RFC3920

可扩展的消息和出席信息协议 (XMPP): 核心协议

关于本文的说明

本文为互联网社区定义了一个互联网标准跟踪协议，并且申请讨论协议和提出了改进的建议。请参照“互联网官方协议标准”的最新版本（STD 1）获得这个协议的标准化进程和状态。本文可以不受限制的分发。

版权声明

摘要

本文定义了可扩展消息和出席信息协议（XMPP）的核心功能，这个协议采用 XML 流实现在任意两个网络终端接近实时的交换结构化信息。XMPP 提供一个通用的可扩展的框架来交换 XML 数据，它主要用来建立即时消息和出席信息应用以实现 RFC 2779 的需求。

目录

---

1. 绪论

1.1. 概览

XMPP 是一个开放式的 XML 协议，设计用于准实时消息和出席信息以及请求-响应服务。其基本的语法和语义最初主要是由 Jabber 开放源代码社区于 1999 年开发的。2002 年，XMPP 工作组被授权接手开发和改编 Jabber 协议以适应 IETF 的消息和出席信息技术。作为 XMPP 工作组的成果，本文定义了 XMPP 1.0 的核心功能；在 RFC 2779 [IMP-REQS] 中指定的提供即时消息和出席信息功能的扩展，定义在 XMPP-IM 协议 [the Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence] 中。

1.2. 术语

本文中大写的关键字 "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", 和 "OPTIONAL" 的确切含义符合 BCP 14, RFC 2119 [TERMS].

2. 通用的架构

2.1. 概览

尽管 XMPP 没有结合任何特定的网络结构，通常认为它是 客户-服务器 架构的一种实现，在这里客户端用 XMPP 的方式访问服务器采用的是 TCP 连接，服务器之间的通信也是 TCP 连接。

以下是这一架构的抽象的示意图 (这里 "-" 表示使用 XMPP 通讯，"=" 表示可使用任何协议通讯)。

```
C1----S1---S2---C3
        |
C2----+---G1===FN1===FC1
```

符号代表的意思如下:

- C1, C2, C3 = XMPP 客户端
- S1, S2 = XMPP 服务器
- G1 = 一个 XMPP 和外部（非 XMPP）消息网络之间进行“翻译”的网关
- FN1 = 一个外部消息网络
- FC1 = 外部消息网络上的一个客户端

2.2. 服务器

服务器充当 XMPP 通信的一个智能抽象层，它主要负责：

- 管理发出的连接或其他实体的会话，在 XML 流（第四章）的表单中接收和发送给授权的客户端，服务器和其他实体。
- 用 XML 流通过实体转发特定地址的 XML 消息（第九章）

大部分 XMPP 兼容的服务器也负责存储客户端使用的数据（比如基于 XMPP 应用的联系人名单)；在这种情况下，XML 数据直接由服务器代替客户端处理而不需要转发到其他实体。

2.3. 客户端

大部分客户端通过 TCP 连接直接连到服务器，并通过 XMPP 获得由服务器和任何相关的服务所提供的全部功能。多个不同资源（比如不同的设备和地点）的客户端可以同时登陆并且并发的连接到一个服务器，每个不同资源的客户端通过 XMPP 地址的资源标识符来区分（比如<node@domain/ home> 和 <node@domain/work>），参见地址空间（第三章）。__建议__的客户端和服务器连接的端口是 5222 ，这个端口已经在 IANA（在第十五章第九节查阅端口号码）注册了。.

2.4. 网关

网关是一个特殊用途的服务器端的服务，主要功能是把 XMPP 翻译成外部(非 XMPP)消息系统，并把返回的消息翻译成 XMPP 。例如到 email（参见 [SMTP] ），IRC（参见 [IRC] ），SIMPLE（参见 [SIMPLE] ），SMS 的网关；还有和别的消息服务的网关，比如 AIM，ICQ，MSN Messenger，Yahoo! Instant Messenger。网关和服务器之间的通信，网关和外部消息系统的通信，不在本文描述范围之内。

2.5. 网络

因为每个服务器都是由一个网络地址来标识的并且服务器之间的通信是 客户-服务器 协议的直接扩展，实际上整个系统是由很多互通的服务器构成的。例如，<juliet@example.com> 可以和 <romeo@example.net> 交换消息,出席信息和其他信息。这种模式常见于那些需要使网络地址标准化的协议（比如 SMTP ）。任意两个服务器之间的通信是可选（OPTIONAL）的。如果被激活，那么这种通信应该（SHOULD）通过 XML 流绑定到 TCP 连接上进行。建议的（RECOMMENDED）服务器之间的连接端口为 5269 ，这个端口号已经在 IANA （在第十五章第九节查阅端口号码）注册了。

---

3. 地址空间

3.1. 概览

一个实体可以是任何一个被认为是一个网络端点的东西(例如网络上的一个 ID)，而且它是通过 XMPP 进行通信的。所有这些实体都有一个具有唯一性的地址，并符合 RFC 2396 [URI]规范要求的格式。由于历史原因，一个 XMPP 实体的地址被称为 Jabber Identifier 或 JID 。一个合法的 JID 包括一组排列好的元素，包括域名（domain identifier），节点名（node identifier），和资源名（resource identifier）。

JID 的语法定义，使用 [ABNF] 中的 Augmented Backus-Naur 格式。（IPv4 地址和 IPv6 地址规则在 附录 B 中的 [IPv6] 中定义；确定节点规则的合法字符顺序由 附录 A 的 [STRINGPREP] 的 Nodeprep 部分来定义；确定资源规则的合法字符顺序由 附录 B 的 [STRINGPREP] 的 Resourceprep 部分来定义；子域名规则参考 [IDNA] 中关于国际域名标签的描述。）。

```
jid = [ node "@" ] domain [ "/" resource ]
domain = fqdn / address-literal
fqdn = (sub-domain 1*("." sub-domain))
sub-domain = (internationalized domain label)
address-literal = IPv4address / IPv6address
```

所有 JID 都是基于上述的结构。类似 <user@host/resource> 这种结构，最常用来标识一个即时消息用户，这个用户所连接的服务器，以及这个用户用于连接的资源（比如特定类型的客户端软件）。不过,节点类型不是客户端也是有可能的，比如一个用来提供多用户聊天服务的特定的聊天室，地址可以是 <room@service>（这里 “room“ 是聊天室的名字而 ”service“ 是多用户聊天服务的主机名），而加入了这个聊天室的某个特定的用户的地址则是 <room@service/nick> （这里 ”nick“ 是用户在聊天室的昵称）。许多其他的 JID 类型都是可能的（例如 <domain/resource> 可能是一个服务器端的脚本或服务）。

一个 JID 的每个合法部分（节点名，域名，资源名）的长度不能（MUST NOT）超过 1023 字节。也就是整体长度（包括 ’@’ 和 ’/’ ）不能超过 3071 字节。

3.2. 域名

域名是一个主要的 ID 并且是 JID 中唯一必需（REQUIRED）的元素（一个纯粹的域名也是一个合法的 JID）。它通常代表网络的网关或者“主”服务器，其他实体通过连接它来实现 XML 转发和数据管理功能。然而，由一个域名标识引用的实体，并非总是一个服务器，它也可能是一个服务器的子域地址，提供额外的功能（比如多用户聊天服务，用户目录，或一个到外部消息系统的网关）。

每个服务器或者服务的域名，可以（MAY）是一个 IP 地址，但应该（SHOULD）是一个完全合法的域名(参见 [DNS]).一个域名 ID 必须(MUST)是 [IANA] 里定义的“国际化域名”，并且按 [STRINGPREP]中的 [NAMEPREP] profile 进行成

功的字符转换。在比较两个域名 ID 之前，服务器必须(MUST),客户端应该(SHOULD)首先按照 Nameprep profile(定义在[IANA]中) 来转换每个域名的字符。

3.3. 节点名

节点名是一个可选（OPTIONAL）的第二 ID，放在域名之前并用符号"@"分开.它通常表示一个向服务器或网关请求和使用网络服务的实体(比如一个客户端),当然它也能够表示其他的实体(比如在多用户聊天系统中的一个房间). 节点名所代表的实体，它的地址依赖于一个特定的域名;在 XMPP 的即时消息和出席信息应用系统中，这个地址是“纯 JID” <node@domain> 中的一部分。

一个节点名必须按 [STRINGPREP] 中的 Nodeprep profile 进行成功的字符转换。在比较两个节点 ID 之前，服务器必须（MUST）,客户端应该(SHOULD)首先按 Nodeprep profile 转换每个 ID 的字符。

3.4. 资源名

资源名是一个可选的第三 ID,它放在域名的后面并由符号"/"分开。资源名可以跟在 <node@domain>后面也可以跟在 <domain> 后面。它通常表示一个特定的会话，连接（比如设备或者所在位置），或者一个附属于某个节点 ID 实体相关实体的对象（比如多用户聊天室中的一个参加者）。对于服务器和和其他客户端来说，资源名是不透明的。当它提供必需的信息以完成资源绑定（参见第七章）的时候，通常是由客户端来实现的（尽管可以由客户端向服务器请求完成），然后显示为“已连接的资源”。一个实体可以(MAY)并发维护多个已连接的资源。每个已连接的资源由不同的资源 ID 来区分。

一个资源名必须（MUST）按 [STRINGPREP] 中的 Resourceprep profile 进行成功的格式化。在比较两个资源 ID 之前，服务器必须(MUST),客户端应该(SHOULD)首先按 Resourceprep profile 转换每个 ID 的字符。

3.5. 地址的确认

在 SASL （见第六章）握手之后（如果必要的话，也在资源绑定（见第七章）之后），正在接收流信息的实体必须（MUST）确认初始实体的 ID 。

对于服务器间的通信，在 SASL 握手时，如果没有指明授权的 ID，这个初始的实体应该（SHOULD）是经过认证实体（参见 简单认证和安全层协议 [SASL] 中的定义）授权的 ID（见第六章）。

对于客户端和服务器的通信，在 SASL 握手时，如果没有指明授权的 ID，“纯 JID” (<node@domain>)应该（SHOULD）是经过认证实体（参见 [SASL] 中的定

义）授权的 ID，“全 JID”（<node@domain/resource>)的资源 ID 部分应该（SHOULD）是由客户端和服务器在资源绑定的时候商定的（参见第七章）。

接收的实体必须（MUST）确保结果 JID（包括节点名，域名，资源名以及分隔符）与本章前面部分描述的规则和格式相一致；为了满足这些约束条件，接收实体可能（MAY）需要把初始实体的发送方 JID 替换成接收实体认可的规范 JID。

---

4. XML 流

4.1. 概览

两个基本概念，XML 流和 XML 节，使得在出席信息已知的实体之间，异步交换低负载的结构化信息成为可能。这两个术语定义如下：

XML 流的定义：一个 XML 流是一个容器，包含了两个实体之间通过网络交换的 XML 元素。一个 XML 流是由一个 XML 打开标签 <stream> (包含适当的属性和名字空间声明)开始的，流的结尾则是一个 XML 关闭 L 标签 </stream> 。在流的整个生命周期，初始化它的实体可以通过流发送大量的 XML 元素，用于流的握手(例如 TLS 握手(第五章) 或 SASL 握手(第六章))或 XML 节(在这里指符合缺省名字空间的元素，包括<message/>,<presence/>, 或 <iq/> 元素）。“初始的流”由初始化实体（通常是一个客户端或服务器）和接收实体（通常是一个服务器）握手，从接收实体来看，它就是那个初始实体的"会话".初始化流允许从初始化实体到接收实体的单向通信;为了使接收实体能够和初始实体交换信息,接收实体必须发起一个反向的握手(应答流).

XML 节的定义: 一个 XML 节是一个实体通过 XML 流向另一个实体发送的结构化信息中的一个离散的语义单位。一个 XML 直接存在于根元素<stream/>的下一级，并且如果这样就能够满足[XML]内容的 production 43,那么它被认为是均衡的.任何 XML 节都是从一个 XML 流的下一级的某个打开标签（如 <presence>）开始，到相应的关闭标签（如 </presence>）。一个 XML 节可以（MAY）包含子元素(相关的属性,元素, 和 XML 字符数据等) 以表达完整的信息.在这里定义的 XML 节仅限于<message/>, <presence/>, 和 <iq/> 元素，具体描述见 XML Stanzas（第九章）；为 TLS 握手（第五章）、SASL 握手（第六章）、服务器回拨（第八章）的需要而发送的 XML 元素，不被认为是一个 XML 节。

设想一个客户端和服务器会话的例子。为了连接一个服务器，一个客户端必须（MUST）发送一个打开标签<stream>给服务器，初始化一个 XML 流，也可选择（OPTIONAL）在这之前发送一段文本声明 XML 版本和支持的字符集（参见文本声明的内容(第十一章第四节)；也可看字符编码(第十一章第五节)）。视本地化策略和提供的服务而定，服务器应该（SHOULD）回复一个 XML 流给客户端，同样的，也可选择在这之前发 送一段文本声明。一旦客户端完成了 SASL 握手（第六章），客户端可以（MAY）通过流发送不限量的 XML 节给网络中的任何接收者。当客户端想关闭这个流，它只需要简单的发送一个关闭标签</stream>给服务器（或者

作为另一个选择，可能由服务器关闭这个流）。然后，客户端和服务器都应该（SHOULD）彻底地终止这个连接（通常是一个 TCP 连接）。

那些习惯认为 XML 是一个以文本为中心风格的人可能希望看看一个与服务器连接的客户端会话，包含两个 打开-关闭 XML 文档：一个是从客户端到服务器，一个是从服务器到客户端。下图中，根元素<stream/> 可以被认为是每个“文档”的文档实体,这两个“文档”通过累积那些在 XML 上传输的 XML 节来搭建的。无论如何，下图只是方便理解；实际上 XMPP 并不处理文档而是处理 XML 流和 XML 节。

基本上，一个 XML 流相当于一个会话期间所有 XML 节的一个信封。我们可以简单的把它描述成下图：

```
|----------------------------
| <stream>
|----------------------------
| <presence>
| <show/>
| </presence>
|----------------------------
| <message to='foo'>
| <body/>
| </message>
|----------------------------
| <iq to='bar'>
| <query/>
| </iq>
|----------------------------
| …
|----------------------------
| </stream>
|----------------------------
```

4.2. 绑定到 TCP

虽然有很多非必需的连接使用 XML 流来绑定[TCP]连接（两个实体可以通过别的机制来互联，比如通过[HTPP]连接轮询），本规范只定义了 XMPP 到 TCP 的绑定。在客户和服务器通信的过程中，服务器必须（MUST）允许客户端共享一个 TCP 连接来传输 XML 节，包括从客户端传到服务器和从服务器传到客户端。在服务器之间的通信过程中，服务器必须（MUST）用一个 TCP 连接 向对方发送 XML 节，另一个 TCP 连接（由对方初始化）接收对方的 XML 节，一共两个 TCP 连接。

4.3. 流的安全

在 XMPP 1.0 中，当 XML 流开始握手时，TLS 应该（SHOULD）按 第五章：TLS 的使用 中的规定来使用，SASL 必须（MUST）按 第六章：SASL 的使用 中的规定来使用。尽管可能（MAY）存在某种共有的机制能够保证双向安全，但是“初始化流”（比如从初始化实体发给接收实体的流）和“应答流”（比如从接收实体发给初始化实体的流）还是必须（MUST）安全的分开。在流被验证之间，实体不应该（SHOULD NOT） 尝试通过流发送 XML 节（第九章）；就算它这样做了，对方的实体也不能（MUST NOT）接受这些 XML 节，并且应该（SHOULD）返回一个 <not-authorized/> 的流错误信息并且终止当前 TCP 连接上双方的 XML 流;注意，这仅仅是针对 XML 节（包含在缺省命名空间中的 <message/>, <presence/>, 和 <iq/> 元素），而不是指那些用于 TLS 握手（第五章）、SASL 握手（第六章）握手的流。

4.4. 流属性

流元素的属性如下:

- to -- 'to'属性应该（SHOULD）仅用于从初始化实体到接收实体的 XML 流的头，并且必须（MUST）设成为接收实体提供服务的主机名。注意，不应该（SHOULD NOT）有 'to'属性出现在接收实体应答初始实体的 XML 流的头中；无论如何，如果'to'属性出现在应答流中，初始化实体应该（SHOULD）忽略它。
- from --'from'属性应该(SHOULD)仅用于接收实体应答初始化实体的 XML 流的头，并且必须（MUST）设成为接收实体（正在给初始实体授权）提供服务的主机名。注意，不应该（SHOULD NOT）有 'from'属性出现在初始实体发送给接收实体的 XML 流的头中；无论如何，如果'from'属性出现在初始化流中，接收实体应该（SHOULD）忽略它。
- id -- 'id'属性应该（SHOULD）仅用于接收实体发送给初始化实体 XML 流的头。这个属性是一个由接收实体创建的具有唯一性的 ID,一个初始实体和接收实体之间的会话 ID，并且它在接收方的应用程序中（通常是一个服务器）必须（MUST）是唯一的。注意，这个流 ID 必须是足够安全的，所以它必须是不可预知的和不可重复的(参见 RANDOM 了解如何获得随机性以保证安全性）。不应该（SHOULD NOT）有 'id'属性出现在初始实体发送给接收实体的 XML 流的头中；无论如何，如果'id'属性出现在初始化流中，接收实体应该（SHOULD）忽略它。
- xml:lang -- 'xml:lang'属性（定义在[XML]中的第二章第十二节）应该（SHOULD）包含在初始化实体发给接收实体的 XML 流的头中，以指定在流中传输的可读 XML 字符所使用的缺省语言。如果这个属性出现了，接收实体应该（SHOULD）记住它的值，作为初始化流和应答流的缺省属性；如果这个属性没有出现，接收实体应该（SHOULD）用一个可配置的缺省值用于双方的流，这个属性值必须（MUST）在应答流的头中传达。对于所有初始化流中传输的节，如果初始实体没有提供'xml:lang'属性，接收实体应该（SHOULD）应用缺省值；如果初始实体提供了'xml:lang'属性，接收实体不能（MUST NOT）修改或删除它（参见第九章第一节第五小节 xml:lang）。'xml:lang'属性的值必须（MUST）是一个 NMTOKEN (定义在

[XML]的第二章第三节）并且必须（MUST）遵守 RFC 3066 [LANGTAGS] 规定的格式。

- version -- version 属性（最少需要"1.0"）为本规范中和流相关的协议提供了支持。关于这个属性的生成和处理的详细规则将在下文中定义。

我们现在可以总结如下：

	初始化方发给接收方	接收方发给初始化方
to	接收方的主机名	忽略
from	忽略	接收方的主机名
id	忽略	会话键值
xml:lang	缺省语言	缺省语言
version	支持 XMPP 1.0	支持 XMPP 1.0

4.4.1. 版本支持

在这里 XMPP 的版本是"1.0"；准确地说，这里囊括了和流相关的所有协议(TLS 的使用 (第五章),SASL 的使用(第六章)， 和流错误(第四章第七节))，以及三个定义好的 XML 节类型(<message/>,<presence/>,和 <iq/>)。XMPP 版本号的编号顺序是“<主版本号>.<副版本号>”。主版本和副版本号必须（MUST）是独立的整数并且每个号码可以（MAY）单独以阿拉伯数字增长。这样，"XMPP 2.4"的版本将比"XMPP 2.13"更低。号码前面 的“0”(比如 XMPP 6.01)必须(MUST)被接收方忽略并且不能(MUST NOT)被发送出去.

