解‘type’属性提供的值，那么它应将这条消息认为是“normal”类型（默认为“normal”）。

服务器处理不同消息类型的描述在第12章中提供。

尽管“type”属性是可选的，它被认为是对任何信息的回复的自然的反映；而且，在一些专门的应用中可能会灵活的使用一种特定的消息类型（如type='groupchat'）。

9.3.3 Body 元素

`<body/>` 元素包含人类可读的XML字符数据，这些数据指定了信息的文本内容；这个子元素通常被包含，但是是可选的。

```
<message
  from='juliet@example.com/balcony'
  id='b4vs9km4'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <body>Wherefore art thou, Romeo?</body>
</message>
```

没有为`<body/>`元素定义属性，但是‘`xml:lang`’属性除外。为了从一个body中提供可选的版本，在一个消息节中可能包含多个`<body/>`元素，但是每个元素都拥有一个有明确语言值的‘`xml:lang`’属性。

```
<message
  from='juliet@example.com/balcony'
  id='z94nb37h'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <body>Wherefore art thou, Romeo?</body>
  <body xml:lang='cs'>
    Pro&#x010D;e&#x017D;jsi ty, Romeo?
  </body>
</message>
```

`<body/>`元素不能包含混合的内容。

9.3.4 Subject 元素

`<subject/>`元素包括人类可读的指定了信息主题的XML字符数据。

```
<message
  from='juliet@example.com/balcony'
  id='c8xg3nf8'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <subject>I implore you!</subject>
  <body>Wherefore art thou, Romeo?</body>
</message>
```


没有为<subject/>元素定义的属性，除了从[XML]中继承来的‘xml: lang’属性。为了给同一个主题提供可选的版本，可能包含多个<subject/>元素实例，但是每个实例都拥有一个明显的语言版本值的‘xml: lang’属性。

```
<message
  from='juliet@example.com/balcony'
  id='jk3v47gw'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <subject>I implore you!</subject>
  <subject xml:lang='cs'>
    &#x00DA;p&#x011B;niv&#x011B;pros&#x00ED;m!
  </subject>
  <body>Wherefore art thou, Romeo?</body>
  <body xml:lang='cs'>
    Pro&#x010D;e&#x017E;jsi ty, Romeo?
  </body>
</message>
```

<subject/>元素不能包含混合的内容（如[XML]第3.2.2节所定义）。

9.3.5 Thread 元素

XMPP<thread/>元素的主要用途是唯一的表示一个会话线程或在两个类型为‘chat’的<message/>节实体化的实体之间的“聊天会话”。但是，XMPP的<thread/>元素也可能被用于标识类型为‘headline’或‘normal’的<message/>节的两个实体之间一个类似的线程，或者是在多实体之中的线程，这个线程在多用户聊天室的场景下被定义为“groupchat”的<message/>节所实例化类型。它也可能被用于与人类会话无关的<message/>节，比如说游戏会话或插件之间的交互。<thread/>元素不用于标识单个的消息，只有对话或消息会话。

对<thread/>元素的包含是可选的。由于<thread/>元素标识了一个特殊的会话线程，这个会话线程是一个消息隶属的，一个消息节不能超过一个<thread/>元素。

<thread/>元素可能拥有‘parent’属性，这个属性标识了分支或子元素为当前线程的另一个线程。‘parent’属性应符合另一个<thread/>元素的语法，并且它的值应与包含‘parent’属性的<thread/>元素的XML字符数据不同。

实现要求：指定一个父线程和一个子线程在覆盖线程的时候引入了线程标识符之间冲突的可能性。例如，一个<thread/>元素可能包含“foo”和值为“bar”的“parent”属性的XML字符数据，第二个<thread/>元素可能包含“bar”和“parent”属性值为“baz”的XML字符数据，第三个<thread/>元素可能包含“baz”和“parent”属性值为“foo”的XML字符数据。对于如何处理那些覆盖线程标识符之间的冲突由实现来处理（如是否会将线程标识符连在一起，通过多级用户界面中显示“foo”，或是否在一个时间里只显示从属的一级）。

<thread/>元素的值为人类不可读的，而且对实体来说应是不透明的；不能从它推出语义，而且只有精确的比较才能对它不利。<thread/>元素的值应是唯一标识当前用户会话之间的或更一般的会话线程（一种确保唯一性的方法是通过生成一个如[UUID]所述的唯一的标识符（UUID））。

安全性注意事项：生成线程ID的应用程序应确保不显示关于实体的标识信息（如运行XMPP应用程序的设备的MAC地址）。

<thread/>元素不能包含混合的内容。

```
<message
  from='juliet@example.com/balcony'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <subject>I implore you!</subject>
  <subject xml:lang='cs'>
    &#x00DA;p&#x011B;niv&#x011B;pros&#x00ED;m!
  </subject>
  <body>Wherefore art thou, Romeo?</body>
  <body xml:lang='cs'>
    Pro&#x010D;e&#x017E; jsi ty, Romeo?
  </body>
  <thread parent='e0ffe42b28561960c6b12b944a092794b9683a38'>
    0e3141cd80894871a68e6fe6b1ec56fa
  </thread>
</message>
```

关于<thread/>元素使用的详细的规范，请参见XEP-0201。

9.4 扩展的内容

XML节可能包含除了被默认名字空间限制的任何子元素；这也适用于消息节。

（在以下的例子中，消息节包括XHTML格式的消息，如XSF XEP-0071所定义）

```
<message
  from='juliet@example.com/balcony'
  to='romeo@example.net'
  type='chat'
  xml:lang='en'>
  <body>Wherefore art thou, Romeo?</body>
  <html xmlns='http://jabber.org/protocol/xhtml-im'>
    <body xmlns='http://www.w3.org/1999/xhtml'>
      <p>Wherefore <span style='font-style: italic'>art</span>
      thou, <span style='color:red'>Romeo</span>?</p>
    </body>
  </html>
</message>
```

```

    </body>
  </html>
</message>