如果流和节的格式或者必需的处理方式有了显著的改变，以至于老版本的实体如果只是简单的忽略它不理解的节和属性并且继续像老版本一样的处理方式，会使得老版本的实体不能够和新版本的实体交互，只有在这时候主版本号才应该（SHOULD）增加。副版本号显示新的性能，它必须（MUST）被副版本号更低的实体忽略，但被高（副）版本号的实体用于了解信息。例如，一个副版本号显示处理某种新定义的"type"属性的值（用于 message,presence 或 IQ 节）的能力；副版本号高的实体将会了解到与之通信的对方不能够理解这个"type"属性的值，所以将不会发送它。

以下规则是用于'版本'属性在实现流的头信息时如何生成和处理：

1. 初始化实体必须（MUST）在初始化流的头信息中把'版本'的值设置成它所支持的最高版本。(比如，如果最高版本支持就是本规范，那么它必须(MUST)设置成"1.0")。
2. 接收实体必须（MUST）在应答流的头信息中把'版本'的值设置成初始化实体所提供的版本或它所支持的最高版本，取其中版本号较低的那一个。接收实体必须(MUST)把主版本号和副版本号作为数字来比较，而不是对"主版本号.副版本号"这个字符串进行比较.
3. 如果在应答流的头信息的版本号中至少有一个主版本号低于初始化流的头信息的版本号，并且如前所述，新版本的实体不能够和旧版本实体交互，初始化实体应该(SHUOULD)生成一个<unsupported-version/>的流错误信息并终止 XML 流和它的 TCP 连接。

4. 如果一个实体收到一个头信息中没有'version'属性的流，这个实体必须（MUST)把对方实体的'version'当成'0.0'并且在它发送的应答流的头中也不应该(SHOULD NOT)包含'version'属性.

4.5. 名字空间声明

流的元素必须(MUST)同时满足一个流名字空间声明和一个缺省名字空间声明("名字空间声明"定义在 XML 名字空间定义 [XML-NAMES]中).关于流名字空间和缺省名字空间的详细信息,参考 名字空间的名字和前缀(第十一章第二节).

4.6. 流的特性

如果初始化的实体在初始化流的头信息中设置'version'属性的信息为"1.0",接收实体必须(MUST)向初始化实体发送一个 <features/> 子元素以声明任何可供协商的流一级的特性(或者其他需要声明的能力).目前,这仅用于声明本文中定义的 TLS 的使用(第五章),SASL 的使用 （第六章） 和资源绑定(第七章),以及 XMPP-IM 中定义的会话的建立;无论如何,流特性这一功能将来可以用于声明任何可协商的特性.如果一个实体不理解或支持安全特性,它应该(SHOULD)忽略它.如果要在一个非安全相关的特性(比如资源绑定)被提议之前,完成一个或多个安全特性(比如 TLS 和 SASL)的协商,这个非安全相关的特性不应该(SHOULD NOT)在相应的安全特性协商完毕之前被声明.

4.7. 流错误

流的根元素可以（MAY）包含一个 <error/> 子元素，由流的名字空间前缀作为它的前缀。这个"错误"子元素必须（MUST）由感知到发生了流级别错误的实体发送（通常是一个服务器而不是一个客户端）。

4.7.1. 规则

以下规则适用于流级别的错误:

- 它假定所有流级别的错误都是不可恢复的；所以，如果一个错误发生在流级别，发现这个错误的实体必须（MUST）发送一个流错误信息给另一个实体，发送一个关闭标签 </stream>，并终止这个流所在的 TCP 连接。
- 如果这个错误发生在流刚开始设置的时候，接收实体必须（MUST）仍然发送一个开放标签 <stream> ，并在流元素中包含一个<error/>的子元素，然后发送一个关闭标签 </stream>，最后终止相应的 TCP 连接。在这种情况下，如果初始化实体在 'to' 属性中提供了一个未知的主机名，服务器应该（SHOULD）在终止之前,先在流的头信息的 'from' 属性中提供一个服务器认证的主机名.

4.7.2. 语法

流错误的语法如下:

```
<stream:error>
     <defined-condition xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
     <text xmlns='urn:ietf:params:xml:ns:xmpp-streams'
           xml:lang='langcode'>
       OPTIONAL descriptive text
     </text>
     [OPTIONAL application-specific condition element]
   </stream:error>
```

<error/>元素:

- 必须(MUST)包含一个子元素以描述一个下文定义的节错误条件;这个子元素必须(MUST)符合'urn:ietf:params:xml:ns:xmpp-streams'名字空间.
- 可以(MAY)包含一个 <text/> 子元素,用 XML 字符数据描述错误的细节;这个元素必须(MUST)符合'urn:ietf:params:xml:ns:xmpp-streams'名字空间并且应该(SHOULD)拥有一个'xml:lang'属性表明 XML 字符数据的自然语言。
- 可以（MAY）包含一个子元素用于描述一个明确的应用程序错误条件；这个元素必须（MUST）符合一个应用程序定义的名字空间，并且它的结构是由那个名字空间定义的。

<text/> 元素是可选的（OPTIONAL）。如果有这个元素，它应该（SHOULD）仅用于提供描述或调试信息以补充一个已定义的条件或应用程序定义的条件。它不应该(SHOULD NOT)被一个应用程序当成一个可编程的信息。它不应该(SHOULD NOT)被用于向用户表达错误信息，但是可以（MAY）作为和条件元素相关的错误信息之外的附加说明。

4.7.3. 已定义的条件

以下流级别的错误条件是已定义的:

- <bad-format/> -- 实体已经发送 XML 但是不能被处理;这个错误可以（可以)被更多特定的 XML 相关的错误替换,比如 <bad-namespace-prefix/>,

<invalid-xml/>, <restricted-xml/>, <unsupported-encoding/>, 以及 <xml-not-well-formed/>,尽管更多特定的错误是首选的。

- <bad-namespace-prefix/> -- 实体发送的名字空间前缀不被支持，或者在一个需要某种前缀的元素中没有发送一个名字空间前缀(参见 XML Namespace Names and Prefixes (第十一章第二节)).
- <conflict/> -- 服务器正在关闭为这个实体激活的流，因为一个和已经存在的流有冲突的新的流已经被初始化。
- <connection-timeout/> -- 实体已经很长时间没有通过这个流发生任何通信流量(可由一个本地服务策略来配置).
- <host-gone/> -- 初始化实体在流的头信息中提供的'to'属性的值所指定的主机已经不再由这台服务器提供
- <host-unknown/> -- 由初始化实体在流的头信息中提供的 'to' 属性的值和由服务器提供的主机名不一致.
- <improper-addressing/> -- 一个在两台服务器之间传送的节缺少 'to' 或 'from' 属性(或者这个属性没有值).
- <internal-server-error/> -- 服务器配置错误或者其他未定义的内部错误,使得服务器无法提供流服务.
- <invalid-from/> -- 在'from'属性中提供的 JID 或 主机名地址，和认证的 JID 不匹配 或服务器之间无法通过 SASL（或回拨）协商出合法的域名，或客户端和服务器之间无法通过它进行认证和资源绑定。
- <invalid-id/> -- 流 ID 或回拨 ID 是非法的或和以前提供的 ID 不一致.
- <invalid-namespace/> -- 流名字空间和 "http://etherx.jabber.org/streams" 不相同或回拨名字空间和 "jabber:server:dialback" 不相同.(参考 XML Namespace Names and Prefixes (第十一章第二节)).
- <invalid-xml/> -- 实体通过流发送了一个非法的 XML 给执行验证的服务器 (参考 Validation (第十一章第三节)).
- <not-authorized/> -- 实体试图在流被验证之前发送数据或不被许可执行一个和流协商有关的动作，接收实体在发送错误信息之前不允许（MUST NOT）处理厌恶的节。
- <policy-violation/> -- 实体违反了某些本地服务策略；服务器可以（MAY）选择在 <text/> 元素或应用程序定义的错误条件（元素）中详细说明策略。
- <remote-connection-failed/> -- 服务器无法正确连接到用于验证或授权的远程实体。
- <resource-constraint/> -- 服务器缺乏必要的系统资源为流服务。
- <restricted-xml/> -- 实体试图发送受限的 XML 特性，比如一个注释，处理指示，DTD，实体参考，或保留的字符(参考 Restrictions (第十一章第一节)).
- <see-other-host/> -- 服务器将不提供服务给初始化实体但是把它重定向到另一台主机；服务器应该（SHOULD）在<see-other-host/>元素的 XML 字符数据中指明替代服务器名或 IP 地址（它必须（必须）是合法的域名标识）。

- <system-shutdown/> -- 服务器正在关机并且所有激活的流正在被关闭。
- <undefined-condition/> -- 错误条件不在本文已定义的错误条件列表之中；这个错误条件应该（SHOULD）仅用于"应用程序定义条件"元素.
- <unsupported-encoding/> -- 初始化实体以一个服务器不不支持的编码方式编码了一个流(参照 Character Encoding (第十一章第五节)).
- <unsupported-stanza-type/> -- 初始化实体发送了一个流的一级子元素但是服务器不支持.
- <unsupported-version/> -- 由初始化实体在流的头信息中指定的'version'属性的值所指定的版本不被服务器支持;服务器可以(MAY)在<text/>元素中指定一个它支持的版本号.
- <xml-not-well-formed/> -- 初始化实体发送了一个不规范的 XML（参考[XML]）.

4.7.4. 应用程序定义条件

大家知道,应用程序可以(MAY)在 error 元素中包含一个适当名字空间的子元素来提供一个应用程序定义流错误信息."应用程序定义"元素应该(SHOULD)补充或甚至限定一个已定义的元素.所以 <error/> 元素将包含两个或三个子元素:

```
<stream:error>

     <xml-not-well-formed

         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>

     <text xml:lang='en' xmlns='urn:ietf:params:xml:ns:xmpp-streams'>

       Some special application diagnostic information!

     </text>

     <escape-your-data xmlns='application-ns'/>

   </stream:error>

   </stream:stream>
```

4.8. 简化的流示例

这里包含两个简化的例子，描述了基于流的客户端在服务器上的“会话”(这里"C"表示从客户端发给服务器,"S"表示从服务器发给客户端);这些例子只是用于举例说明原理.

一个基本的 "会话":

```
C: <?xml version='1.0'?>
      <stream:stream
          to='example.com'
          xmlns='jabber:client'
          xmlns:stream='http://etherx.jabber.org/streams'
          version='1.0'>
   S: <?xml version='1.0'?>
      <stream:stream
          from='example.com'
          id='someid'
          xmlns='jabber:client'
          xmlns:stream='http://etherx.jabber.org/streams'
          version='1.0'>
   ...  encryption, authentication, and resource binding ...
   C:   <message from='juliet@example.com'
                 to='romeo@example.net'
                 xml:lang='en'>
   C:     <body>Art thou not Romeo, and a Montague?</body>
   C:   </message>
   S:   <message from='romeo@example.net'
                 to='juliet@example.com'
                 xml:lang='en'>
   S:     <body>Neither, fair saint, if either thee dislike.</body>
   S:   </message>
```

C: </stream:stream>

S: </stream:stream>

一个不成功的 "会话" :

```
C: <?xml version='1.0'?>
      <stream:stream
          to='example.com'
          xmlns='jabber:client'
          xmlns:stream='http://etherx.jabber.org/streams'
          version='1.0'>
   S: <?xml version='1.0'?>
      <stream:stream
          from='example.com'
          id='someid'
          xmlns='jabber:client'
          xmlns:stream='http://etherx.jabber.org/streams'
          version='1.0'>
   ⋯  encryption, authentication, and resource binding ...
   C: <message xml:lang='en'>
        <body>Bad XML, no closing body tag!
      </message>
   S: <stream:error>
       <xml-not-well-formed
           xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
      </stream:error>
```

S: </stream:stream>

---

5. TLS 的使用

5.1. 概览

XMPP 包含的一个保证流安全的方法来防止篡改和偷听.这个传输层安全协议[TLS]的频道加密方法, 模拟了类似的其他"STARTTLS"(见 RFC 2595 [USINGTLS])的扩展,如 IMAP [IMAP], POP3 [POP3], and ACAP [ACAP]."STARTTLS"的扩展名字空间是'urn:ietf:params:xml:ns:xmpp-tls'.

一个给定域的管理员可以(MAY)要求客户端和服务器通信以及服务器之间通信时使用 TLS，或者两者都要求。客户端应该（SHOULD）在尝试完成 SASL （第六章）握手之前使用 TLS，服务器应该（SHOULD）在两个域之间使用 TLS 以保证服务器间通信的安全。

以下是使用规则:

1. 一个遵守本协议的初始化实体必须（MUST）在初始化流的头信息中包含一个'version'属性并把值设为“1.0”。
2. 如果 TLS 握手发生在两个服务器之间,除非服务器声称的 DNS 主机名已经被解析(见第十四章第四节 Server-to-Server Communications)，通信不能（MUST NOT）继续进行。
3. 当一个遵守本协议的接收实体接收了一个初始化流(它的头信息中包含一个'version'属性并且值设为“1.0”)，在发送应答流的的头信息（其中包含版本标记）之后，它必须发送（MUST）<starttls/>元素（名字空间为 'urn:ietf:params:xml:ns:xmpp-tls'）以及其他它支持的流特性 。
4. 如果初始化实体选择使用 TLS,TLS 握手必须在 SASL 握手之前完成；这个顺序用于帮助保护 SASL 握手时发送的认证信息的安全，同时可以在必要的时候在 TLS 握手之前为 SASL 外部机制提供证书。
5. TLS 握手期间,一个实体不能（MUST NOT）在流的根元素中发送任何空格符号作为元素的分隔符(在下面的 TLS 示例中的任何空格符都仅仅是为了便于阅读）；这个禁令用来帮助确保安全层字节精度。
6. 接收实体必须（MUST）在发送<proceed/> 元素的关闭符号">" 之后立刻开始 TLS 协商。初始化实体必须（MUST）在从接收实体接收到<proceed/>元素的关闭符号">" 之后立刻开始 TLS 协商。
7. 初始化实体必须（MUST）验证接收实体出示的证书；关于证书验证流程参见 Certificate Validation （ 第十四章第二节)。
8. ？？ 证书必须（MUST）检查初始化实体(比如一个用户)提供的主机名；而不是通过 DNS 系统解析出来的主机名；例如，如果用户指定一个主机名"example.com"而一个 DNS SRV [SRV]查询返回"im.example.com"，证书必须（MUST）检查"example.com".如果任何种类的 XMPP 实体（例如客户端或服务器）的 JID 出现在一个证书里，它必须（MUST）表现为一个别名

实体里面的UTF8字符串,存在于subjectAltName之中。如何使用 [ASN.1] 对象标识符 "id-on-xmppAddr" 定义在本文的第五章第一节第一小节。

9. 如果 TLS 握手成功了，接收实体必须（MUST） 丢弃 TLS 生效之前从初始化实体得到的任何不可靠的信息.
10. 如果 TLS 握手成功了，初始化实体必须（MUST） 丢弃 TLS 生效之前从接收实体得到的任何不可靠的信息.
11. 如果 TLS 握手成功了，接收实体不能(MUST NOT)在流重新开始的时候通过提供其他的流特性来向初始化实体提供 STARTTLS 扩展.
12. 如果 TLS 握手成功了，初始化实体必须(MUST)继续进行 SASL 握手。
13. 如果 TLS 握手失败了，接收实体必须（MUST）终止 XML 流和相应的 TCP 连接。
14. 关于必须（MUST）支持的机制，参照 Mandatory-to-Implement Technologies（第十四章第七节）。

5.1.1. 用于 XMPP 地址的 ASN.1 对象标识符

上文提到的[ASN.1] 对象标识符 "id-on-xmppAddr"定义如下:

```
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6)
internet(1) security(5) mechanisms(5) pkix(7) }

id-on OBJECT IDENTIFIER ::= { id-pkix 8 } -- other name forms

id-on-xmppAddr OBJECT IDENTIFIER ::= { id-on 5 }

XmppAddr ::= UTF8String
```

对象标识符也可以(MAY)使用点分隔的格式,如 "1.3.6.1.5.5.7.8.5".