```

10 交换 IQ 节

IQ节提供结构化的请求-响应机制。鉴于要求完成特定的语义，在一切情况下通过扩展名字空间来定义特殊的用法，这种名字空间准许类型为“get”或“set”的IQ节的直接子元素。‘jabber:client’或‘jabber:server’对于所有节类型的名字空间不定义除了<error/>元素的任何IQ子元素。本标准为了管理名册，定义了一种这样的扩展名字空间。然而，一个IQ节可能包含受扩展名字空间限制的结构化信息。

11 一个示例会话

本章的例子阐明了一个可能的即时消息传递和表示会话。用户是<romeo@example.net>，它有一个资源域为“orchard”的可用资源，在它的名册中有以下几个用户：

<juliet@example.com>（subscription=“both”，并且它有两个可用的资源，分别为“chamber”和“balcony”）

<benvolio@example.net>（subscription=“to”）

<mercutio@example.org>（subscription=“from”）

首先，用户完成YD/T 2935《扩展消息与表示协议（XMPP）核心协议》规范的前提条件（流建立，TLS和SASL协商和资源绑定）；这些协议流在本章中不再赘述。

然后，用户请求它的名册。

本标准示例中的UC表示用户到客户端；US表示用户到服务器；CS表示客户端到服务器；CC表示客户端到客户端。

示例1：用户向服务器请求目前的名册。其中 UC 表示用户向服务器。

```

UC: <iq from='romeo@example.net/orchard'
    id='hf61v3n7'
    type='get'>
  <query xmlns='jabber:iq:roster'/>
</iq>

```

示例2：用户从服务器接收名册。其中 US 表示用户从服务器。

```

US: <iq id='hf61v3n7'
    to='romeo@example.net/orchard'
    type='result'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@example.com'
      name='Juliet'
      subscription='both'>
      <group>Friends</group>
    </item>
  </query>
</iq>

```

```

    <item jid='benvolio@example.org'
      name='Benvolio'
      subscription='to' />
    <item jid='mercutio@example.org'
      name='Mercutio'
      subscription='from' />
  </query>
</iq>

```

现在用户开始表示会话。

示例3：用户发送初始表示信息。

UC: <presence/>

示例4：用户的服务器代表用户发送表示检测信息与 subscription="to"和 subscription="both"联系。

US: <presence

```

  from='romeo@example.net'
  to='juliet@example.com'
  type='probe' />

```

US: <presence

```

  from='romeo@example.net'
  to='benvolio@example.org'
  type='probe' />

```

示例5：用户的服务器代表用户可用的资源发送初始表示信息与 subscription="from"和 subscription="both"联系，这点和用户一样。

US: <presence

```

  from='romeo@example.net/orchard'
  to='juliet@example.com' />

```

US: <presence

```

  from='romeo@example.net/orchard'
  to='mercutio@example.org' />

```

US: <presence

```

  from='romeo@example.net/orchard'
  to='romeo@example.net' />

```

示例6：联系人的服务器代表所有可用的资源回复表示探测。

CS: <presence

```

  from='juliet@example.com/balcony'
  to='romeo@example.net'

```



```

    xml:lang='en'>
<show>away</show>
<status>be right back</status>
<priority>0</priority>

```

CS: <presence

```

    from='juliet@example.com/chamber'
    to='romeo@example.net'>
    <priority>1</priority>
</presence>

```

CS: <presence

```

    from='benvolio@example.org/pda'
    to='romeo@example.net'
    xml:lang='en'>
    <show>dnd</show>
    <status>gallivanting</status>
</presence>

```

</presence>

示例7：联系人的服务器递送用户的初始表示信息到所有可用的资源。

CS: <presence

```

    from='romeo@example.net/orchard'
    to='juliet@example.com' />

```

CS: <presence

```

    from='romeo@example.net/orchard'
    to='juliet@example.com' />

```

CS: <presence

```

    from='romeo@example.net/orchard'
    to='mercutio@example.org' />

```

示例8：用户向不在它名册中的另一个用户发送定向的表示信息。

UC: <presence

```

    from='romeo@example.net/orchard'
    to='nurse@example.com'
    xml:lang='en'>
    <show>dnd</show>

```

```
<status>courting Juliet</status>
```

```
<priority>0</priority>
```

```
</presence>
```

现在用户和它的一个联系人参与到聊天会话中。其中CC表示客户端到客户端。

示例9：一个线程对话

CC: <message

```
  from='juliet@example.com/balcony'
```

```
  to='romeo@example.net'
```

```
  type='chat'
```

```
  xml:lang='en'>
```

```
<body>My ears have not yet drunk a hundred words</body>
```

```
<thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
```

```
</message>
```

CC: <message

```
  from='juliet@example.com/balcony'
```

```
  to='romeo@example.net'
```

```
  type='chat'
```

```
  xml:lang='en'>
```

```
<body>Of that tongue's utterance, yet I know the sound:</body>
```

```
<thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
```

```
</message>
```

CC: <message

```
  from='juliet@example.com/balcony'
```

```
  to='romeo@example.net'
```

```
  type='chat'
```

```
  xml:lang='en'>
```

```
<body>Art thou not Romeo, and a Montague?</body>
```

```
<thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
```

```
</message>
```

UC: <message

```
  from='romeo@example.net/orchard'
```

```
  to='juliet@example.com/balcony'
```

```
  type='chat'
```

```
  xml:lang='en'>
```

```

<body>Neither, fair saint, if either thee dislike.</body>
<thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
</message>

```

CC: <message

```

    from='juliet@example.com/balcony'
    to='romeo@example.net/orchard'
    type='chat'
    xml:lang='en'>

```

```

<body>How cam'st thou hither, tell me, and wherefore?</body>
<thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
</message>

```

依次类推。

用户也可以发送后续的表达广播。

示例10：用户为广播发送最新可用的表示信息。

UC: <presence xml:lang='en'>

```

    <show>away</show>
    <status>I shall return!</status>
    <priority>1</priority>
</presence>