5.2. 叙述

当一个初始化实体用 TLS 保护一个和接收实体之间的流，其步骤如下:

1. 初始化实体打开一个 TCP 连接，发送一个打开的 XML 流头信息（其'version'属性设置为"1.0"）给接收实体以初始化一个流。
2. 接收实体打开一个 TCP 连接，发送一个 XML 流头信息（其'version'属性设置为"1.0"）给初始化实体作为应答。
3. 接收实体向初始化实体提议 STARTTLS 范围（包括其他支持的流特性），如果 TLS 对于和接收实体交互是必需的，它应该(SHOULD)在<starttls/>元素中包含子元素<required/>.
4. 初始化实体发出 STARTTLS 命令(例如，一个符合'urn:ietf:params:xml:ns:xmpp-tls'名字空间的 <starttls/> 元素)以通知接收实体它希望开始一个 TLS 握手来保护流。

5. 接收实体必须（MUST）以'urn:ietf:params:xml:ns:xmpp-tls'名字空间中的<proceed/>元素或<failure/>元素应答。如果失败，接收实体必须（MUST）终止 XML 流和相应的 TCP 连接。如果继续进行，接收实体必须（MUST）尝试通过 TCP 连接完成 TLS 握手并且在 TLS 握手完成之前不能（MUST NOT）发送任何其他 XML 数据。
6. 初始化实体和接收实体尝试完成 TLS 握手。（要符合[TLS]规范）
7. 如果 TLS 握手不成功，接收实体必须（MUST）终止 TCP 连接. 如果 TLS 握手成功，初始化实体必须（MUST）发送给接收实体一个打开的 XML 流头信息来初始化一个新的流（先发送一个关闭标签</stream>是不必要的，因为接收实体和初始化实体必须(MUST)确保原来的流在 TLS 握手成功之后被关闭） 。
8. 在从初始化实体收到新的流头信息之后，接收实体必须（MUST）发送一个新的 XML 流头信息给初始化实体作为应答，其中应包含可用的特性但不包含 STATRTTLS 特性。

5.3. 客户端-服务器 示例

以下例子展示一个客户端使用 STARTTLS 保护数据流（注意：以下步骤举例说明协议中的失败案例；在这个例子中它们并不详尽并且也不是必须被数据传输触发的）.

步骤 1：客户端初始化流给服务器：

```
<stream:stream

        xmlns='jabber:client'

        xmlns:stream='http://etherx.jabber.org/streams'

        to='example.com'

        version='1.0'>
```

步骤 2：服务器发送一个流标签给客户端作为应答：

```
<stream:stream

        xmlns='jabber:client'

        xmlns:stream='http://etherx.jabber.org/streams'

        id='c2s_123'

        from='example.com'

        version='1.0'>
```

步骤 3: 服务器发送 STARTTLS 范围给客户端（包括验证机制和任何其他流特性）:

```
<stream:features>

     <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>

       <required/>

     </starttls>

     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

       <mechanism>DIGEST-MD5</mechanism>

       <mechanism>PLAIN</mechanism>

     </mechanisms>

   </stream:features>
```

步骤 4: 客户端发送 STARTTLS 命令给服务器:

```
<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

步骤 5: 服务器通知客户端可以继续进行:

```
<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

步骤 5 (或者): 服务器通知客户端 TLS 握手失败并关闭流和 TCP 连接:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

   </stream:stream>
```

步骤 6: 客户端和服务器尝试通过已有的 TCP 连接完成 TLS 握手.

步骤 7: 如果 TLS 握手成功, 客户端初始化一个新的流给服务器:

```
<stream:stream

       xmlns='jabber:client'

       xmlns:stream='http://etherx.jabber.org/streams'

       to='example.com'

       version='1.0'>
```

步骤 7 (或者): 如果 TLS 握手不成功, 服务器关闭 TCP 连接.

步骤 8: 服务器发送一个流头信息应答客户端，其中包括任何可用的流特性:

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        from='example.com'
        id='c2s_234'
        version='1.0'>
   <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>DIGEST-MD5</mechanism>
       <mechanism>PLAIN</mechanism>
       <mechanism>EXTERNAL</mechanism>
     </mechanisms>
   </stream:features>
```

步骤 9: 客户端继续 SASL 握手 (第六章).

5.4. 服务器-服务器示例

以下例子展示两个服务器之间使用 STARTTLS 保护数据流 (注意: 以下步骤举例说明协议中的失败案例;在这个例子中它们并不详尽并且也不是必须被数据传输触发的).

步骤 1: Server1 初始化流给 Server2:

```
<stream:stream
        xmlns='jabber:server'
        xmlns:stream='http://etherx.jabber.org/streams'
        to='example.com'
```

version='1.0'>

步骤 2: Server2 发送一个流标签给 Server1 作为应答:

```
<stream:stream

       xmlns='jabber:server'

       xmlns:stream='http://etherx.jabber.org/streams'

       from='example.com'

       id='s2s_123'

       version='1.0'>
```

步骤 3: Server2 发送 STARTTLS 范围给 Server1 ，包括验证机制和任何其他流特性:

```
<stream:features>

     <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>

       <required/>

     </starttls>

     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

       <mechanism>DIGEST-MD5</mechanism>

       <mechanism>KERBEROS_V4</mechanism>

     </mechanisms>

   </stream:features>
```

步骤 4: Server1 发送 STARTTLS 命令给 Server2:

```
<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

步骤 5: Server2 通知 Server1 允许继续进行:

```
<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

步骤 5 (或者): Server2 通知 Server1 TLS 握手失败并关闭流:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>

   </stream:stream>
```

步骤 6: Server1 和 Server2 尝试通过 TCP 完成 TLS 握手.

步骤 7: 如果 TLS 握手成功, Server1 初始化一个新的流给 Server2:

```
<stream:stream
        xmlns='jabber:server'
        xmlns:stream='http://etherx.jabber.org/streams'
        to='example.com'
        version='1.0'>
```

步骤 7 (或者): 如果 TLS 握手不成功, Server2 关闭 TCP 连接.

步骤 8: Server2 发送一个包含任何可用流特性的流头信息给 Server1 :

```
<stream:stream
        xmlns='jabber:server'
        xmlns:stream='http://etherx.jabber.org/streams'
        from='example.com'
        id='s2s_234'
        version='1.0'>
   <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>DIGEST-MD5</mechanism>
       <mechanism>KERBEROS_V4</mechanism>
       <mechanism>EXTERNAL</mechanism>
     </mechanisms>
   </stream:features>
```

步骤 9: Server1 继续进行 SASL 握手(第六章).

6. SASL 的使用

6.1. 概览

XMPP 有一个验证流的方法，即 XMPP 特定的 SASL（简单验证和安全层）[SASL]。SASL 提供了一个通用的方法为基于连接的协议增加验证支持，而 XMPP 使用了一个普通的 XML 名字空间来满足 SASL 的需要。

以下规则应用于:

1. 如果 SASL 协商发生在两台服务器之间，除非服务器宣称的 DNS 主机名得到解析，不能（MUST NOT）进行通信。(参见 服务器间的通信(第十四章第四节)).
2. 如果初始化实体有能力使用 SASL 协商，它必须（MUST）在初始化流的头信息中包含一个值为"1.0"的属性'version'。
3. 如果接收实体有能力使用 SASL 协商，它必须（MUST）在应答从初始化实体收到的打开流标签时(如果打开的流标签包含一个值为"1.0"的'version'属性),通过'urn:ietf:params:xml:ns:xmpp-sasl'名字空间中的<mechanisms/>元素声明一个或多个验证机制.
4. 当 SASL 协商时，一个实体不能（MUST NOT）在流的根元素中发送任何空格符号(匹配 production [3] content of [XML])作为元素之间的分隔符(在以下的 SASL 例子中任何空格符号的出现仅仅是为了增加可读性)；这条禁令帮助确保安全层字节的精确度。
5. 当 SASL 握手时，在 XML 元素中使用的任何 XML 字符数据必须被编码成 base64，编码遵循 RFC 3548 第三章的规定。
6. ？？如果一个 简单名字"simple username" 规范被选定的 SASL 机制所支持，(比如，这被 DIGEST-MD5 和 CRAM-MD5 机制支持但不被 EXTERNAL 和 GSSAPI 机制支持)，验证的时候初始化实体应该（SHOULD）在服务器间通信时提供 简单名字 自身的发送域（IP 地址或包含在一个域标识符中的域名全称），在客户端与服务器之间通信时提供注册用户名（包含在一个 XMPP 节点标识符中的用户或节点名）。
7. 如果初始化实体希望以另一个实体的身份出现并且 SASL 机制支持授权 ID 的传输，初始化实体在 SASL 握手时必须（MUST）提供一个授权 ID。如果初始化实体不希望以另一个实体的身份出现，初始化实体在 SASL 握手时不能（MUST NOT）提供一个授权 ID。在 [SASL] 的定义中,除非授权 ID 不同于从验证 ID（详见[SASL]）中得到的缺省的授权 ID，初始化实体不能（MUST NOT）提供授权 ID。如果提供了，这个授权 ID 的值必须（MUST）是<domain>的格式(对于服务器来说)或<node@domain>的格式(对于客户端来说).
8. 在成功进行包括安全层的 SASL 握手之后，接收实体必须（MUST）丢弃任何从初始化实体得到的而不是从 SASL 协商本身获得的信息。
9. 在成功进行包括安全层的 SASL 握手之后，初始化实体必须（MUST）丢弃任何从接收实体得到的而不是从 SASL 协商本身获得的信息。
10. 参看 强制执行的技术（第十四章第七届），了解关于必须（MUST）支持的机制.

6.2. 叙述

一个初始化实体使用 SASL 和接收实体做验证的步骤如下:

1. 初始化实体请求 SASL 验证,它发送一个打开的 XML 流头信息给接收实体,其'version'属性的值为"1.0".
2. 在发送一个 XML 流头回复之后，接收实体声明一个可用的 SASL 验证机制清单；每个机制作为一个<mechanism/>元素，作为子元素包含在<mechanisms/>容器元素（其名字空间为'urn:ietf:params:xml:ns:xmpp-sasl'）中，而<mechanisms/>则包含在流名字空间中的<features/>元素中。如果在使用任何验证机制之前需要使用 TLS（见第五章），接收实体不能（MUST NOT）在 TLS 握手之前提供可用的 SASL 验证机制清单。如果初始化实体在优先的 TLS 协商过程中呈现了一个合法的证书，接收实体应该（SHOULD）在 SASL 握手中提出一个 SASL 外部机制给初始化实体，尽管这个外部机制可以（MAY）在其它环境下提供。
3. 初始化实体发送一个符合'urn:ietf:params:xml:ns:xmpp-sasl'名字空间的<auth/>元素（其中包含了适当的'mechanism'属性值）给接收实体，以选择一个机制。如果这个机制支持或需要，这个元素可以（MAY）包含 XML 字符数据（在 SASL 术语中，即“初始化应答”）；如果初始化实体需要发送一个零字节的初始化应答，它必须（MUST）传输一个单独的等号作为应答，这表示应答有效但不包含数据。
4. 如果必要，接收实体向初始化实体发送一个符合'urn:ietf:params:xml:ns:xmpp-sasl'名字空间的<challenge/>元素来发出挑战；这个元素可以（MAY）包含 XML 字符数据（必须按照初始化实体选择的 SASL 机制进行一致性运算）。
5. 初始化实体向接收实体发送符合'urn:ietf:params:xml:ns:xmpp-sasl'名字空间的<response/>元素作为应答；这个元素可以（MAY）包含 XML 字符数据（必须按照初始化实体选择的 SASL 机制进行一致性运算）。
6. 如果必要，接收实体发送更多的挑战给初始化实体，初始化实体发送更多的回应。

这一系列的 挑战/应答 组，持续进行直到发生以下三件事中的一件为止：

1. 初始化实体向接收实体发送符合'urn:ietf:params:xml:ns:xmpp-sasl'名字空间的<abort/>元素以中止握手。在接收到<abort/>元素之后，接收实体应该（SHOULD）允许一个可配置的但是合理的重试次数（至少 2 次），然后它必须（MUST）终止 TCP 连接；这使得初始化实体（如一个最终用户客户端）能够容忍可能不正确的 credentials（如密码输入错误）而不用强制重新连接。
2. 接收实体向初始化实体发送符合'urn:ietf:params:xml:ns:xmpp-sasl'名字空间的<failure/>元素以报告握手失败（详细的失败原因应该在<failure/>的一个适当的子元素中沟通，在第六章第四节中的 SASL Errors 中定义）。如果失败的情况发生了，接收实体应该（SHOULD）允

许一个可配置的但是合理的重试次数（至少 2 次）， 然后它必须（MUST）终止 TCP 连接；这使得初始化实体（如一个最终用户客户端）能够容忍可能不正确的 credentials（如密码输入错误）而不用强制重新连接。

3. 接收实体向初始化实体发送符合'urn:ietf:params:xml:ns:xmpp-sasl'名字空间的<success/>元素以报告握手成功；如果所选择的 SASL 机制要求，这个元素可以（MAY）包含 XML 字符数据（见 SASL 术语，“成功的额外数据”）。接收到<success/> 元素之后，初始化实体必须（MUST）发送一个打开的 XML 流头信息给接收实体以发起一个新的的流（不需要先发送一个 </stream>标签，因为在发送和接收到<success/>元素之后，接收实体和初始化实体必须确认原来的流被关闭了）。从初始化实体接收到新的流头信息之后，接收实体必须（MUST）发送一个新的流头信息给初始化实体作为回应，附上任何可用的特性（但不包括 STARTTLS 和 SASL 特性）或一个空的 <features/> 元素（这表示没有更多的特性可用）；任何没有在本文定义的附加特性必须（MUST）在 XMPP 的相关扩展中定义。

6.3. SASL 定义

[SASL]的必要条件要求通过协议定义来提供以下信息:

service name（服务名）: "xmpp"

initiation sequence（开始序列）: 当初始化实体提供一个打开的 XML 流头信息并且接收实体善意回应之后，接收实体提供一个可接受的验证方法清单。初始化实体从这个清单中选择一个方法，把它作为 <auth/> 元素的 'mechanism' 属性的值发送给接收实体，也可以选择发送一个初始化应答以避免循环。

exchange sequence（交换序列）: 挑战和回应的交换，从接收实体发送给初始化实体的 <challenge/> 元素和从初始化实体发送给接收实体的 <response/> 元素信息。接收实体通过发送 <failure/>元素报告失败，发送<success/>元素报告成功；初始化实体通过发送<abort/> 元素中止交换。成功的协商之后，两边都认为原来的 XML 流已经关闭并且都开始发送新的流头信息。

security layer negotiation（安全层协商）: 安全层在接收实体发送 <success/> 元素的关闭字符">"之后立刻生效，在初始化实体发送 <success/> 元素的关闭字符">"之后也立刻生效。层的顺序是 [TCP],[TLS],然后是 [SASL],然后是 [XMPP]。

use of the authorization identity（授权 ID 的使用）: 授权 ID 可在 xmpp 中用于表示一个客户端的非缺省的<node@domain>，或一个服务器的<domain> 。

6.4. SASL 错误

SASL 相关的错误条件定义如下:

- <aborted/> -- 接收实体认可由初始化实体发送的<abort/>元素；在回应一个<abort/>元素时发送。
- <incorrect-encoding/> -- 由初始化实体提供的数据无法处理，因为[BASE64]编码不正确（例如，因为编码不符合[BASE64]的第三章）；在回应一个包含初始化响应数据的<response/> 元素或<auth/>元素时发送.
- <invalid-authzid/> -- 由初始化实体提供的授权 id 是非法的，因为它的格式不正确或初始化实体无权给那个 ID 授权；在回应一个包含初始化响应数据的<response/> 元素或<auth/>元素时发送。
- <invalid-mechanism/> -- 初始化实体不能提供一个机制活、或请求一个不被接受实体支持的机制；在回应一个<auth/>元素时发送。
- <mechanism-too-weak/> -- 初始化实体请求的机制比服务器策略对它的要求弱；在回应一个包含初始化响应数据的<response/> 元素或<auth/>元素时发送。
- <not-authorized/> -- 验证失败，因为初始化实体没有提供合法的 credentials（这包括但不限于未知用户名等情形）；在回应一个包含初始化响应数据的<response/> 元素或<auth/>元素时发送。
- <temporary-auth-failure/> -- 验证失败，因为接收实体出现了临时的错误；在回应一个<response/> 元素或<auth/>元素时发送。

6.5. 客户端-服务器 示例

以下例子展示了一个客户端和一个服务器使用 SASL 作验证的数据流，通常是在成功的 TLS 握手之后（注意：以下显示的替代步骤仅用于举例说明协议的失败情形；它们不够详尽也不需要由例子中的数据传送来触发）。

步骤 1: 客户端初始化流给服务器:

```
<stream:stream
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       to='example.com'
       version='1.0'>
```

步骤 2: 服务器向客户端发送流标签作为应答:

```
<stream:stream
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       id='c2s_234'
       from='example.com'
       version='1.0'>
```

步骤 3: 服务器通知客户端可用的验证机制:

```
<stream:features>

     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
```

```
        <mechanism>DIGEST-MD5</mechanism>

        <mechanism>PLAIN</mechanism>

      </mechanisms>

   </stream:features>
```

步骤 4: 客户端选择一个验证机制:

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='DIGEST-MD5'/>
```

步骤 5: 服务器发送一个 [BASE64] 编码的挑战给客户端:

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgi
   LGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNzCg==
   </challenge>
```

解码后的挑战信息是:

```
realm="somerealm",nonce="OA6MG9tEQGm2hh",
   qop="auth",charset=utf-8,algorithm=md5-sess
```

步骤 5 (替代): 服务器返回一个错误给客户端:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

      <incorrect-encoding/>

   </failure>

   </stream:stream>
```

步骤 6: 客户端发送一个[BASE64]编码的回应这个挑战:

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
   dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0i
   T0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5jPTAw
   MDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20i
   LHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNo
   YXJzZXQ9dXRmLTgK
   </response>
```

解码后的回应信息是:

```
username="somenode",realm="somerealm",
   nonce="OA6MG9tEQGm2hh",cnonce="OA6MHXh6VqTrRk",
   nc=00000001,qop=auth,digest-uri="xmpp/example.com",
   response=d388dad90d4bbd760a152321f2143af7,charset=utf-8
```

步骤 7: 服务器发送另一个[BASE64]编码的挑战给客户端:

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
```

```
   cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3YjgOZGJhYmNkZmZmZAo=
   </challenge>
```

解码后的挑战信息是:

```
rspauth=ea40f60335c427b5527b84dbabcdfffd
```

步骤 7 (或者): 服务器返回一个错误给客户端:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

     <temporary-auth-failure/>

   </failure>

   </stream:stream>
```

步骤 8: 客户端应答这个挑战:

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

步骤 9: 服务器通知客户端验证成功:

```
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

步骤 9 (或者): 服务器通知客户端验证失败:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

     <temporary-auth-failure/>

   </failure>

   </stream:stream>
```

步骤 10: 客户端发起一个新的流给服务器:

```
<stream:stream
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       to='example.com'
       version='1.0'>
```

步骤 11: 服务器发送一个流头信息回应客户端,并附上任何可用的特性(或空的 features 元素):

```
<stream:stream
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       id='c2s_345'
       from='example.com'
       version='1.0'>

   <stream:features>
```

```
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>

     <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>

   </stream:features>
```

6.6. 服务器-服务器 示例

以下例子展示了一个服务器和另一个服务器使用 SASL 作验证的数据流，通常是在成功的 TLS 握手之后（注意：以下显示的替代步骤仅用于举例说明协议的失败情形；它们不够详尽也不需要由例子中的数据传送来触发）。

步骤 1: 服务器 1 发起一个流给 服务器 2 :

```
<stream:stream
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'
       to='example.com'
       version='1.0'>
```

步骤 2: 服务器 2 回应一个流标签给 服务器 1:

```
<stream:stream
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'
       from='example.com'
       id='s2s_234'
       version='1.0'>
```

步骤 3: 服务器 2 通知 服务器 1 可用的验证机制:

```
<stream:features>

     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

       <mechanism>DIGEST-MD5</mechanism>

       <mechanism>KERBEROS_V4</mechanism>

     </mechanisms>

   </stream:features>
```

步骤 4: 服务器 1 选择一个验证机制:

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
        mechanism='DIGEST-MD5'/>
```

步骤 5: 服务器 2 发送一个[BASE64]编码的挑战给 服务器 1:

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
```

```
   cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9
   ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNz
   </challenge>
```

解码后的挑战信息是:

```
realm="somerealm",nonce="OA6MG9tEQGm2hh",
   qop="auth",charset=utf-8,algorithm=md5-sess
```

步骤 5 (替代): 服务器 2 返回一个错误给 服务器 1:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

     <incorrect-encoding/>

   </failure>

   </stream:stream>
```

步骤 6: 服务器 1 发送一个[BASE64]编码的回应这个挑战:

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
   dXNlcm5hbWU9ImV4YW1wbGUub3JnIixyZWFsbT0ic29tZXJlYWxtIixub25j
   ZT0iT0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5j
   PTAwMDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5v
   cmciLHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3
   LGNoYXJzZXQ9dXRmLTgK
   </response>
```

解码后的应答信息是:

```
username="example.org",realm="somerealm",
   nonce="OA6MG9tEQGm2hh",cnonce="OA6MHXh6VqTrRk",
   nc=00000001,qop=auth,digest-uri="xmpp/example.org",
   response=d388dad90d4bbd760a152321f2143af7,charset=utf-8
```

步骤 7: 服务器 2 发送另外一个[BASE64]编码的挑战给 服务器 1:

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
   cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3YjgOZGJhYmNkZmZmZAo=
   </challenge>
```

解码后的挑战信息是:

```
rspauth=ea40f60335c427b5527b84dbabcdfffd
```

步骤 7 (或者): 服务器 2 返回一个错误给 服务器 1:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

     <invalid-authzid/>

   </failure>

   </stream:stream>
```

步骤 8: 服务器 1 回应挑战:

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

步骤 8 (或者): 服务器 1 中止协商:

```
<abort xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

步骤 9: 服务器 2 通知 服务器 1 验证成功:

```
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

步骤 9 (或者): 服务器 2 通知 服务器 1 验证失败:

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>

     <aborted/>

   </failure>

   </stream:stream>
```

步骤 10: 服务器 1 重新发起一个新的流给 服务器 2:

```
<stream:stream
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'
       to='example.com'
       version='1.0'>
```

Step 11: 服务器 2 发送一个流头信息应答 服务器 1 ,并附上任何可用的特性(或一个空的 features 元素).:

```
<stream:stream
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       from='example.com'
       id='s2s_345'
       version='1.0'>

   <stream:features/>
```

---

7. 资源绑定

在和接收实体完成 SASL 协商(第六章)之后,初始化实体可能(MAY)想要或者需要绑定一个特定的资源到流上.通常这仅适用于客户端: 为了满足本文定义的寻址格式(第三章)和节传输规则(第十章),客户端<node@domain>必须(MUST)拥有一个相关的资源 ID（由服务器生成或由客户端程序提供）；以确保在流上使用的地址是一个“全 JID”（<node@domain/resource>）。

接收到一个成功的 SASL 握手之后，客户端必须（MUST）发送一个新的流头信息给服务器，服务器必须（MUST）返回一个包含可用的流特性列表的头信息。特别是,在成功的 SASL 握手之后如果服务器需要客户端绑定一个资源,它必须(MUST)

在握手成功之后（而不是之前）发送给客户端的应答流特性中包含一个空的符合'urn:ietf:params:xml:ns:xmpp-bind'名字空间的<bind/>元素。:

服务器向客户端声明资源绑定特性:

```
<stream:stream
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       id='c2s_345'
       from='example.com'
       version='1.0'>

   <stream:features>

     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>

   </stream:features>
```

收到要求资源绑定的通知后，客户端必须（MUST）通过发送一个符合'urn:ietf:params:xml:ns:xmpp-bind'名字空间的“set”类型的 IQ 节(参见 IQ 语义 (第九章第二节第三小节))给服务器来绑定一个资源到流中。

如果客户端端希望允许服务器给自己生成一个资源 ID，它可以发送一个包含空的<bind/>元素的“set”类型的 IQ 节。:

客户端请求服务器绑定资源:

```
<iq type='set' id='bind_1'>

     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>

   </iq>
```

一个支持资源绑定的服务器必须（MUST）自动生成一个资源 ID 给客户端。一个由服务器生成的资源 ID 对于那个<node@domain>必须（MUST）是唯一的。

如果客户端希望指定资源 ID，它发送一个包含期望资源 ID 的“set”类型的 IQ 节，把资源 ID 作为<bind/>元素下的<resource/>子元素的 XML 字符数据:

客户端绑定一个资源:

```
<iq type='set' id='bind_2'>

     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>

       <resource>someresource</resource>
```

```
    </bind>

  </iq>
```

一旦服务器为客户端生成了一个资源 ID 或接受了客户端自己提供的资源 ID，它必须（MUST）返回一个“result”类型的 IQ 节给客户端，这个节必须包含一个指明全 JID 的<jid/>子元素表示服务器决定连接的资源:

服务器通知客户端资源绑定成功:

```
<iq type='result' id='bind_2'>

    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>

      <jid>somenode@example.com/someresource</jid>

    </bind>

  </iq>
```

一个服务器应该（SHOULD）接受客户端提供的资源 ID，但是可以（MAY）用服务器生成的资源 ID 覆盖它；在这种情况下，服务器不应该（SHOULD NOT）返回一个错误信息 (如<forbidden/>)给客户端，而应该（SHOULD）在上文所述的 IQ result 中返回生成的资源 ID。

当一个客户端自行提供资源 ID 时，可能发生以下的节错误(参见 Stanza Errors (第九章第三节)):

- 提供的资源 ID 服务器无法处理，因为不符合资源字符规范 Resourceprep（附录 B）。
- 客户端不被允许绑定一个资源(如节点或用户已经达到允许连接的资源数限制）。
- 提供的资源 ID 已经被使用但是服务器不允许以同一个资源 ID 绑定多个连接。

协议对这些错误条件规定如下.

资源 ID 不能处理:

```
<iq type='error' id='bind_2'>

    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>

      <resource>someresource</resource>

    </bind>
```

```
    <error type='modify'>
      <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </iq>
```

客户端不允许绑定一个资源:

```
<iq type='error' id='bind_2'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
      <resource>someresource</resource>
    </bind>
    <error type='cancel'>
      <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </iq>
```

资源 ID 已经在使用:

```
<iq type='error' id='bind_2'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
      <resource>someresource</resource>
    </bind>
    <error type='cancel'>
      <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </iq>
```

如果,在完成资源绑定步骤之前,客户端试图以符合'urn:ietf:params:xml:ns:xmpp-bind'名字空间的<bind/>子元素发送一个非 IQ 类型的 XML 节,服务器不能(MUST NOT)处理这个节,而应该(SHOULD)返回一个<not-authorized/>节错误信息给客户端.

---

8. 服务器回拨

8.1. 概览

Jabber 协议接受的来自 XMPP 的包括“服务器回拨”方法，用于防治域名欺骗，使得欺骗 XML 节更为困难。服务器回拨不是一个安全机制，并且它的服务器身份认证结果很弱（参见服务器之间的通信（第十四章第四节）中关于这个方法的安全特性）。需要健壮的安全性的域名应该（SHOULD）使用 TLS 或 SASL；细节参见服务器间的通信（第十四章第四节）。如果 SASL 用于服务器间通信，回拨就不需要用了（SHOULD NOT）。本文描述回拨的好处在于向后兼容现存的实现和部署。

服务器回拨方法由现存的 DNS 系统的存在而成为可能，因为一个服务器（通常）可以查询给定域的授权服务器。由于服务器回拨依赖于 DNS，除非服务器宣称的 DNS 主机得到解析，域之间的通信无法进行（参见服务器间的通信（第十四章第四节））。

服务器回拨是单向性的，可在单一方向上对一个流进行（微弱的）身份验证. 因为服务器回拨不是一个验证机制，通过回拨获得相互的认证是不可能的. 因此，为了使得服务器之间的双向通信，服务器回拨必须（MUST）在每个方向上完成。

在服务器回拨中生成和检验密钥的方法必须（MUST）考虑计算被使用的主机名，由接收服务器生成的流 ID，和只有授权服务器网络才知道的秘密。在服务器回拨中流 ID 是安全性的关键所以必须（MUST）是不可预知的和不可重复的（见 [RANDOM]中关于使用随机数获得安全性的建议）.

任何回拨协商过程中发生的错误必须（MUST）被当成一个流错误，并导致流以及相关的 TCP 连接的终止. 可能发生的错误情况协议中描述如下.

以下术语适用:

- 发起服务器 -- 尝试在两个域之间建立连接的那个服务器.
- 接收服务器 -- 尝试验证发起服务器声称的域名的那台服务器.
- 授权（权威? Authoritative） 服务器 -- 回答发起服务器声称的 DNS 主机名的服务器；基本上这应该是那台发起服务器，但是它也可能是一个在发起服务器网络中独立的服务器。

8.2. 事件顺序

以下是回拨中的事件顺序的简介:

1. 发起服务器和接收服务器建立一个连接。
2. 发起服务器通过到连接发送一个 ’key’ 值给接收服务期。
3. 接收服务器建立一个连接到授权服务器。

4. 接收服务器发送一个相同的 'key' 值给授权服务器。
5. 授权服务器回答这个 key 是否合法。
6. 接收服务器通知发起服务器是否被验证通过。

我们用用以下图形展示这个事件流程(见附件 s2s.png):

8.3. 协议

服务器之间互动的细节协议如下:

1. 发起服务器和接受服务器建立 TCP 连接.

2. 发起服务器发送流头信息给接收服务器:

```
<stream:stream
       xmlns:stream='http://etherx.jabber.org/streams'
       xmlns='jabber:server'
       xmlns:db='jabber:server:dialback'>
```

注意: 'to'和'from'属性在流的根元素是可选的(OPTIONAL). 其中包含的 xmlns:db 名字空间向接收服务器声明了发起服务器支持回拨. 如果名字空间不正确，接收实体必须（MUST）生成一个<invalid-namespace/>流错误条件并且终止 XML 流和相应的 TCP 连接.

3. 接收服务器应该（SHOULD）回送一个流头信息给发起服务器，为这次交互生成一个唯一性的 ID :

```
<stream:stream
       xmlns:stream='http://etherx.jabber.org/streams'
       xmlns='jabber:server'
       xmlns:db='jabber:server:dialback'
       id='457F9224A0...'>
```

注意: 'to'和'from'属性在流的根元素是可选的(OPTIONAL). 如果名字空间不正确，发起服务器必须生成一个<invalid-namespace/>流错误条件并且终止 XML 流和相应的 TCP 连接.也要注意，在这里接收服务器应该（SHOULD）应答但是可以(MAY)出于安全策略考虑只是悄悄地终止 XML 流和 TCP 连接；无论如何，如果接收服务器希望继续，它必须（MUST）回送一个流头信息给发起服务器.

4. 发起服务器发送一个回拨密钥给接收服务器:

```
<db:result
       to='Receiving Server'
       from='Originating Server'>
```

98AF014EDC0…
</db:result>
注意: 这个密钥不由接收服务器检查，因为接收服务器在会话之间（between sessions)不保存发起服务器的信息. 这个由发起服务器生成的密钥必须(MUST)是基于接收服务器在上一步骤中提供的 ID 值，以及发起服务器与授权服务器共享的安全机制生成的。如果 'to' 地址的值和接收服务器知道的主机名不匹配，接收服务器必须（MUST）生成一个<host-unknown/>流错误条件并且终止 XML 流和相应的 TCP 连接. 如果 'from' 地址和接收服务器已经建立的连接的域名相吻合,接收服务器必须(MUST)维护这个已经存在的连接,直到证明这个新的连接是合法的为止;另外,接收实体可以(MAY)选择生成一个<not-authorized/> 流错误条件给这个新的连接并且终止和新连接申请相关的 XML 流及相应的 TCP 连接.

5. 接收服务器向发起服务器声明的那个域建立一个 TCP 连接,作为结果它连接到授权服务器. (注意: 为了优化性能, 在这里一个实现可以(MAY)重用现有的连接.)

6. 接收服务器发送一个流头信息给授权服务器:

```
<stream:stream
       xmlns:stream='http://etherx.jabber.org/streams'
       xmlns='jabber:server'
       xmlns:db='jabber:server:dialback'>
```

注意: 'to'和'from'属性在流的根元素是可选的(OPTIONAL). 如果名字空间不正确，授权服务器必须生成一个<invalid-namespace/>流错误条件并且终止 XML 流和相应的 TCP 连接.