```

示例11：用户的服务器向订阅类型为“both”或“from”的联系人广播最新的表示信息，其中不包括那些服务器向其发送定向表示信息的实体。

US: <presence

```

    from='romeo@example.net/orchard'
    to='juliet@example.com'
    xml:lang='en'>

```

```

<show>away</show>
<status>I shall return!</status>
<priority>1</priority>
</presence>

```

US: <presence

```

    from='romeo@example.net/orchard'
    to='mercutio@example.org'
    xml:lang='en'>

```

```

<show>away</show>
<status>I shall return!</status>
<priority>1</priority>

```

</presence>

示例12: 联系人的服务器递送最新的表示信息。

```
CS: <presence
    from='romeo@example.net/orchard'
    to='juliet@example.com'
    xml:lang='en'>
    <show>away</show>
    <status>I shall return!</status>
    <priority>1</priority>
</presence>
```

```
CS: <presence
    from='romeo@example.net/orchard'
    to='juliet@example.com'
    xml:lang='en'>
    <show>away</show>
    <status>I shall return!</status>
    <priority>1</priority>
</presence>
```

```
CS: <presence
    from='romeo@example.net/orchard'
    to='mercutio@example.org'
    xml:lang='en'>
    <show>away</show>
    <status>I shall return!</status>
    <priority>1</priority>
</presence>
```

示例13: 联系人的一个资源广播不可用通知。

```
CC:<presence from='juliet@example.com/chamber' type='unavailable' />
```

示例14: 联系人的服务器向用户发送不可用通知。

```
CS: <presence
    from='juliet@example.com/chamber'
    to='romeo@example.net'
    type='unavailable' />
```

现在用户结束了它的会话表示。

示例15: 用户发送不可用通知。

```
UC: <presence type='unavailable' xml:lang='en'>
```



```

    <status>gone home</status>
  </presence>

```

示例16: 用户的服务器向联系人广播不可用通知, 其中联系人也包括那些用户向其发送定向表示信息的实体。

```

US: <presence
    from='romeo@example.net/orchard'
    to='juliet@example.com'
    type='unavailable'
    xml:lang='en'>
    <status>gone home</status>
  </presence>

```

```

US: <presence
    from='romeo@example.net/orchard'
    to='mercutio@example.org'
    type='unavailable'
    xml:lang='en'>
    <status>gone home</status>
  </presence>

```

```

US: <presence
    from='romeo@example.net/orchard'
    to='nurse@example.com'
    type='unavailable'
    xml:lang='en'>
    <status>gone home</status>
  </presence>

```

最后用户关闭流并且服务器友好的回应。

示例17: 用户关闭流。

```
UC: </stream:stream>
```

示例18: 用户的服务器关闭流。

```
US: </stream:stream>
```

结束。

12 服务器处理 XML 节的规则

12.1 一般性事项

YD/T 2935《扩展消息与表示协议(XMPP) 核心协议》中给出了节递送的一般性事项, 特别是在提供一个可接受的节递送服务水平和防止收到目录攻击与表示泄露两者之间的权衡。然而, 目录受到攻击的概念不适用于用户知道并且信任联系人的情况(因为联系人在用户的名册中)。类似的, 表示泄露

的概念也不适用于当联系人有权知道（通过第7章描述的表示订阅的方法）用户的表示或者用户自愿的发送表示信息给一个实体（通过8.6中描述的发送定向的表示信息的方法）。因此，在以下的章节，当防止目录受到攻击与表示泄露时提供了两种选择：（1）默认忽略一个节；（2）返回一个错误，当初始实体在用户的名册中时（此时错误会表明用户的账户是否存在）；或者当其有权限收到用户的表示信息时；或者当其从用户那里收到定向的表示信息时（此时错误会表明用户资源的表示），服务器应当返回一个错误。

安全性注意事项：下面章节描述的所有的节处理规则是建立在它们会受到有关隐私和安全政策的保护的理解基础上的，例如那些通过XSF XEP-0016和XSF XEP-0191部署的节。以下章节中的一致性语言并不意味着覆盖任何本地的服务策略。

12.2 没有“to”地址

如果处理的节没有“to”属性，在YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》中定义的规则适用。

12.3 远程域

如果出站节的“to”属性中包含的地址域和服务器本身配置的域不匹配，那么YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》中定义的规则适用。

兼容性要求：IETF RFC6121详细的说明了怎样通过`_im._xmpp`和`_pres._xmpp` SRV记录[IMP-SRV]来作为一种发现远程的即时消息和表示消息是否在通过XMPP进行通信的反馈方法。因为那些SRV记录没有被广泛的部署，所以本标准没有明确指出其用途并且不支持新的实现。

12.4 本地域

如果“to”属性中包含的JID的地址域匹配服务器配置域中的某一个，那么这个域由服务器自身服务（不是通过当地指定的服务器），并且JID的形式为`<domainpart>` 或者`<domainpart/resourcepart>`，YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》中定义的规则适用。

12.5 本地用户

12.5.1 概述

如果“to”地址具体指定一个纯JID如`<localpart@domainpart>` 或者全JID如`<localpart@domainpart/resourcepart>`，并且JID的域匹配服务器自身服务的配置域时，服务器应按照以下规则处理。

12.5.2 没有这样的用户

如果“to”属性鉴定出用户的账户不存在时，节的处理规则取决于节的类型。

对于一个IQ节，服务器应向发送者返回一个`<service-unavailable/>`节错误。

对于一个信息节，服务器应默认忽略这条信息或者向发送者返回一个`<service-unavailable/>`节错误。

对于一个没有“type”属性或者“type”属性为“unavailable”的表示节，服务器应默认忽略这个节。

对于类型为“subscribe”、“subscribed”、“unsubscribe”或者“unsubscribed”的表示节，服务器应默认忽略这个节。

对于类型为“probe”的表示节，服务器应默认忽略这个节或者返回一个类型为“unsubscribed”的表示节。

12.5.3 本地部分@域部分

12.5.3.1 可用的或已连接的资源

如果至少有一个可用的或连接的资源，节的处理规则取决于节的类型。

12.5.3.1.1 消息

对于类型为“normal”的信息节：

如果所有可用资源的表示优先级为负，那么服务器应该存储离线信息以便稍后的递送或者向发送者返回一个节错误，此节错误的类型应该为<service-unavailable/>。