7. 授权服务器发送流头信息给接收服务器:

```
<stream:stream
       xmlns:stream='http://etherx.jabber.org/streams'
       xmlns='jabber:server'
       xmlns:db='jabber:server:dialback'
       id='1251A342B...'>
```

注意: 如果名字空间不正确，接收服务器必须生成一个<invalid-namespace/>流错误条件并且终止它和授权服务器之间的 XML 流和相应的 TCP 连接. 如果一个流错误发生在接收服务器和 **授权服务器** 之间,接收服务器必须(MUST)生成一个<remote-connection-failed/> 流错误条件并且终止它和 **发起服务器** 之间的 XML 流和相应的 TCP 连接.

8. 接收服务器发送一个密钥检查请求给授权服务器:

```
<db:verify
       from='Receiving Server'
       to='Originating Server'
       id='457F9224A0...'>
```

98AF014EDC0…
</db:verify>
注意: 现在这里已经有了主机名,第三步中接收服务器发送给发起服务器的原始,第四步中发起服务器发送给接收服务器的密钥. 基于这些信息, 加上授权服务器网络共享的安全信息, 这个密钥被证实了.任何可用于验证的办法都可以（MAY)用于生成密钥. 如果 'to' 地址的值和授权服务器知道的主机名不匹配，授权服务器必须（MUST）生成一个<host-unknown/>流错误条件并且终止 XML 流和相应的 TCP 连接. 打开这个连接时,如果 'from' 地址和接收服务器声明的主机名（或任何合法的域，如验证过的接收服务器的子域名或寄宿在接收服务器上的其他经过验证的域）不符,授权服务器必须(MUST)生成一个<invalid-from/>流错误条件并且终止 XML 流和相应的 TCP 连接.

9. 授权服务器检查密钥是否合法:

```
<db:verify
       from='Originating Server'
       to='Receiving Server'
       type='valid'
       id='457F9224A0...'/>
```

或

```
<db:verify
       from='Originating Server'
       to='Receiving Server'
       type='invalid'
       id='457F9224A0...'/>
```

注意: 如果 ID 和第三步中接收服务器提供的不符，接收服务器必须（MUST）生成一个<invalid-id/>流错误条件并且终止 XML 流和相应的 TCP 连接. 如果 'to' 地址的值和接收服务器知道的主机名不匹配，接收服务器必须（MUST）生成一个<host-unknown/>流错误条件并且终止 XML 流和相应的 TCP 连接. 打开这个连接时，如果 'from' 地址和发起服务器声明的主机名（或任何合法的域，如验证过的发起服务器的子域名或寄宿在发起服务器上的其他经过验证的域）不符,接收服务器必须(MUST)生成一个<invalid-from/>流错误条件并且终止 XML 流和相应的 TCP 连接. 在返回验证信息给接收服务器之后，授权服务器应该（SHOULD）终止它们之间的流.

10. 接收服务器通知发起服务器结果:

```
<db:result
       from='Receiving Server'
       to='Originating Server'
       type='valid'/>
```

注意: 在这一个点上, 连接已经由 type='valid' 确认验证是通过了,还是没通过. 如果连接是非法的,接收服务器必须(MUST)终止 XML 流和相应的 TCP 连接. 如果连接是合法的, 数据可以从发起服务器发送由接收服务器读取; 在此之前,所有发送给接收服务器的 XML 节应该（SHOULD）被丢弃.

进一步的结果是，接收服务器已经验证了发起服务器的 ID，所以通过初始化流("initial stream"，例如从发起服务器到接收服务器的流)发起服务器可以发送，接收服务器可以接受 XML 节. 为了使用应答流（"response stream"，例如从接收服务器到发起服务器的流）验证实体 ID，回拨必须（MUST）在相对的两个方向上都完成.

在成功的回拨协商之后，接收服务器应该（SHOULD）接受接下来发起服务器通过当前的合法连接发送的 <db:result/> 包（例如，发送给子域或其他寄宿在接收服务器上的主机名的验证请求)；这在单一方向上激活了原始合法连接的"piggybacking" .

即使回拨协商成功了，服务器仍然必须（MUST）检查从其他服务器接收的所有 XML 节的'from'和'to'属性；如果一个节不符合这些限定，收到这些节的服务器必须（MUST）生成一个<improper-addressing/> 流错误条件并终止 XML 流和相应的 TCP 连接. 而且，一个服务器也必须(MUST)检查从其他的有合法域名的服务器的流中收到的节的'from'属性；如果一个节不符合这一限定，接收节的服务器必须(MUST)生成一个 <invalid-from/> 流错误条件并终止 XML 流和相应的 TCP 连接. 所有这些检查都用来帮助防止特定的节伪造行为.

---

9. XML 节

在 TLS 协商(第五章)（如果想要)，SASL 协商(第六章)，和资源绑定(第七章)（如果需要）之后，XML 节就可以通过流发送了. 在'jabber:client'和'jabber:server'名字空间中定义了三种 XML 节:<message/>,<presence/>，和<iq/>. 另外，这三种节有五种通用的属性. 这些通用属性，加上三种节的术语，在这里定义；更多关于和即时消息及出席信息应用相关的 XML 节语法详细信息在 XMPP-IM 中提供.

9.1. 通用属性

以下五种属性通用于 message，presence，和 IQ 节:

9.1.1. to

'to' 属性表示节的预期接收者的 JID.

在'jabber:client'名字空间中，一个节应该（SHOULD）处理一个'to'属性，尽管由服务器处理的从客户端发给服务器的节(如，发送给服务器用于广播给其他实体的出席信息）应该不（SHOULD NOT）处理'to'属性.

在'jabber:server'名字空间中，一个节必须（MUST）处理一个'to'属性；如果一个服务器收到一个不符合此限定的节，它必须（MUST）生成一个

<improper-addressing/>流错误条件并终止和这个非法服务器的 XML 流和相应的 TCP 连接.

如果'to'属性的值非法或无法联络, 发现这个事实的实体(通常是发送者或接收者的服务器)必须(MUST)返回一个适当的错误给发送者, 错误节的'from'属性设置成非法节的提供的'to'属性的值.

9.1.2. from

'from' 属性表示发送者的 JID .

当一个服务器接收了一个符合'jabber:client'名字空间的合法流的 XML 节, 它必须(MUST)做以下步骤中的一步:

1. 验证客户端提供的'from'属性值就是那个相关实体连接的资源
2. 为生成这个节的已连接的资源增加一个'from'地址(由服务器决定是纯 JID 或全 JID)(参见 地址的决定 Determination of Addresses (第三章第五节))

如果一个客户端试图发送一个 XML 节,而它的'from'属性和这个实体已连接的资源不符, 服务器应该(SHOULD)返回一个<invalid-from/>流错误条件给客户端. 如果一个客户端试图通过一个尚未验证的流发送一个 XML 节, 服务器应该(SHOULD)返回一个<not-authorized/>流错误条件给客户端. 如果生成了, 所有这些条件必须(MUST)导致流的关闭和相应的 TCP 连接的终止; 这有助于防止不诚实的客户的拒绝服务攻击.

当一个服务器从服务器自身生成一个节用于一个已连接的客户端的信息发布(例如, 在服务器为客户端提供数据存储服务的情况下), 这个节必须(MUST) (1) 不包含 'from' 属性 或 (2) 包含一个'from'属性,它的值是这个账号的纯 JID (<node@domain>) 或 客户端的全 JID (<node@domain/resource>). 如果节不是由服务器自身生成的, 那么一个服务器不能(MUST NOT)发送不带'from'属性的节. 当一个客户端接收到一个不包含'from'属性的节, 它必须(MUST)认为这个节是从客户端连接的服务器发来的.

在'jabber:server'名字空间, 一个节必须(MUST)处理一个'from'属性; 如果一个服务器接收到一个不符合此限定的节, 它必须(MUST)生成一个<improper-addressing/>流错误条件. 而且, 'from'属性的 JID 值的域名 ID 部分必须(MUST)和以 SASL 协商连接或以回拨协商连接的发送服务器的主机名(或任何合法的域,如发送服务器主机名的合法子域,或其他寄宿在发送服务器上的合法域)吻合; 如果一个服务器接收到的节不符合此限定, 它必须(MUST)生成一个<invalid-from/>流错误条件. 所有这些条件都必须(MUST)导致流的关闭和相应的 TCP 连接的终止; 这有助于防止不诚实的客户端发起的拒绝服务攻击.

9.1.3. id

可选的'id' 属性可以(MAY)用于为节的内部跟踪发送实体，从 IQ 节 语义来讲，就是通过发送和接收这些节来跟踪“请求-应答”型的交互行为。这个可选的（OPTIONAL）'id'属性值在一个域或一个流中是全局唯一的。IQ 节语义学中对此有附加限定；见 IQ Semantics (第九章第二节第三小节)。

9.1.4. type

'type' 属性指明消息、出席信息或 IQ 节的意图或上下文的详细信息。'type'属性所允许的值依据节的类型是消息、出席信息还是 IQ 而有很大不同； 用于消息和出席信息节的值定义在即时消息和出席信息应用中，所以在 XMPP-IM 中定义，反之用于 IQ 节的值定义了在一个 请求-应答 的“会话”中 IQ 节的角色，所以定义在 IQ 语义学中（第九章第二节第三小节）。 所有三种节的通用'type'值是"error"；见 Stanza Errors (第九章第三节)。

9.1.5. xml:lang

如果一个节包含用于显示给人 human user 看的 XML 字符数据(在 RFC 2277 中有所解释[CHARSET],"internationalization is for humans")，这个节应该（SHOULD）处理一个'xml:lang'属性(定义在第二章第十二节[XML])。'xml:lang'属性的值指明任何一个人类可读的 XML 字符数据的缺省语言， 它可以（MAY）被特定的子元素的'xml:lang'值重载。 如果一个节不处理一个'xml:lang'属性,一个实现必须(MUST)认为缺省的语言就是流属性中定义的语言(第四章第四节). 'xml:lang'属性值必须(MUST)是一个 NMTOKEN 并且必须(MUST)遵守 RFC 3066 [LANGTAGS]中定义的格式.

9.2. 基本语义学

9.2.1. 消息语义学

<message/>节类型可以被看作是一个"push"机制用于一个实体推送信息给另一个实体,类似发生在 email 系统中的通信. 所有消息节应该（SHOULD）处理一个表明预定的消息接收者的'to'属性;接收了这样一个节之后,一个服务器应该（SHOULD）路由或递送它给预定的接收者(见 Server Rules for Handling XML Stanzas (第十章)的 XML 节相关的通用路由和递送规则).

9.2.2. 出席信息语义学

<presence/> 元素可以被看作一个基本的广播或“出版-订阅”机制,用于多个实体接收某个已订阅的实体的信息(在这里,是网络可用性信息). 通常,一个发

行实体应该（SHOULD）不带'to'属性发送一个出席信息,这时这个实体所连接的服务器应该（SHOULD）广播或多播(multiplex?)那个节给所有订阅的实体.无论如何,一个发行实体也可以(MAY)带'to'属性发送一个出席信息节,这时服务器应该(SHOULD)路由或递送这个节给预定的接收者.见 Server Rules for Handling XML Stanzas (第十章)的 XML 节相关的通用路由和递送规则,以及 XMPP-IM 中即时消息和出席信息应用中出席信息的特定规则.

9.2.3. IQ 语义学

信息/查询(Info/Query),或曰 IQ,是一个 请求-回应 机制,某些情况下类似[HTTP].IQ 语义学使一个实体能够向另一个实体做出请求并做出应答. 请求和应答所包含的数据定义在 IQ 元素的一个直接的子元素的名字空间声明中, 并且由请求实体用'id'属性来跟踪这一交互行为. 因而,IQ 交互伴随着一个结构化的数据交换的通用模式例如 get/result 或 set/result (尽管有时候会以一个错误信息应答某个请求):

以下是 IQ 的流程图，参见附件 iq.png

为了强制执行这些语义学,要应用以下规则:

1. 对于 IQ 节来说'id'属性是必需的（REQUIRED）.

2. 对于 IQ 节来说'type'属性是必需的（REQUIRED）. 它的值必须（MUST）是以下之一:

- get -- 这个节是一个对信息或需求的请求.
- set -- 这个节提供需要的数据, 设置新的值, 或取代现有的值.
- result -- 这个节是一个对一个成功的 get 或 set 请求的应答.
- error -- 发生了一个错误，关于处理或递送上次发送的 get 或 set 的(参见 节错误 Stanza Errors(第九章第三节)).

3. 一个接收到"get" 或 "set" 类型的 IQ 请求的实体必须（MUST）回复一个"result"或"error"类型的 IQ 应答(这个应答必须（MUST）保留相关请求的'id'属性).

4. 一个接收到"result"或"error"类型的 IQ 节的实体不能（MUST NOT）再发送更多的"result"或"error"类型的 IQ 应答; 无论如何, 如上所述, 请求实体可以（MAY）发送另一个请求(如, 一个"set"类型的 IQ，通过 get/result 对提供查询(discovery)所需的信息).

5. 一个"get" 或 "set" 类型的 IQ 节必须（MUST）包含并只包含一个子元素指明特定请求或应答的语义.

6. 一个"result"类型的 IQ 节必须（MUST）包含零或一个子元素.

7. 一个"error"类型的 IQ 节应该（SHOULD）包含和"get"或"set"相关联的那个子元素并且必须（MUST）包含一个<error/>子元素;详细信息,见 Stanza Errors (第九章第三节).

9.3. 节错误

节相关的错误的处理方式类似流错误(第四章第七节). 无论如何, 不像流错误, 节错误是可恢复的;所以错误节包含了暗示原来的发送者可以采取什么行动来补救这个错误.

9.3.1. 规则

以下规则适用于节相关的错误:

- 接收或处理实体察觉到一个节相关的错误条件时应该（MUST）返回给发送实体一个同类型的节(消息,出席信息,或 IQ),这个节的'type'属性值则设置为"error"(这里这样的节称之为"error stanza").
- 生成一个错误节的实体应该（SHOULD）包含原来发送的 XML，这样发送者可以检查它，并且如果必要，在尝试重新发送之前纠正它.
- 一个错误节必须（MUST）包含一个<error/>子元素.
- 如果'type'属性值不是"error"（或没有"type"属性），节不能(MUST NOT)包含一个<error/>子元素.
- 接收到一个错误节的实体不能(MUST NOT)应答这个节更多的错误节; 这有助于防止死循环.

9.3.2. 语法

节相关错误的语法如下:

```
<stanza-kind to='sender' type='error'>

     [RECOMMENDED to include sender XML here]

     <error type='error-type'>

       <defined-condition
xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
```

```
        <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'
              xml:lang='langcode'>
          OPTIONAL descriptive text
        </text>
        [OPTIONAL application-specific condition element]
      </error>
    </stanza-kind>
```

stanza-kind 是 message, presence, 或 iq 中的一个.

<error/> 元素的'type'属性值必须(MUST)是以下之一:

- cancel -- 不重试(这个错误是不可恢复的)
- continue -- 继续进行(这个条件只是一个警告)
- modify -- 改变数据之后重试
- auth -- 提供证书之后重试
- wait -- 等待之后重试(错误是暂时的)

<error/>元素:

- 必须(MUST)包含一个子元素,符合以下定义的节错误条件之一;这个元素(MUST)符合'urn:ietf:params:xml:ns:xmpp-stanzas'名字空间.
- 可以(MAY)包含一个<text/>子元素容纳 XML 字符数据用来描述错误的更多细节;这个元素必须(MUST)符合'urn:ietf:params:xml:ns:xmpp-stanzas'名字空间并且应该(SHOULD)处理'xml:lang'属性.
- 可以(MAY)包含一个应用程序定义的错误条件子元素;这个元素必须(MUST)符合一个应用程序定义的名字空间,并且它的结构由这个名字空间定义.

<text/>元素是可选的(OPTIONAL).如果包含它,它应该(SHOULD)仅用于提供描述或诊断信息以补充一个已定义的条件或应用程序定义的条件. 它不应(SHOULD NOT)被应用程序认为是一个程序性的. 它不应(SHOULD NOT)被用作向一个使用者展示的错误信息,但是可以(MAY)展示除条件元素(或元素们)相关的错误信息之外的信息.

最后,为了维护向后兼容性, 这个 schema (定义在 XMPP-IM)允许可选的在<error/>元素中包含一个("code")属性.

9.3.3. 已定义的条件

以下条件定义用于节错误.