如果其中一个可用资源的表示优先级为非负，那么服务器应向此资源递送本条信息。

如果不只一个资源的表示优先级为非负，那么服务器应递送本条信息到最可用的资源（按照服务器实现特定的算法处理的资源，例如把具有最高表示优先级的资源当做最可用的资源）；或者递送本条信息到所有非负优先级的资源。

对于类型为“chat”的信息节：

如果唯一可用资源的表示优先级为负，那么服务器应该存储离线信息以便稍后的递送或者向发送者返回一个节错误，此节错误的类型应该为<service-unavailable/>。

如果唯一可用资源的表示优先级为非负，那么服务器应向此资源递送本条信息。

如果不只一个资源的表示优先级为非负，那么服务器应递送此信息到最可用的资源（按照服务器实现特定的算法处理的资源，例如把具有最高表示优先级的资源当做最可用的资源）；或者递送此信息到所有具有非负优先级的、在接收聊天信息时选择的资源。

对于类型为“groupchat”的信息节，服务器不能递送此信息到任何可用的资源，但是应向发送者返回一个节错误，节错误的类型应该为<service-unavailable/>。

对于类型为“headline”的信息节：

如果唯一可用资源的表示优先级为负，那么服务器应默认忽略这个节。

如果唯一可用资源的表示优先级为非负，那么服务器应向这个资源递送此信息。

如果不只一个资源的表示优先级为非负，那么服务器应递送此信息到所有优先级为非负的资源。

对于类型为“error”的信息节，服务器应默认忽略这个信息节。

然而，对于任何类型的信息，服务器不能递送信息节到任何优先级为负的可用资源；如果唯一可用资源的优先级为负，那么服务器应该按照没有可用或者连接的资源的情况处理此信息。

在所有的情况下，服务器不能重写“to”属性（如应让它保持为<localpart@domainpart>，而不是把它改写为<localpart@domainpart/resourcepart>）。

12.5.3.1.2 表示

对于一个没有类型或者类型为“unavailable”的表示节，服务器应递送它到所有可用的资源。

对于一个类型为“subscribe”、“subscribed”、“unsubscribe”或者“unsubscribed”的表示节，服务器应遵守第7章中定义的和附录A中总结的规则。

对于类型为“probe”的表示节，服务器应直接处理它。

在所有的情况中，服务器不能重写“to”属性（如应让它保持为<本地部分@域部分>，而不是把它改写为<本地部分@域部分/资源部分>）。

12.5.3.1.3 IQ

对于一个IQ节，服务器本身应代表用户回复一个IQ结果或者一个IQ错误，并且不能递送此IQ节到任何用户可用的资源。特别的，当资源名空间的语法规则定义服务器可以代表用户提供回复时，服务器应代表用户回复IQ节一个类型为“result”的IQ节或者一个类型为“error”的适合原始负载的IQ节；反之，服务器应回复一个属性为<service-unavailable/>的节错误。

12.5.3.2 没有可用的或已连接的资源

如果和用户关联的没有可用或者连接的资源，节的处理取决于节的类型。

12.5.3.2.1 消息

对于类型为“normal”或者“chat”的信息节，服务器应该增加信息到离线存储或者向发送者返回一个节错误，错误的类型应该为<service-unavailable/>。

对于类型为“groupchat”的信息节，服务器应向发送者返回一个错误，错误的属性为<service-unavailable/>。

对于类型为“headline”或者“error”的信息节，服务器应默认忽略这个节。

12.5.3.2.2 表示

对于没有类型或者类型为“unavailable”的表示节，服务器应该默认忽略这个节，不为以后的递送存储它并且不代表用户回复它。

对于类型为“subscribe”、“subscribed”、“unsubscribe”或者“unsubscribed”的表示节，服务器应遵守第7章中定义的和附录A中总结的规则。

对于类型为“probe”的表示节，服务器应直接处理它。

12.5.3.2.3 IQ

对于一个IQ节，服务器本身应代表用户回复一个IQ结果或者一个IQ错误。特别的，当资源名空间的语法规则定义服务器可以代表用户提供回复时，服务器应代表用户回复节一个类型为“result”的IQ节或者一个类型为“error”的适合原始负载的IQ节；反之，服务器应回复一个属性为<service-unavailable/>的节错误。

12.5.4 本地部分@域部分/资源部分

12.5.4.1 概述

如果一个入站节的“to”属性中包含的JID域匹配服务器本身的域并且“to”属性中包含的JID的形式为<localpart@domainpart/resourcepart>，那么服务器应遵守以下规则。

12.5.4.2 资源配置

如果可用或者连接的资源可以匹配全JID，节的处理取决于节的类型。

对于一个类型为“get”或者“set”的IQ节，如果预定的接收者没有和请求实体共享订阅类型为“both”或“form”或者定向的表示信息，那么服务器不应该递送IQ节但是应该返回一个<service-unavailable/>节错误给请求实体。这种策略可以帮助防止表示泄露。

如果类型为“result”或者“error”的IQ节，服务器应向资源递送这个节。

对于信息节，服务器应向资源递送这个节。

对于类型为“subscribe”、“subscribed”、“unsubscribe”或者“unsubscribed”的表示节，服务应遵守第7章中提供的准则。

对于类型为“probe”的表示节，服务器应遵守8.3中提供的准则。

12.5.4.3 无匹配资源

如果没有可用或连接的资源匹配全JID，那么节的处理取决于节的类型。

12.5.4.3.1 消息

对于类型为“normal”、“groupchat”或者“headline”的信息节，服务器应默认忽略这个节或者向发送者返回一个节错误，错误的属性应该为<service-unavailable/>。

对于类型为“chat”的信息节：

如果没有可用或者连接的资源，那么服务器应存储离线信息以便稍后的递送或者向发送者返回一个节错误，错误的属性应该为<service-unavailable/>。

如果所有可用资源的表示优先级为负，那么服务器应该存储离线信息以便稍后的递送或者向发送者返回一个节错误，此节错误的类型为<service-unavailable/>。

如果其中有一个可用资源的表示优先级非负，那么服务器应向这个资源递送信息。

如果不只一个资源的表示优先级为非负，那么服务器应递送此信息到最可用的资源（按照服务器实现特定的算法处理的资源，例如把具有最高表示优先级的资源当做最可用的资源）；或者递送此信息到所有非负优先级的、在接收聊天信息时选择的资源。

对于类型为“error”的信息节，服务器应默认忽略这个节。

12.5.4.2.2 表示

对于没有类型或者类型为“unavailable”的表示节，服务器应默认忽略这个节。

对于类型为“subscribe”的表示节，服务器应遵守7.1.3中提供的规则。

对于类型为“subscribed”、“unsubscribed”或者“unsubscribed”的表示节，服务器应忽略这个节。

对于类型为“probe”的表示节，服务器应遵守8.3中提供的准则。

12.5.4.2.3 IQ

对于一个IQ节，服务器应向发送者返回一个<service-unavailable/>节错误。

12.5.5 信息递送规则汇总

表1对前面章节中描述的信息（不是节）递送规则进行了汇总。左边的列表示各种条件的组合（不存在的账户，不活跃的资源，唯一并且是负优先级的资源，唯一的并且是非负优先级的资源，或者是不只一个资源并且每一个都有非负的表示优先级）和“to”地址（纯JID，全JID匹配一个可用的资源，或者全JID不匹配可用资源）。后续列列出了4种主要的信息类型（normal，chat，groupchat，headline）和6种可能的递送选择：存储离线信息（O），返回这条信息并附带节错误（E），默认忽略这条信息（S），向“to”地址中指定的资源递送这条信息（D），递送此信息到最可用的资源（按照服务器实现特定的算法处理的资源，例如把具有最高表示优先级的资源当做最可用的资源）（M），或者递送信息到所有非负表示优先级的资源（A—对聊天信息来说所有资源指的是那些为接收每个聊天信息而明确选择出来的资源集合）。符号“/”代表“异或”。服务器在选择对特定的信息做出什么反应时应该遵守12.1节给出的规则。

表1 信息递送规则

条件	正常	聊天	组聊	标题
账户不存在				
纯JID	S/E	S/E	E	E
全JID	S/E	S/E	S/E	S/E
账户存在，但没有活跃的资源				
纯JID	O/E	O/E	E	S
全JID（不匹配）	S/E	O/E	S/E	S/E

表1（续）

条件	正常	聊天	组聊	标题
1+ 负的资源，但是非负资源为零				
纯JID	O/E	O/E	E	S
全匹配	D	D	D	D
完全不匹配	S/E	O/E	S/E	S/E
1 非负资源				
纯JID	D	D	E	D
全匹配	D	D	D	D
完全不匹配	S/E	D	S/E	S/E
1+ 非负资源				
纯JID	M/A	M/A*	E	A
全匹配	D	D/A*	D	D
完全不匹配	S/E	M/A*	S/E	S/E

*为类型是“chat”的信息，服务器不应该按照选项A反应，除非客户端明确选择要接收所有的聊天信息；然而，选择的方法不在本标准内。

13 URIs 的处理

通过XMPP网络进行通信的XMPP实体地址（如在XML节中的“form”和“to”地址）不能在前面加上统一资源标识符（URI）。

然而，XMPP本身外部的应用（例如：万维网上的一张网页）可能需要通过URI或者是国际化资源标识符（IRI）来辨别一个XMPP实体，并且一个XMPP客户端可能需要与外部的应用（如一个XMPP客户端可能通过点击网页来发出恳求）交互操作。在这种互操作的场景下，XMPP客户端批准处理按照XMPP编码的URIs和IRIs地址，如XEP-0147中的说明和XEP-0147中进一步的描述。尽管XMPP客户端可以处理按照‘im: ’ URIs编码（如RFC3860规定）和‘pres: ’ URIs编码（如[CPP]中规定）的地址，它们也可以去掉‘im: ’ 或者‘pres: ’ 并且委托服务器进行地址解析，按照12.3中的明确说明。

14 国际化事项

待定。

15 安全性事项

XMPP的核心安全注意事项包括信道加密、验证、信息泄露、拒绝服务攻击和域间融合的论述。

YD/T 2935《扩展消息与表示协议（XMPP） 核心协议》中概括性地描述了在XMPP典型部署中客户端和服务端的关键性角色，并论述了与上述角色相关的安全性能。这些角色对即时信息的安全、表示的订阅以及本标准中描述的表示的通知都有影响。实质上，XMPP用户在XMPP服务器上注册（或者预留）一个账户来设置不同级别的优先级以便服务器可以完成各种任务，增强安全策略等。因此，服务器有如下职责：

- （1）本地客户端和远程服务器通信时最好使用信道加密。
- （2）任何试图访问用户账户的客户端都要进行身份验证。

(3) 处理经过用户验证的客户端发送或者接收的XML节（特别是有关即时通信和表示功能，存储用户的名册，处理入站或者出站的请求订阅和回复，发出或者处理表示检测，广播出站的表示通知，路由出站信息，并且递送入站信息和表示通知）。

即使服务器完全满足上述要求，客户端仍然不能确认和其他客户端（无论是在同一个服务器还是在远程的服务器中）交换的节在XMPP通信路径或者在服务器自身的所有跳中都受到保护。如果XML节希望确保端到端的保密和通信的完整性时，客户端有责任使用一种适当的技术来加密和标记XML节。