- <bad-request/> -- 发送者发送的 XML 是不规范的或不能被处理(例如 一个 IQ 节包含了一个未被承认的'type'属性值); 相关的错误类型应该（SHOULD）是"modify".
- <conflict/> -- 不同意访问，因为相同的名字或地址已存在一个资源或会话;相关的错误类型应该（ SHOULD）是"cancel".
- <feature-not-implemented/> -- 请求的特性未被接收者或服务器实现所以不能处理;相关的错误类型应该（SHOULD）是"cancel".
- <forbidden/> -- 请求实体没有必需的许可来执行这一动作;相关的错误类型应该（SHOULD）是"auth".
- <gone/> -- 接收者或服务器无法再以这个地址进行联系(错误节可以（MAY)在<gone/>元素的 XML 字符数据中包含一个新的地址);相关的错误类型应该（SHOULD）是"modify".
- <internal-server-error/> -- 服务器不能处理节，因为错误的配置或其他未定义的内部服务器错误;相关的错误类型应该（SHOULD）是"wait".
- <item-not-found/> -- JID 地址或申请的条目无法找到;相关的错误类型应该（SHOULD）是"cancel".
- <jid-malformed/> -- 发送的实体提供的 XMPP 地址或与之通信的某个 XMPP 地址(如一个'to'属性值)或这个 XMPP 地址中的一部分(如一个资源 ID)不符合寻址方案的语法(第三章);相关的错误类型应该（SHOULD）是"modify".
- <not-acceptable/> -- 接收者或服务器理解这个请求但是拒绝处理，因为它不符合某些接收者或服务器定义的标准(例如,一个关于消息中可接收的单词的本地策略);相关错误类型应该（SHOULD）是"modify".
- <not-allowed/> -- 接收者或服务器不允许任何实体执行这个动作;相关错误类型应该（SHOULD）是"cancel".
- <not-authorized/> -- 在被允许执行某个动作之前发送者必须提供适当的证书,或已提供了不正确的证书;相关错误类型应该(SHOULD)是"auth".
- <payment-required/> -- 请求实体未被授权访问请求的服务，因为需要付费;相关错误类型应该（SHOULD）是"auth".
- <recipient-unavailable/> -- 预定的接收者暂时不可用;相关错误类型应该（SHOULD）是"wait"(注意: 如果这样做会导致泄露预定接收者的网络可用性给一个未被授权了解此信息的实体，应用程序不应该(MUST NOT)返回这个错误).
- <redirect/> -- 接收者或服务器重定向这个请求信息到另一个实体，通常是暂时的(这个错误节应该 （SHOULD）在<redirect/>元素的 XML 字符数据中包含一个预备的地址,它必须（MUST）是一个合法的 JID); 相关的错误类型应该（SHOULD）是"modify".
- <registration-required/> -- 请求实体未被授权访问请求的服务，因为需要注册;相关错误类型应该（SHOULD）是"auth".
- <remote-server-not-found/> -- 在预定的接收者的全部或部分 JID 中的一个远程服务器或服务不存在;相关错误类型应该(SHOULD)是"cancel".

- <remote-server-timeout/> -- 在预定的接收者（或需要完成的一个申请)的全部或部分 JID 中的一个远程服务器或服务无法在合理的时间内联系到;相关错误类型应该（SHOULD）是"wait".
- <resource-constraint/> -- 服务器或接收者缺乏足够的系统资源来服务请求;相关错误类型应该（SHOULD）是"wait".
- <service-unavailable/> -- 服务器或接收者目前无法提供被请求的服务;相关错误类型应该（SHOULD）是"cancel".
- <subscription-required/> -- 请求实体未被授权访问能请求的服务，因为需要订阅;相关错误类型应该（SHOULD）是"auth".
- <undefined-condition/> -- 错误条件不是本列表中定于的其他条件之一;任何错误类型可能和这个条件有关,并且它应该(SHOULD)仅用于关联一个应用程序定义的条件.
- <unexpected-request/> -- 接收者或服务器理解这个请求但是不希望是在这个时间(比如,请求的顺序颠倒);相关错误类型应该（SHOULD）是"wait".

9.3.4. 应用程序定义条件

大家知道,一个应用程序可以(MAY)通过在错误元素里包含一个适当名字空间的字元素来提供应用程序定义的节错误信息. 应用程序定义的元素应该（SHOULD）补充或进一步限定一个已定义的元素. 因而,<error/>元素将包含两个或三个子元素:

```
<iq type='error' id='some-id'>

    <error type='modify'>

      <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>

      <too-many-parameters xmlns='application-ns'/>

    </error>

  </iq>

  <message type='error' id='another-id'>

    <error type='modify'>

      <undefined-condition

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

      <text xml:lang='en'
```

```
            xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>

          Some special application diagnostic information...

        </text>

        <special-application-condition xmlns='application-ns'/>

      </error>

    </message>
```

---

10. 服务器处理 XML 节的规则

兼容的服务器实现必须（MUST）确保两个实体之间的 XML 节按次序处理.

在按次序处理的需求之外，每个服务器实现将包含它自己的递送树"delivery tree"以处理它接收到的节.这个树决定一个节是否需要路由到其他域，在内部处理，还是递送到和一个已连接的节点相关的资源. 以下规则适用:

10.1. 没有'to'地址

如果这个节没有'to'属性，服务器应该(SHOULD)为发送它的实体处理这个节.因为所有从其他服务器收到的节必须(MUST)拥有'to'属性，这个规则仅适用于从一个连接到这台服务器的已注册实体(如一个客户端)收到的节.如果这个服务器收到一个没有'to'属性的出席信息节，服务器应该(SHOULD)向那些订阅了这个发送实体的出席信息的所有实体广播它，如果可能的话(即时消息和出席信息应用程序中出席信息广播的语义定义在 XMPP-IM). 如果服务器接收到一个类型为"get" 或 "set" 的没有'to'属性的节并且它理解这个节的名字空间下的内容,它必须(MUST)为这个发送实体处理节(在这里"process"的含义是由相关的名字空间的语义所决定的)或返回一个错误给发送实体.

10.2. 外部域

如果'to'属性中的 JID 的域 ID 部分的主机名和服务器自身或其子域配置的主机名不匹配，服务器应该（SHOULD）路由这个节到外部域(取决于本地服务规定或安全策略关于域间通信的规定).有两种可能的情况:

- 一个服务器之间的流已经存在于两个域之间: 发送者的服务器通过这个已存在的流为这个外部域路由这个节到授权服务器

- 两个域之间不存在服务器间的流: 发送者服务器 (1) 解析这个外部域的主机名(定义在服务器间的通信 Server-to-Server Communications (第十四章第四

节)), (2) 在两个域之间进行服务器到服务器的流协商(定义在 Use of TLS (第五章) 和 Use of SASL (第六章)), 然后 (3) 通过这个新建的流为外部域路由这个节到授权服务器

如果路由到接收者的服务器不成功, 发送者的服务器必须(MUST)返回一个错误给发送者; 如果接收者的服务器联系上了但是从接收者的服务器递送到接收者不成功, 接收者服务器必须(MUST)通过发送者的服务器返回一个错误给发送者.

10.3. 子域

如果'to'属性中的 JID 的域 ID 部分的主机名和服务器自身配置的主机名的一个子域名匹配,服务器必须 (MUST) 自己处理这个节或路由这个节到专门负责这个子域的特定服务(如果子域被配置了),或者返回一个错误给发送者(如果子域没有配置).

10.4. 纯粹的域或特定的资源

如果'to'属性中的 JID 的域 ID 部分的主机名和服务器自身配置的主机名本身匹配,并且'to'属性中的 JID 类型是<domain> 或 <domain/resource>, 服务器(或其中定于的资源)必须 (MUST) 根据节的类型适当的处理这个节或返回一个错误节给发送者.

10.5. 同一域中的节点

如果'to'属性中的 JID 的域 ID 部分的主机名和服务器自身配置的主机名本身匹配,并且'to'属性中的 JID 类型是<node@domain> 或 <node@domain/resource>, 服务器应该(SHOULD)递送这个节到节的'to'属性中的 JID 所指明的预定的接收者. 以下规则适用:

1. 如果这个 JID 包含一个资源 ID(例如, 格式是<node@domain/resource>)并且存在一个连接的资源符合这个全 JID, 接收者服务器应该 (SHOULD) 递送这个节给正确符合这个资源 ID 的流或会话.
2. 如果这个 JID 包含一个资源 ID(例如, 格式是<node@domain/resource>)并且不存在一个连接的资源符合这个全 JID, 接收者服务器应该 (SHOULD) 返回一个<service-unavailable/> 节错误给发送者.
3. 如果这个 JID 的格式是<node@domain>并且存在至少一个此节点的连接资源, 接收服务器应该 (SHOULD) 递送这个节给至少其中一个已连接的资源, 依据应用程序定义的规则(一系列即时消息和出席信息应用程序的递送规则定义在 XMPP-IM).

---

11. XMPP 中的 XML 用法

11.1. 限制

XMPP 是一个简单的流式 XML 元素的专用协议用于接近实时地交换结构化信息. 因为 XMPP 不需要任意的解析和所有的 XML 文档, 所以 XMPP 不需要支持[XML]的所有功能. 特殊的, 适用以下限制.

关于 XML 生成, 一个 XMPP 实现不能(MUST NOT)在 XML 流中注入以下任何东西:

- 注释 (第二章第五节[XML])
- 处理指示(第二章第六节 同上)
- 内部或外部的 DTD 子集 (第二章第八节 同上)
- 内部或外部的实体参考 (第四章第二节 同上) 除了预定实体以外(第四章第六节 同上)
- 字符数据或属性值包含和预定实体列表中吻合的未逃逸的 unescaped 字符(第四章第六节 同上); 这些字符必须 (MUST) 逃逸

关于 XML 处理, 如果一个 XMPP 实现接收到这些受限的 XML 数据,它必须 (MUST) 忽略这些数据.

11.2. XML 名字空间的名字和前缀

XML 名字空间[XML-NAMES]为所有 XMPP 兼容的 XML 建立数据所有权的严格界限. 这个名字空间的基本功能是把结构上混合在一起的 XML 元素区分出不同的词汇. 确保 XMPP 兼容的 XML 有名字空间的感知能力使得任何允许的 XML 可以被结构化的混合到任何 XMPP 数据元素中. XML 名字空间的名字和前缀的规则在下一小节中.

11.2.1. 流名字空间

在所有的 XML 流头中必须声明一个流名字空间. 流名字空间必须 (MUST) 是 'http://etherx.jabber.org/streams'. 元素名<stream/>和它的<features/>和<error/>子元素必须(MUST)在所有实例中符合这个流名字空间前缀. 一个实现应该 (SHOULD) 只为这些元素生成'stream:'前缀, 并且由于历史原因可以 (MAY) 只接受'stream:'前缀.

11.2.2. 缺省名字空间

在所有的 XML 流中必须声明一个缺省的流名字空间用于定义允许的流根元素的一级子元素. 这个名字空间声明对于初始化流和应答流必须(MUST)是相同的使得双方的流都是合格一致的. 缺省的名字空间声明适用于流和所有在流中发送的节(除非由流名字空间或回拨名字空间的前缀显式的符合另一个名字空间).

一个服务器实现必须（MUST）支持以下两个缺省名字空间(由于历史原因, 一些实现可能(MAY)只支持这两个缺省名字空间):

- jabber:client -- 当流用于客户端和服务器的通信时声明这个缺省名字空间
- jabber:server -- 当流用于两个服务器间的通信时声明这个缺省名字空间

一个客户端实现必须(MUST)支持'jabber:client'缺省名字空间,并且由于历史原因可以（MAY）只支持这个缺省名字空间.

如果缺省名字空间是'jabber:client'或'jabber:server'，一个实施不能(MUST NOT）在这个名字空间下为元素生成名字空间前缀. 一个实现不应该（SHOULD NOT）按照元素的内容（可能和流相反）生成不同于'jabber:client'和'jabber:server'名字空间前缀.

注意: 'jabber:client' 和 'jabber:server' 名字空间接近于相同但是用于不同的上下文(客户端服务器通信用'jabber:client' 而服务器间通信用'jabber:server'). 这两者之间仅有的不同在于在'jabber:client'中被发送的节中'to'和'from'属性是可选的(OPTIONAL),而在'jabber:server'中被发送的节中它们是必需的(REQUIRED). 如果兼容的实施接受一个符合'jabber:client'或'jabber:server'名字空间的流, 它必须(MUST)支持通用属性(第九章第一节)和三个核心节类型(message, presence, 和 IQ)的基本语义(第九章第二节).

11.2.3. 回拨名字空间

所有用于服务器回拨的(第八章)元素都必须声明一个回拨名字空间. 回拨名字空间的名字必须(MUST)是'jabber:server:dialback'. 所有符合此名字空间的元素必须（MUST）有前缀. 一个实现应该（SHOULD）只为这些元素生成'db:'前缀并且只可以（MAY）接受'db:'前缀.

11.3. 确认

除了注意'jabber:server'名字空间中关于节中'to'和'from'地址的规定,一个服务器不需要为转发到客户端或其他服务器负责检查 XML 元素;一个实现可以（MAY）选择仅提供有效数据元素但这是可选的（OPTIONAL）(尽管一个实现不能（MUST NOT）接受不规范的 XML). 客户端不应该（SHOULD NOT）滥用发送不符合 schema 的数据的能力, 并且应该（SHOULD）忽略接收到的 XML 流中任何和 schema 不一致的元素或属性值. XML 流和节的有效性确认是可选的(OPTIONAL),并且在这里提到的 schemas 仅用于描述的用途.

11.4. 文本声明的包含

实现应该（SHOULD）在发送一个流头信息之前发送一个文本声明. 应用程序必须（MUST）遵守[XML]中关于环境（那里对文本声明做了规定）的规则.

11.5. 字符编码

实现必须（MUST）支持通用字符集 Universal Character Set (ISO/IEC 10646-1 [UCS2])字符到 UTF-8(RFC 3629 [UTF-8])的转换, 必须符合 RFC 2277 [CHARSET]. 实现不能（MUST NOT）试图使用任何其他的编码.

---

12. 核心的兼容性要求

本章总结了可扩展的消息和出席信息协议中的某些方面，为了实施的兼容性，它们必须（MUST）被服务器和客户端支持，当然协议的其他方面也应该（SHOULD）被支持. 为了兼容的目的, 我们在核心协议(它必须(MUST)被任何服务器或客户端支持, 无论是什么特定的应用)和即时消息协议(仅仅是在核心协议之上的即时消息和出席信息应用必须(MUST)支持它)之间划了一个级别. 在本章中定义了所有服务器和客户端的兼容性要求; 即时消息服务器和客户端的兼容性要求在 XMPP-IM 的相关章节中定义.

12.1. 服务器

除了所有已定义的关于安全, XML 使用, 和国际化的要求之外, 一个服务器还必须(MUST)支持以下核心协议以保证兼容性:

- 在地址中应用[STRINGPREP] 的 [NAMEPREP], Nodeprep (附录 A),和 Resourceprep (附录 B) profiles (包括确保域 ID 是[IDNA]中定义的国际化域名)
- XML 流(第四章), 包括 Use of TLS(第五章), Use of SASL(第六章), 和 Resource Binding (第七章)
- 三个在 stanza semantics(第九章第二节)中已定义的节类型 (即,<message/>, <presence/>, 和<iq/>)的基本语义
- 生成错误的语法及相关的流, TLS, SASL, 和 XML 节的语义

另外, 一个服务器可以(MAY)支持以下核心协议:

- 服务器回拨 (第八章)

12.2. 客户端

一个客户端必须(MUST)支持以下核心协议以满足兼容性:

- XML 流(第四章), 包括 Use of TLS(第五章), Use of SASL(第六章), 和 Resource Binding (第七章)
- 三个在 stanza semantics(第九章第二节)中已定义的节类型 (即,<message/>, <presence/>, 和<iq/>)的基本语义
- 处理(并且, 如果可能, 生成) 错误的语法及相关的流, TLS, SASL, 和 XML 节的语义

另外, 一个客户端应该(SHOULD)支持以下核心协议:

- 地址的生成应用[STRINGPREP] 的 [NAMEPREP], Nodeprep (附录 A),和 Resourceprep (附录 B) profiles

---

13. 国际化事项

XML 流在 Character Encoding(第十一章第五节)中定义为必须(MUST)被编码成 UTF-8. 在 Stream Attributes(第四章第四节)中特别定义了, 一个 XML 流应该 (SHOULD)包含一个'xml:lang'属性,它被认为是通过这个用于人类用户解读的流发送的任何 XML 字符数据的缺省语言. 在 xml:lang (第九章第一节第五小节) 特别定义了, 如果一个 XML 节是用来给人类用户解读,这个节应该(SHOULD)包含一个'xml:lang'属性. 服务器在为连接的实体路由或递送节的时候应该 (SHOULD)应用缺省的'xml:lang'属性, 并且不能(MUST NOT)修改或删除它从其他实体收到的节的'xml:lang'属性.

---

14. 安全性事项

14.1. 高安全性

为了 XMPP 通信的目的(客户端-服务器 和 服务器-服务器), "高安全性"条款谈的是相互验证和完整性检查安全性技术的使用; 特别是, 当使用基于证书的验证来提供高安全性, 应该(SHOULD)建立一个带外的信任链,尽管一个共享签名证书可能允许以前不知道的证书在带内建立信任关系. 参见以下第十四章第二节关于证书确认程序.

实施必须(MUST)支持高安全性. 服务提供者应该(SHOULD)基于本地安全策略使用高安全性.