仅适用于XMPP的即时通信和表示应用的其他注意事项在本标准内的多处定义，具体如下：

- 当服务器处理一个类型为“probe”的入站节，其中它的预定接收者是和服务器的一个配置域相关的用户时，如果发送者是一个没有权限接受信息的实体，服务器不能透露用户的表示，如表示订阅所定义；
- 用户的服务器不能向无权限知道用户表示的实体泄露用户的网络可用性。对XMPP本身而言，授权在联系人和用户之间采取一种明确的形式（在第7章中定义）。然而，当实体与发出表示信息(如果机制命令接受表示信息作为雇佣协议的一部分时，XMPP的整体部署可能自动添加用户的表示信息到雇员的私人目录)的用户之间存在可信任的关系时，一些XMPP的部署可能认为实体是获得授权的；
- 当服务器处理没有类型或者类型为“unavailable”的出站表示节时，服务器应遵守第8章中定义的规则，来确保此表示信息不被送到没有授权知道此信息的实体；
- 当元素包含在类型为“subscribe”、“subscribed”、“unsubscribe”或者“unsubscribed”的表示节中时，客户端可以忽略<status/>元素；这样可以防止“表示订阅垃圾邮件”。

16 一致性要求

本章描述了协议的功能设置，它总结了此规范的一致性要求。这个功能设置在软件认证、互操作性测试和执行情况报告中是有用的。对于每个功能，本章提供以下信息：

- 一个可读的名字
- 一个信息描述
- 对标准地定义功能文档的特定章节的一个参考
- 功能是否适用于客户端角色，服务器角色或者两者都适用（其中“N/A”表明功能不适用于指定的角色）
- 功能是必须还是应该被采用，在哪类情况下应该理解重要的术语，如[KEYWORDS]中的描述。

此处说明的功能配置试图坚持Larry Masinter2005年在IETF工作组中提出的概念和形式，如[INTEROP]中所述。尽管此功能配置比[REPORTS]描述得更为详细，它为执行报告的生成提供了一个合适的原则，此执行报告被呈送以支持本标准的进展，本标准来自于和[PROCESS]相关的建议实行标准和标准草案。

功能：信息体

描述：支持<message/>节的子元素<body/>

章节：9.3.3

角色：客户端必须，服务器端 N/A

功能：信息主体

描述：支持<message/>节的<subject/>子元素

章节：9.3.4

角色：客户端应该，服务器端 N/A

功能：信息线程

描述：支持<message/>节的<thread/>子元素

章节：9.3.5

角色：客户端应该，服务器端 N/A

功能：信息类型支持

描述：支持接收类型为“normal”、“chat”、“groupchat”、“headline”和“error”的信息

章节：9.3.2

角色：客户端应该，服务器端 N/A

功能：信息类型递送

描述：适当的递送类型为“normal”、“chat”、“groupchat”、“headline”和“error”的信息

章节：12

角色：客户端 N/A，服务器端应该

功能：无类型表示

描述：处理没有类型属性的表示节就像指出可用资源那样

章节：8.7.1

角色：客户端必须，服务器端必须

功能：表示检测

描述：当检测是否有表示信息时，发送和接收类型属性为“probe”的表示节

章节：8.7.1

角色：客户端N/A，服务器端必须

功能：表示订阅批准

描述：当批准先前从另一个实体接收的表示请求订阅时，处理类型为“subscribed”的出站节，并且当批准另一个实体订阅时处理类型为“subscribed”的入站节

章节：7.2

角色：客户端必须，服务器端必须

功能：表示取消订阅

描述：当拒绝从另一个实体接收到的请求订阅或者取消之前对另一个实体授予的订阅许可时，处理类型为“unsubscribe”的出站表示节；并且当拒绝或取消另一个实体的请求订阅时，处理类型为“unsubscribed”的入站表示节

章节：7.3

角色：客户端必须，服务器端必须

功能：表示订阅预批准

描述：在某些特定的情况下，如预批准另一个实体发出的请求订阅时，处理类型为“subscribed”的出站节；这包括在‘jabber:iq:roster’名字空间内<item/>元素的“approved”属性的支持

章节：7.5

角色：客户端可能，服务器可能

功能：表示请求订阅

描述：当向另一个实体请求表示信息的订阅时，处理类型为“subscribed”的出站表示节，并且当接收另一个实体的表示请求订阅时，处理类型为“subscribed”的入站表示节

章节：7.2

角色：客户端必须，服务器端必须

功能：取消订阅表示信息

描述：当另一个实体取消订阅表示信息时，处理类型为“unsubscribe”的出站表示节，并且当收到另一个实体的取消订阅通知时处理类型为“unsubscribe”的入站表示节

章节：7.4

角色：客户端必须，服务器端必须

功能：表示不可用

描述：当指示缺乏可用性时处理属性类型为“unavailable”的表示节

章节：8.7.1

角色：客户端必须，服务器端必须

功能：获取名册

描述：当请求重新获取服务器上某个账户的名册信息时，‘jabber:iq:roster’名空间授予了处理包含在空的<query>元素中类型为“get”的IQ节

章节：6.2.3

角色：客户端必须，服务器端必须

功能：名册推送

描述：无论何时当服务器方表明名册信息有变化时，发送一个名册推送给有关的资源，或者当收到服务器的信息时处理这个推送

章节：6.2.6

角色：客户端必须，服务器端必须

功能：名册版本

描述：作为名册信息被发送或者接收的特别版本的一个标记，‘jabber:iq:roster’ 名字空间授予了处理<query/>元素的” ver” 属性的资格

章节：6.2.1

角色：客户端应该，服务器端必须

附录 A

(规范性附录)