14.2. 证书确认

当一个 XMPP 点和另一个 XMPP 点安全的地通信, 它必须(MUST)确认对方终端的证书.有三种可能的情况:

情形 #1: 点包含一个终端实体证书，以根证书的证书链中一环出现(见[X509]中的第六章第一节).

情形 #2: 点的证书由一个对方不知道的证书授权.

情形 #3: 点的证书由自己签名.

在情形#1, 确认方必须(MUST)做以下两条之一:

1. 根据[X509]的规则确认对方证书.然后证书应该(SHOULD)被对方接下来在[HTTP-TLS]中描述的规则反向确认预期的身份。但如果"xmpp"是subjectAltName扩展类型，则必须（MUST）使用证书中的显示的身份。如果这两项检查之一失败,用户导向的客户端必须（MUST）通知用户（客户端可以（MAY）给用户机会继续连接）或以一个坏证书的错误终止连接。自动客户端应该（SHOULD）终止连接(以一个坏证书错误)并在适当的日志中记录这个错误。自动客户端可以（MAY）提供一个配置设置成禁止检查,但同时必须（MUST）提供一个激活检查的配置。
2. 点应该（SHOULD）出示证书给一个用户用于批准,包括完整的证书链.点必须（MUST）缓存这个证书(或一些其他不会忘记的表达方式比如一个哈希值).在将来的连接中,点必须（MUST）展示相同的证书并且如果改变了证书必须（MUST）通知用户.

在情形#2 和情形#3, 实现应该（SHOULD）执行上述第二条.

14.3. 客户端-服务器通信

一个兼容的客户端实现必须（MUST）支持 TLS 和 SASL 用于连接到服务器.

用于加密 XML 流的 TLS 协议(在 Use of TLS(第五章)定义)提供可信的机制帮助确保机密性和实体之间数据交换的完整性.

用于验证 XML 流的 SASL 协议(在 Use of SASL(第六章)定义)提供可靠的机制用于确认一个连接到服务器的客户端确实是它自己所声明的那个客户端.

服务器宣称的 DNS 主机名被解析之前，客户端-服务器通信不能（MUST NOT）继续进行。应该首先尝试解析[SRV]记录，其服务名为"xmpp-client"，协议名为"tcp"，整个资源记录类似"_xmpp-client._tcp.example.com."(使用字符"xmpp-client"表示服务 ID 是经过 IANA 注册).如果 SRV 查找失败,退而求其次,将查找一个正规的 IPv4/IPv6 地址记录来决定 IP 地址,使用"xmpp-client"端口5222，这个端口是在 IANA 注册了的。

客户端的 IP 地址和访问方法不能（MUST NOT）被服务器公开, 也不能被被任何原始服务器之外的服务器索取。这帮助保护客户端的服务器避免受到直接攻击（译者注：似乎应该是客户避免受到直接攻击，但原文如此）或被第三方知道它的身份。

14.4. 服务器-服务器通信

一个兼容的服务器实现必须（MUST）支持 TLS 和 SASL，用于域间的通信.因为历史原因,一个兼容的实施也应该（SHOULD）支持服务器回拨(第八章).

因为服务提供者是一个策略问题,对于任何给定域和其他域的通信中，它是可选的（OPTIONAL），服务器之间的通信可以（MAY）被任何特定部署的管理员禁止。如果一个特殊的域允许域间的通信，它应该（SHOULD）允许高安全性。

管理员可能想在服务器间使用 SASL 来通信,以确保双方的验证和保密性(比如在机构的私有网络).兼容实施应该（SHOULD）为这个目的支持 SASL.

服务器宣称的 DNS 主机名被解析之前，服务器-服务器通信不能（MUST NOT）继续进行。应该首先尝试解析[SRV]记录，其服务名为"xmpp-server"，协议名为"tcp"，整个资源记录类似"_xmpp-server._tcp.example.com."(使用字符"xmpp-server"表示服务 ID 是经过 IANA 注册的,注意要用"xmpp-server"取代以前用的"jabber",因为以前的用法不符合[SRV]标准;希望保持向后兼容的实现可以继续查找或应答"jabber"服务 ID).如果 SRV 查找失败,退而求其次，将查找一个正规的 IPv4/IPv6 地址记录来决定 IP 地址，使用"xmpp-server"端口 5269，这个端口是在 IANA 注册了的。

服务器回拨防止域欺骗,从而使得伪造 XML 节更为困难.它和 SASL 和 TLS 不一样,它不是一个用于验证、安全或加密服务器之间的流的机制, 所以只是服务器身份的微弱确认而已。而且除非它使用了 DNSSec [DNSSEC]否则它 容易受到 DNS 中毒攻击,即使 DNS 信息是正确的,如果攻击者劫持了远程域,回拨也不能防止它的攻击.需要健壮的安全性的域应该(SHOULD)使用 TLS 和 SASL.如果服务器间的验证使用了 SASL,回拨就不应该(SHOULD NOT)使用了,因为它是多余的.

14.5. 层的次序

协议中的层的次序必须(MUST)如下堆积:

1. TCP
2. TLS
3. SASL
4. XMPP

这个次序的原理是,[TCP]是基于连接的层,被所有使用,所以处于最上层, [TLS]经常是由操作系统层提供,[SASL]经常由应用程序层提供, XMPP 则是应用程序本身.

14.6. 缺乏绑定到 TLS 的 SASL 通道

SASL 构架不提供一个机制来绑定 SASL 验证到一个提供机密性和完整性保护的安全层。这一通道绑定"channel binding"的缺乏阻碍了 SASL 确认低层安全性所绑定的源和目标终端和 SASL 所验证的结果是否一致。如果终端不一致，低层安全性不能被信任用来保护 SASL 验证的实体之间的数据传输。在这种情况下，一个 SASL 安全层进行握手的时候应该有效的忽略低层安全性的存在。

14.7. 强制实现的技术

最低要求，所有实现必须（MUST）支持以下机制:

对于验证: SASL [DIGEST-MD5] 机制

对于机密性: TLS (使用 TLS_RSA_WITH_3DES_EDE_CBC_SHA 密码)

对于两者: TLS 加 SASL EXTERNAL(使用 TLS_RSA_WITH_3DES_EDE_CBC_SHA 密码支持客户端证书)

14.8. 防火墙

使用 XMPP 通信通常是通过[TCP]连接到 5222 端口(客户端-服务器)或 5269 端口(服务器-服务器)，正如在 IANA 注册的那样(见 IANA Considerations (第十五章))．使用这些广为人知的端口允许管理员很容易的通过一般的防火墙来激活或禁止 XMPP 活动.

14.9. 在 SASL 中使用 base64

客户端和服务器都必须(MUST)确认在 SASL 协商中收到的任何[BASE64]数据．一个实现必须(MUST)拒绝(不是忽略)任何非显式允许 base64 字母的字符串；这有助于预防建立隐蔽通道泄漏信息的行为。一个实现不能（MUST NOT）在非法输入处中断，如果那个('=')被包含在一些和最后的数据字符(如，"=AAA" or "BBBB=CCC")不同的东西里面，必须(MUST)拒绝接下来的任何包含('=')的 base64 字符；这有助于防止对这个实现的缓存溢出攻击和其他攻击。Base 64 编码从外表看隐藏了容易辨认的信息，例如密码，但是不提供任何算法机密性。Base 64 编码必须（MUST）按照 RFC 3548[BASE64]第三章的定义执行。

14.10. Stringprep Profiles

为了处理域 ID,XMPP 使用了[STRINGPREP]中的[NAMEPREP] profile; 和 Nameprep 有关的安全性考虑，参考[NAMEPREP]中的相关章节.

另外, XMPP 定义了两个[STRINGPREP]的 profiles: 用于 node identifiers 的 Nodeprep(附录 A)和用于 resource identifiers 的 Resourceprep(附录 B).

Unicode 和 ISO/IEC 10646 集有许多字符看起来相似. 在许多时候, 安全协议的使用者可能看起来吻合,比如当比较信任的第三方的名字的时候. 因为没有很多上下文的时候不可能映射看起来相似的字符,比如知道所用的字符集, stringprep 不匹配相似字符串,也不因为一些字符串象看起来象别的字符串而禁止它们.

一个节点 ID 可能被作为一个实体的 XMPP 地址的一部分. 一个通常的用途是作为一个即时消息用户的用户名;另一个用途是作为一个多用途聊天室的名字; 很多其他种类的实体可能使用节点 ID 作为他们的地址的一部分。这些服务的安全性可能会受到国际化节点 ID 的不同表达的威胁;例如, 一个用户键入一个单独的国际化的节点 ID 可能访问了另一个用户的帐号信息, 或一个用户可能获得访问一个受限的聊天室或服务的访问权限.

一个资源 ID 可能被作为一个实体的 XMPP 地址的一部分。一个通常的用户是即时消息用户所连接的资源(激活的会话)的名字; 另一个是作为多用户聊天室的某用户的昵称; 许多其他种类的实体可能使用资源 ID 作为他们地址的一部分.这些服务的安全性可能会受到国际化资源 ID 的不同表达的威胁;例如, 一个用户可能尝试以同一个名字初始化多个会话,或一个用户可能发送一个消息给多用户聊天室的一个人但实际上发给了另外一个人.

---

15. IANA 事项

15.1. 用于 TLS 数据的 XML 名字空间名

XMPP 中用于 TLS 相关数据的 URN 子名字空间定义如下. (这个名字空间的名字遵守 The IETF XML Registry [XML-REG]定义的格式.)

URI: urn:ietf:params:xml:ns:xmpp-tls

Specification: RFC 3920

Description: This is the XML namespace name for TLS-related data in the Extensible Messaging and Presence Protocol (XMPP) as defined by RFC 3920.

Registrant Contact: IETF, XMPP Working Group, <xmppwg@jabber.org>

15.2. 用于 SASL 数据的 XML 名字空间名

XMPP 中用于 SASL 相关数据的 URN 子名字空间定义如下. (这个名字空间的名字遵守 The IETF XML Registry [XML-REG]定义的格式.)

URI: urn:ietf:params:xml:ns:xmpp-sasl

Specification: RFC 3920

Description: This is the XML namespace name for SASL-related data in the Extensible Messaging and Presence Protocol (XMPP) as defined by RFC 3920.

Registrant Contact: IETF, XMPP Working Group, <xmppwg@jabber.org>

15.3. 用于流错误的 XML 名字空间名

XMPP 中用于流相关错误的 URN 子名字空间定义如下. (这个名字空间的名字遵守 The IETF XML Registry [XML-REG]定义的格式.)

URI: urn:ietf:params:xml:ns:xmpp-streams

Specification: RFC 3920

Description: This is the XML namespace name for stream-related error data in the Extensible Messaging and Presence Protocol (XMPP) as defined by RFC 3920.

Registrant Contact: IETF, XMPP Working Group, <xmppwg@jabber.org>

15.4. 用于资源绑定的 XML 名字空间名

XMPP 中用于资源绑定的 URN 子名字空间定义如下. (这个名字空间的名字遵守 The IETF XML Registry [XML-REG]定义的格式.)

URI: urn:ietf:params:xml:ns:xmpp-bind

Specification: RFC 3920

Description: This is the XML namespace name for resource binding in the Extensible Messaging and Presence Protocol (XMPP) as defined by RFC 3920.

Registrant Contact: IETF, XMPP Working Group, <xmppwg@jabber.org>

15.5. 用于节错误的 XML 名字空间名

XMPP 中用于节相关错误数据的 URN 子名字空间定义如下. (这个名字空间的名字遵守 The IETF XML Registry [XML-REG]定义的格式.)

URI: urn:ietf:params:xml:ns:xmpp-stanzas

Specification: RFC 3920

Description: This is the XML namespace name for stanza-related error data in the Extensible Messaging and Presence Protocol (XMPP) as defined by RFC 3920.

Registrant Contact: IETF, XMPP Working Group, <xmppwg@jabber.org>

15.6. Nodeprep Profile of Stringprep

The Nodeprep profile of stringprep 是在 Nodeprep(附录 A)定义的. IANA 已经在 stringprep profile registry 中注册了 Nodeprep.

Name of this profile:

Nodeprep

RFC in which the profile is defined:

RFC 3920

Indicator whether or not this is the newest version of the profile:

This is the first version of Nodeprep

15.7. Resourceprep Profile of Stringprep

The Resourceprep profile of stringprep 是在 Resourceprep(附录 B)定义的. IANA 已经在 stringprep profile registry 中注册了 Resourceprep.

Name of this profile:

Resourceprep

RFC in which the profile is defined:

RFC 3920

Indicator whether or not this is the newest version of the profile:

This is the first version of Resourceprep

15.8. GSSAPI 服务名

IANA 已经注册了 "xmpp" 作为一个 GSSAPI [GSS-API] 服务名, 在 SASL Definition (第六章第三节)定义了.

15.9. 端口号

IANA 已经注册了"xmpp-client" 和 "xmpp-server" 作为[TCP]端口号 5222 和 5269 的关键字.

这些端口应该（SHOULD）用于客户端-服务器 和 服务器-服务器通信,但它们的使用是可选的（OPTIONAL）.

---

16. 参考

16.1. 标准化参考

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

[CHARSET] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[DIGEST-MD5] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

[DNS] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[GSS-API] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[HTTP-TLS] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[IDNA] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[IPv6] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[LANGTAGS] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[NAMEPREP] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[RANDOM] Eastlake 3rd, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[SASL] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[SRV] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[STRINGPREP] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[TERMS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[UCS2] International Organization for Standardization, "Information Technology - Universal Multiple-octet coded Character Set (UCS) - Amendment 2: UCS Transformation Format 8 (UTF-8)", ISO Standard 10646-1 Addendum 2, October 1996.

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[X509] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML) 1.0 (2nd ed)", W3C REC-xml, October 2000, <http://www.w3.org/TR/REC-xml>.

[XML-NAMES] Bray, T., Hollander, D., and A. Layman, "Namespaces in XML", W3C REC-xml-names, January 1999, <http://www.w3.org/TR/REC-xml-names>.

16.2. 信息参考

[ACAP] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[ASN.1] CCITT, "Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1)", 1988.

[DNSSEC] Eastlake 3rd, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[HTTP] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[IMAP] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[IMP-REQS] Day, M., Aggarwal, S., Mohr, G., and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[IRC] Oikarinen, J. and D. Reed, "Internet Relay Chat Protocol", RFC 1459, May 1993.

[JEP-0029] Kaes, C., "Definition of Jabber Identifiers (JIDs)", JSF JEP 0029, October 2003.

[JEP-0078] Saint-Andre, P., "Non-SASL Authentication", JSF JEP 0078, July 2004.

[JEP-0086] Norris, R. and P. Saint-Andre, "Error Condition Mappings", JSF JEP 0086, February 2004.

[JSF] Jabber Software Foundation, "Jabber Software Foundation", <http://www.jabber.org/>.

[POP3] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[SIMPLE] SIMPLE Working Group, "SIMPLE WG", <http://www.ietf.org/html.charters/simple-charter.html>.

[SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[USINGTLS] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.

[XML-REG] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

XMPP-IM Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 3921, October 2004.

附录 A. Nodeprep

A.1. 介绍

这个附录定义了 "Nodeprep" profile of [STRINGPREP]. 它定义了处理规则让用户能够在 XMPP 中输入国际化节点标识符并尽可能正确的获取正确的字符内容.(一个 XMPP 节点标识符是 XMPP 地址的可选部分,它在域名标识符和那个'@'分隔符之前; 它经常但不是专门用来关联一个即时消息用户名)这些处理规则仅仅适用于 XMPP 节点标识符但不适用于任意文本或任何 XMPP 的其他方面.

这个脚本定义了以下这些, 正如 [STRINGPREP]要求的:

- 脚本预期的适用性: XMPP 的国际化节点标识符
- 用于 stringprep 的输入输出字符集: Unicode 3.2, 定义在本附录的第二章
- 使用的映射: 定义在第三章
- Unicode 正规化使用: 定义在第四章
- 禁止输出的字符串: 定义在第五章
- 双字节字符处理: 定义在第六章

A.2. 字符集

本脚本使用 Unicode 3.2 的未分配编码列表, 指向 Table A.1, 也定义在[STRINGPREP]的附录 A 中.

A.3. 映射

本脚本指定使用[STRINGPREP]的以下表:

Table B.1 Table B.2

A.4. 正规化

本脚本指定 Unicode 正规化使用 form KC, 定义在 [STRINGPREP]中.

A.5. 禁止输出

本脚本指定使用以下的[STRINGPREP]表禁止输出.

Table C.1.1

Table C.1.2

Table C.2.1

Table C.2.2

Table C.3

Table C.4

Table C.5

Table C.6

Table C.7

Table C.8

Table C.9

另外, 以下 Unicode 字符也被禁止:

#x22 (")

#x26 (&)

#x27 (')

#x2F (/)

#x3A (:)

#x3C (<)

#x3E (>)

#x40 (@)

A.6. 双字节

本脚本指定按照[STRINGPREP]第六章检查双字节.