订阅状态

A.1 状态定义

有4种主要的订阅状态（这些状态是从用户而不是联系人的角度定义的）：

None：用户没有联系人表示的订阅，并且联系人也没有用户表示的订阅。

To：用户有联系人表示的订阅，但是联系人没有用户表示的订阅。

Form：联系人有用户表示的订阅，但是用户没有联系人表示的订阅。

Both：用户和联系人都有对方表示的订阅（是“To”和“Form”的联合）。

补充说明：为了对随后章节所描述的与订阅相关的表示节的处理，订阅状态“None”包含联系人不在用户名册中的情况，例如，用户名册中一个未知的实体。

先前的状态由各种与有关等待出站与入站的订阅状态补充，因此产生出九种可能的订阅状态：

a) “None”=联系人和用户都没有订阅对方，并且不能接受对方的请求订阅；这种状态在用户的名册中体现为订阅状态为“None”。

b) “None + Pending Out”=联系人和用户没有订阅对方，并且用户向联系人发送一个请求订阅但是联系人没有回复；这种状态在用户的名册中体现为订阅状态为“None”和回复为“subscribe”。

c) “None + Pending In”=联系人和用户没有订阅对方，并且联系人向用户发送一个请求订阅但是用户没有回复。这种状态在用户名册中有可能或者没有可能体现，如下：如果用户为联系人创建了一个名册项，那么服务器应维持这个名册项并且记录下入站表示请求订阅的存在；但是如果用户没有为联系人创建名册项，那么服务器应记录下入站表示请求订阅的存在但是不能为联系人创建名册（反而，服务器应等待直到添加联系人到用户的名册后才批准请求订阅）。

d) “None + Pending Out + In”=用户和联系人没有订阅对方，联系人向用户发送一个请求订阅但是用户没有回复，并且用户也向联系人发送一个请求订阅而联系人没有回复；这种状态在用户的名册中体现为订阅状态为“None”和回复状态为“subscribe”。

e) “To”=用户被联系人订阅（单向），这种状态在用户名册中体现为订阅状态为“To”。

f) “To + Pending In”=用户被联系人订阅，并且联系人向用户发出请求订阅，然而用户没有回复；这种状态在用户的名册中体现为订阅状态为“To”。

g) “Form”=联系人被用户订阅（单向）；这种状态在用户的名册中体现为订阅状态为“Form”。

h) “Form + Pending Out”=联系人被用户订阅，并且用户向联系人发送请求订阅然而联系人没有回复；这种状态在用户的名册中体现为订阅状态为“Form”和回复为“Subscribe”。

i) “Both”=用户和联系人都订阅了对方（双向的），这种状态在用户的名册中体现为订阅状态为“Both”。

A.2 服务器处理出站表示订阅节

A.2.1 概述

出站表示订阅节使得服务器可以管理联系人表示的订阅（通过“subscribe”和“unsubscribe”类型），并且管理联系人订阅用户表示的方式（通过“subscribed”和“unsubscribed”类型）。

下面的规则不仅适用于节的出站路由，还适用于用户名册的改变。表A.1～表A.4给出了出站各状态的处理情况。

这些规则是从用户的角度描述，而不是从联系人的角度。补充说明：“S.N.”代表不应该，“M.N.”代表不必）。

A.2.2 订阅

表A.1 出站“订阅”节的处理

当前状态	递送?	新状态
“None”	必须 ^{a)}	“None + Pending Out”
“None + Pending Out”	必须	状态不变
“None + Pending In”	必须 ^{a)}	“None + Pending Out + In”
“None + Pending Out/In”	必须	状态不变
“To”	必须	状态不变
“To + Pending In”	必须	状态不变
“From”	必须 ^{a)}	“From + Pending Out”
“From + Pending Out”	必须	状态不变
“Both”	必须	状态不变
^{a)} “Pending Out”状态的改变包括在用户的名册中设置“ask”的标志位为“subscribe”		

A.2.3 取消订阅

表A.2 出站“取消订阅”节的处理

当前状态	递送?	新状态
“None”	MUST	状态不变
“None + Pending Out”	MUST	“None”
“None + Pending In”	MUST	状态不变
“None + Pending Out/In”	MUST	“None + Pending In”
“To”	MUST	“None”
“To + Pending In”	MUST	“None + Pending In”
“From”	MUST	状态不变
“From + Pending Out”	MUST	“Form”
“Both”	MUST	“Form”

A.2.4 订阅

表A.3 出站“订阅”节的处理

当前状态	递送?	新状态
“None”	M.N	预批准a)
“None + Pending Out”	M.N.	预批准a)
“None + Pending In”	必须	“Form”
“None + Pending Out/In”	必须	“Form + Pending Out”
“To”	M.N	预批准a)
“To + Pending In”	必须	“Both”
“From”	M.N	状态不变

表3 (续)

当前状态	递送?	新状态
“From + Pending Out”	M.N	状态不变
“Both”	M.N.	状态不变
a) 有关预批准订阅的详细信息见7.4节所述		

A.2.5 取消订阅

表A.4 出站“取消订阅”节的处理

当前状态	递送?	新状态
“None”	S.N.	状态不变a)
“None + Pending Out”	S.N.	状态不变a)
“None + Pending In”	必须	“None”
“None + Pending Out/In”	必须	“None + Pending Out”
“To”	S.N.	状态不变a)
“To + Pending In”	必须	“To”
“From”	必须	“None”
“From + Pending Out”	必须	“None + Pending Out”
“Both”	必须	“To”
a) 此事件会导致预订阅的取消，见7.4节所述		

A.3 服务器处理入站表示订阅节

A.3.1 概述

入站表示订阅节从用户请求与订阅相关的行为（通过“subscribe”类型），把联系人所发生的与订阅相关的行为告知用户（通过“unsubscribe”类型），或者使得用户可以管理联系人获取用户表示信息的方式（通过“subscribed”和“unsubscribed”类型）。

下面的规则不仅适用于递送入站节，并且适用于用户名册的改变（服务器对入站表示订阅节的处理是从用户的角度描述的，而不是从联系人的角度。补充说明：“S.N.”代表不应该）。表A.5～表A.8给出了入站各状态的处理情况。

A.3.2 订阅

表A.5 入站“订阅”节的处理

当前状态	递送?	新状态
“None”	必须a)	“None + Pending In”
“None + Pending Out”	必须	“None + Pending Out + In”
“None + Pending In”	S.N.	状态不变
“None + Pending Out/In”	S.N.	状态不变
“To”	必须	“To + Pending In”
“To + Pending In”	S.N.	状态不变
“From”	S.N. b)	状态不变
“From + Pending Out”	S.N. b)	状态不变
“Both”	S.N. b)	状态不变
a) 如果用户之前发送过类型为“subscribe”的表示节，如附录A.2.3和7.4所述，则服务器可能以“subscribe”节自动回复并且更改状态为“Form”而不是“None + Pending In”。		
b) 服务器应该以“subscribed”节自动回复		

A.3.3 取消订阅

当用户的服务器收到联系人向其发送的类型为“unsubscribe”的表示节时，如果从用户的角度来看这个节导致了订阅状态的改变那么用户的服务器应改变状态，联系人应向服务器递送这个节，并且应该代表用户以类型为“unsubscribed”的表示节自动回复联系人。否则用户的服务器不能改变状态并且不应该递送这个节。这些规则在表A.6中总结。

表A.6 入站“取消订阅”节的处理

当前状态	递送?	新状态
“None”	S.N.	状态不变
“None + Pending Out”	S.N.	状态不变
“None + Pending In”	必须a)	“None”
“None + Pending Out/In”	必须a)	“None + Pending Out”
“To”	S.N.	状态不变
“To + Pending In”	必须a)	“TO”
“From”	必须a)	“None”
“From + Pending Out”	必须a)	“None + Pending Out”
“Both”	必须a)	“To”
a) 服务器应该以“unsubscribe”节自动回复		

A.3.4 订阅

当用户的服务器收到联系人向其发送的类型为“subscribe”的表示节，如果联系人的表示信息没有待入站请求，那么用户的服务器应改变订阅状态（因为状态不变）并且应该向用户递送这个节。如果存在联系人表示信息的待入站请求并且类型为“subscribe”的入站表示信息导致了订阅状态的改变，那么用户的服务器应改变订阅状态并且应向用户递送这个节。如果联系人的表示信息已经订阅了用户，那么类型为“subscribe”的入站表示节不会导致订阅状态的改变；因此用户的服务器不能改变订阅状态（因为状态不变）并且不应该向用户递送这个节。这些规则在表A.7中总结。

表A.7 入站“订阅”节的处理

当前状态	递送?	新状态
“None”	S.N.	状态不变
“None + Pending Out”	必须	“TO”
“None + Pending In”	S.N.	状态不变
“None + Pending Out/In”	必须	“To + Pending In”
“To”	S.N.	状态不变
“To + Pending In”	S.N.	状态不变
“From”	S.N.	状态不变
“From + Pending Out”	必须	“Both”
“Both”	S.N.	状态不变

A.3.5 取消订阅

当用户的服务器收到联系人向其发送的类型为“unsubscribed”的表示节时，如果联系人的表示信息没有待出站请求或者联系人的表示信息目前已经订阅了用户，那么用户的服务器应改变订阅状态并且应向用户递送这个节。否则，用户的服务器不能改变订阅状态（因为状态不变）并且不应该向用户递送这个节。这些规则在表A.8中总结。

表A.8 入站“取消订阅”节的处理

当前状态	递送?	新状态
“None”	S.N.	状态不变
“None + Pending Out”	必须	“None”
“None + Pending In”	S.N.	状态不变
“None + Pending Out/In”	必须	“None + Pending In”
“To”	必须	“None”
“To + Pending In”	必须	“None + Pending In”
“From”	S.N.	状态不变
“From + Pending Out”	必须	“Form”
“Both”	必须	“Form”

附 录 B
(资料性附录)
通信阻塞

8.6.6和IETF RFC 2779中的9.4.10节要求一个应存在即时通信和表示技术可以处理用户与选择的服务器之间的通信阻塞。这个要求的协议参见XEP-0016和XEP-0191。

附 录 C
(资料性附录)
vCards

RFC2779的7.1.3和8.1.4要求为其他用户接收带外的联系人信息 (如电话号码或电子邮件地址)是可能的。在XMPP社区中通常使用IETF RFC2426定义的以XML表示的vCard规范来提供这类信息，但是这超出本标准的范围。

附录 D

(资料性附录)

‘jabber:iq:roster’的 XML 规划

接下来的规划正式的定义了本标准中用到的 ‘jabber:iq:roster’ 名空间，与[XML-SCHEMA]相一致。因为XML流和节的确是选择性的，所以本规划不是规范性的，而是描述性的。规划定义的核心XMPP名字空间参见YD/T 2935 《扩展消息与表示协议（XMPP） 核心协议》。

```
<?xml version='1.0' encoding='UTF-8'?>
  <xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:iq:roster'
    xmlns='jabber:iq:roster'
    elementFormDefault='qualified'>

    <xs:element name='query'>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref='item'
            minOccurs='0'
            maxOccurs='unbounded'/>
        </xs:sequence>
        <xs:attribute name='ver'
          type='xs:string'
          use='optional'/>
      </xs:complexType>
    </xs:element>

    <xs:element name='item'>
      <xs:complexType>
        <xs:sequence>
          <xs:element ref='group'
            minOccurs='0'
            maxOccurs='unbounded'/>
        </xs:sequence>
        <xs:attribute name='approved'
          type='xs:boolean'
          use='optional'/>
        <xs:attribute name='ask'
```

```

        use='optional'>
    <xs:simpleType>
        <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='subscribe'/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name='jid'
                type='xs:string'
                use='required'/>
<xs:attribute name='name'
                type='xs:string'
                use='optional'/>
<xs:attribute name='subscription'
                use='optional'
                default='None'>
    <xs:simpleType>
        <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='both'/>
            <xs:enumeration value='from'/>
            <xs:enumeration value='None'/>
            <xs:enumeration value='remove'/>
            <xs:enumeration value='to'/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>

<xs:element name='group' type='xs:string'/>
</xs:schema>

```

中华人民共和国
通信行业标准
扩展消息与表示协议（XMPP）
即时消息与表示
YD/T 2936-2015

*

人民邮电出版社出版发行
北京市丰台区成寿寺路 11 号邮电出版大厦
邮政编码：100164
北京康利胶印厂印刷
版权所有 不得翻印

*

开本：880×1230 1/16 2016 年 2 月第 1 版
印张：5.25 2016 年 2 月北京第 1 次印刷
字数：142 千字

15115 • 865

定价：55 元

本书如有印装质量问题，请与本社联系 电话：(010)81055492