附录 B. Resourceprep

B.1. 介绍

这个附录定义了 "Resourceprep" profile of [STRINGPREP]. 它定义了处理规则让用户能够在 XMPP 中输入国际化资源标识符并尽可能正确的获取正确的字符内容. (一个 XMPP 资源标识符是 XMPP 地址的可选部分，它在域名标识符和'/@'分隔符之后；它经常但不是专门用来关联一个即时消息会话名)这些处理规则仅仅适用于 XMPP 资源标识符但不适用于任意文本或任何 XMPP 的其他方面.

这个脚本定义了以下这些, 正如 [STRINGPREP]要求的:

- 脚本预期的适用性: XMPP 的国际化节点标识符
- 用于 stringprep 的输入输出字符集: Unicode 3.2, 定义在本附录的第二章
- 使用的映射: 定义在第三章
- Unicode 正规化使用: 定义在第四章
- 禁止输出的字符串: 定义在第五章
- 双字节字符处理: 定义在第六章

B.2. 字符集

本脚本使用 Unicode 3.2 的未分配编码列表，指向 Table A.1, 也定义在[STRINGPREP]的附录

B.3. 映射

本脚本指定使用[STRINGPREP]的以下表:

Table B.1

B.4. 正规化

本脚本指定使用 Unicode normalization form KC, 定义在 [STRINGPREP]中.

B.5. 禁止输出

本脚本指定使用以下的[STRINGPREP]表禁止输出.

Table C.1.2 Table C.2.1 Table C.2.2 Table C.3 Table C.4 Table C.5 Table C.6 Table C.7 Table C.8 Table C.9

B.6. 双字节

本脚本指定按照[STRINGPREP]第六章检查双字节.

附录 C. XML 规划

以下 XML schemas 是描述性的, 不是标准的. 'jabber:client' 和 'jabber:server' 名字空间的标准定义, 参照 XMPP-IM.

C.1. Streams namespace

```
<?xml version='1.0' encoding='UTF-8'?>

  <xs:schema

      xmlns:xs='http://www.w3.org/2001/XMLSchema'

      targetNamespace='http://etherx.jabber.org/streams'

      xmlns='http://etherx.jabber.org/streams'

      elementFormDefault='unqualified'>

    <xs:element name='stream'>

      <xs:complexType>

        <xs:sequence xmlns:client='jabber:client'

                     xmlns:server='jabber:server'

                     xmlns:db='jabber:server:dialback'>

          <xs:element ref='features' minOccurs='0' maxOccurs='1'/>
```

```
  <xs:any namespace='urn:ietf:params:xml:ns:xmpp-tls'
          minOccurs='0'
          maxOccurs='unbounded'/>
  <xs:any namespace='urn:ietf:params:xml:ns:xmpp-sasl'
          minOccurs='0'
          maxOccurs='unbounded'/>
  <xs:choice minOccurs='0' maxOccurs='1'>
    <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='client:message'/>
      <xs:element ref='client:presence'/>
      <xs:element ref='client:iq'/>
    </xs:choice>
    <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='server:message'/>
      <xs:element ref='server:presence'/>
      <xs:element ref='server:iq'/>
      <xs:element ref='db:result'/>
      <xs:element ref='db:verify'/>
    </xs:choice>
  </xs:choice>
  <xs:element ref='error' minOccurs='0' maxOccurs='1'/>
</xs:sequence>
<xs:attribute name='from' type='xs:string' use='optional'/>
<xs:attribute name='id' type='xs:NMTOKEN' use='optional'/>
```

```
        <xs:attribute name='to' type='xs:string' use='optional'/>

        <xs:attribute name='version' type='xs:decimal'
use='optional'/>

        <xs:attribute ref='xml:lang' use='optional'/>

      </xs:complexType>

    </xs:element>

    <xs:element name='features'>

      <xs:complexType>

        <xs:all xmlns:tls='urn:ietf:params:xml:ns:xmpp-tls'

                xmlns:sasl='urn:ietf:params:xml:ns:xmpp-sasl'

                xmlns:bind='urn:ietf:params:xml:ns:xmpp-bind'

                xmlns:sess='urn:ietf:params:xml:ns:xmpp-session'>

          <xs:element ref='tls:starttls' minOccurs='0'/>

          <xs:element ref='sasl:mechanisms' minOccurs='0'/>

          <xs:element ref='bind:bind' minOccurs='0'/>

          <xs:element ref='sess:session' minOccurs='0'/>

        </xs:all>

      </xs:complexType>

    </xs:element>

    <xs:element name='error'>

      <xs:complexType>

        <xs:sequence
xmlns:err='urn:ietf:params:xml:ns:xmpp-streams'>

          <xs:group   ref='err:streamErrorGroup'/>
```

```
        <xs:element ref='err:text'
                    minOccurs='0'
                    maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

C.2. Stream error namespace

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-streams'
    xmlns='urn:ietf:params:xml:ns:xmpp-streams'
    elementFormDefault='qualified'>

  <xs:element name='bad-format' type='empty'/>

  <xs:element name='bad-namespace-prefix' type='empty'/>

  <xs:element name='conflict' type='empty'/>

  <xs:element name='connection-timeout' type='empty'/>

  <xs:element name='host-gone' type='empty'/>

  <xs:element name='host-unknown' type='empty'/>

  <xs:element name='improper-addressing' type='empty'/>

  <xs:element name='internal-server-error' type='empty'/>

  <xs:element name='invalid-from' type='empty'/>
```

```
<xs:element name='invalid-id' type='empty'/>
<xs:element name='invalid-namespace' type='empty'/>
<xs:element name='invalid-xml' type='empty'/>
<xs:element name='not-authorized' type='empty'/>
<xs:element name='policy-violation' type='empty'/>
<xs:element name='remote-connection-failed' type='empty'/>
<xs:element name='resource-constraint' type='empty'/>
<xs:element name='restricted-xml' type='empty'/>
<xs:element name='see-other-host' type='xs:string'/>
<xs:element name='system-shutdown' type='empty'/>
<xs:element name='undefined-condition' type='empty'/>
<xs:element name='unsupported-encoding' type='empty'/>
<xs:element name='unsupported-stanza-type' type='empty'/>
<xs:element name='unsupported-version' type='empty'/>
<xs:element name='xml-not-well-formed' type='empty'/>
<xs:group name='streamErrorGroup'>
  <xs:choice>
    <xs:element ref='bad-format'/>
    <xs:element ref='bad-namespace-prefix'/>
    <xs:element ref='conflict'/>
    <xs:element ref='connection-timeout'/>
    <xs:element ref='host-gone'/>
    <xs:element ref='host-unknown'/>
    <xs:element ref='improper-addressing'/>
```

```
      <xs:element ref='internal-server-error'/>
      <xs:element ref='invalid-from'/>
      <xs:element ref='invalid-id'/>
      <xs:element ref='invalid-namespace'/>
      <xs:element ref='invalid-xml'/>
      <xs:element ref='not-authorized'/>
      <xs:element ref='policy-violation'/>
      <xs:element ref='remote-connection-failed'/>
      <xs:element ref='resource-constraint'/>
      <xs:element ref='restricted-xml'/>
      <xs:element ref='see-other-host'/>
      <xs:element ref='system-shutdown'/>
      <xs:element ref='undefined-condition'/>
      <xs:element ref='unsupported-encoding'/>
      <xs:element ref='unsupported-stanza-type'/>
      <xs:element ref='unsupported-version'/>
      <xs:element ref='xml-not-well-formed'/>
    </xs:choice>
  </xs:group>
  <xs:element name='text'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute ref='xml:lang' use='optional'/>
```

```
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

C.3. TLS namespace

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-tls'
    xmlns='urn:ietf:params:xml:ns:xmpp-tls'
    elementFormDefault='qualified'>
  <xs:element name='starttls'>
    <xs:complexType>
      <xs:sequence>
        <xs:element
            name='required'
            minOccurs='0'
```

```
            maxOccurs='1'
            type='empty'/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name='proceed' type='empty'/>
    <xs:element name='failure' type='empty'/>
    <xs:simpleType name='empty'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value=''/>
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>
```

C.4. SASL namespace

```
<?xml version='1.0' encoding='UTF-8'?>
  <xs:schema
      xmlns:xs='http://www.w3.org/2001/XMLSchema'
      targetNamespace='urn:ietf:params:xml:ns:xmpp-sasl'
      xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
      elementFormDefault='qualified'>
    <xs:element name='mechanisms'>
      <xs:complexType>
        <xs:sequence>
```

```
      <xs:element name='mechanism'
                  maxOccurs='unbounded'
                  type='xs:string'/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name='auth'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='empty'>
        <xs:attribute name='mechanism'
                      type='xs:string'
                      use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name='challenge' type='xs:string'/>
<xs:element name='response' type='xs:string'/>
<xs:element name='abort' type='empty'/>
<xs:element name='success' type='empty'/>
<xs:element name='failure'>
  <xs:complexType>
    <xs:choice minOccurs='0'>
```

```
            <xs:element name='aborted' type='empty'/>
            <xs:element name='incorrect-encoding' type='empty'/>
            <xs:element name='invalid-authzid' type='empty'/>
            <xs:element name='invalid-mechanism' type='empty'/>
            <xs:element name='mechanism-too-weak' type='empty'/>
            <xs:element name='not-authorized' type='empty'/>
            <xs:element name='temporary-auth-failure' type='empty'/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <xs:simpleType name='empty'>
        <xs:restriction base='xs:string'>
          <xs:enumeration value=''/>
        </xs:restriction>
      </xs:simpleType>
    </xs:schema>
```

C.5. Resource binding namespace

```
<?xml version='1.0' encoding='UTF-8'?>

    <xs:schema
        xmlns:xs='http://www.w3.org/2001/XMLSchema'
        targetNamespace='urn:ietf:params:xml:ns:xmpp-bind'
        xmlns='urn:ietf:params:xml:ns:xmpp-bind'
        elementFormDefault='qualified'>
```

```
    <xs:element name='bind'>
      <xs:complexType>
        <xs:choice minOccurs='0' maxOccurs='1'>
          <xs:element name='resource' type='xs:string'/>
          <xs:element name='jid' type='xs:string'/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

C.6. Dialback namespace

```
<?xml version='1.0' encoding='UTF-8'?>
  <xs:schema
      xmlns:xs='http://www.w3.org/2001/XMLSchema'
      targetNamespace='jabber:server:dialback'
      xmlns='jabber:server:dialback'
      elementFormDefault='qualified'>
    <xs:element name='result'>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base='xs:token'>
            <xs:attribute name='from' type='xs:string'
use='required'/>
            <xs:attribute name='to' type='xs:string' use='required'/>
            <xs:attribute name='type' use='optional'>
```

```
            <xs:simpleType>
              <xs:restriction base='xs:NCName'>
                <xs:enumeration value='invalid'/>
                <xs:enumeration value='valid'/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name='verify'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:token'>
          <xs:attribute name='from' type='xs:string'
use='required'/>
          <xs:attribute name='id' type='xs:NMTOKEN'
use='required'/>
          <xs:attribute name='to' type='xs:string' use='required'/>
          <xs:attribute name='type' use='optional'>
            <xs:simpleType>
              <xs:restriction base='xs:NCName'>
                <xs:enumeration value='invalid'/>
```

```
                    <xs:enumeration value='valid'/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:schema>
```

C.7. Stanza error namespace

```
<?xml version='1.0' encoding='UTF-8'?>

    <xs:schema

        xmlns:xs='http://www.w3.org/2001/XMLSchema'

        targetNamespace='urn:ietf:params:xml:ns:xmpp-stanzas'

        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'

        elementFormDefault='qualified'>

      <xs:element name='bad-request' type='empty'/>

      <xs:element name='conflict' type='empty'/>

      <xs:element name='feature-not-implemented' type='empty'/>

      <xs:element name='forbidden' type='empty'/>

      <xs:element name='gone' type='xs:string'/>

      <xs:element name='internal-server-error' type='empty'/>

      <xs:element name='item-not-found' type='empty'/>
```

```
<xs:element name='jid-malformed' type='empty'/>
<xs:element name='not-acceptable' type='empty'/>
<xs:element name='not-allowed' type='empty'/>
<xs:element name='payment-required' type='empty'/>
<xs:element name='recipient-unavailable' type='empty'/>
<xs:element name='redirect' type='xs:string'/>
<xs:element name='registration-required' type='empty'/>
<xs:element name='remote-server-not-found' type='empty'/>
<xs:element name='remote-server-timeout' type='empty'/>
<xs:element name='resource-constraint' type='empty'/>
<xs:element name='service-unavailable' type='empty'/>
<xs:element name='subscription-required' type='empty'/>
<xs:element name='undefined-condition' type='empty'/>
<xs:element name='unexpected-request' type='empty'/>
<xs:group name='stanzaErrorGroup'>
  <xs:choice>
    <xs:element ref='bad-request'/>
    <xs:element ref='conflict'/>
    <xs:element ref='feature-not-implemented'/>
    <xs:element ref='forbidden'/>
    <xs:element ref='gone'/>
    <xs:element ref='internal-server-error'/>
    <xs:element ref='item-not-found'/>
    <xs:element ref='jid-malformed'/>
```

```
    <xs:element ref='not-acceptable'/>
    <xs:element ref='not-allowed'/>
    <xs:element ref='payment-required'/>
    <xs:element ref='recipient-unavailable'/>
    <xs:element ref='redirect'/>
    <xs:element ref='registration-required'/>
    <xs:element ref='remote-server-not-found'/>
    <xs:element ref='remote-server-timeout'/>
    <xs:element ref='resource-constraint'/>
    <xs:element ref='service-unavailable'/>
    <xs:element ref='subscription-required'/>
    <xs:element ref='undefined-condition'/>
    <xs:element ref='unexpected-request'/>
  </xs:choice>
</xs:group>
<xs:element name='text'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:string'>
        <xs:attribute ref='xml:lang' use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

```
    <xs:simpleType name='empty'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value=''/>
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>
```

附录 D. 核心 Jabber 协议和 XMPP 的不同

本章是非标准的.

译者注：附录 D 对于新接触 XMPP 的人没有什么意义，就不翻译了，免得浪费时间。因为现在 RFC 公布已经很久了，以前的 Jabber 实现很多都进化到 XMPP 了。

XMPP has been adapted from the protocols originally developed in the Jabber open-source community, which can be thought of as "XMPP 0.9". Because there exists a large installed base of Jabber implementations and deployments, it may be helpful to specify the key differences between the relevant Jabber protocols and XMPP in order to expedite and encourage upgrades of those implementations and deployments to XMPP. This section summarizes the core differences, while the corresponding section of XMPP-IM summarizes the differences that relate specifically to instant messaging and presence applications.

D.1. Channel Encryption

It was common practice in the Jabber community to use SSL for channel encryption on ports other than 5222 and 5269 (the convention is to use ports 5223 and 5270). XMPP uses TLS over the IANA-registered ports for channel encryption, as defined under Use of TLS (Section 5) herein.

D.2. Authentication

The client-server authentication protocol developed in the Jabber community used a basic IQ interaction qualified by the 'jabber:iq:auth' namespace (documentation of this protocol is contained in [JEP-0078], published by the Jabber Software Foundation [JSF]). XMPP uses SASL for authentication, as defined under Use of SASL (Section 6) herein.

The Jabber community did not develop an authentication protocol for server-to-server communications, only the Server Dialback (Section 8) protocol to prevent server spoofing. XMPP supersedes Server Dialback with a true server-to-server authentication protocol, as defined under Use of SASL (Section 6) herein.

D.3. Resource Binding

Resource binding in the Jabber community was handled via the 'jabber:iq:auth' namespace (which was also used for client authentication with a server). XMPP defines a dedicated namespace for resource binding as well as the ability for a server to generate a resource identifier on behalf of a client, as defined under Resource Binding (Section 7).

D.4. JID Processing

JID processing was somewhat loosely defined by the Jabber community (documentation of forbidden characters and case handling is contained in [JEP-0029], published by the Jabber Software Foundation [JSF]). XMPP specifies the use of [NAMEPREP] for domain identifiers and supplements Nameprep with two additional [STRINGPREP] profiles for JID processing: Nodeprep (Appendix A) for node identifiers and Resourceprep (Appendix B) for resource identifiers.

D.5. Error Handling

Stream-related errors were handled in the Jabber community via XML character data text in a <stream:error/> element. In XMPP, stream-related errors are handled via an extensible mechanism defined under Stream Errors (Section 4.7) herein.

Stanza-related errors were handled in the Jabber community via HTTP-style error codes. In XMPP, stanza-related errors are handled via an extensible mechanism defined under Stanza Errors (Section 9.3) herein. (Documentation of a mapping between Jabber and XMPP error handling mechanisms is contained in [JEP-0086], published by the Jabber Software Foundation [JSF].)

D.6. Internationalization

Although use of UTF-8 has always been standard practice within the Jabber community, the community did not define mechanisms for specifying the language of human-readable text provided in XML character data. XMPP specifies the use of the 'xml:lang' attribute in such contexts, as defined under Stream Attributes (Section 4.4) and xml:lang (Section 9.1.5) herein.

D.7. Stream Version Attribute

The Jabber community did not include a 'version' attribute in stream headers. XMPP specifies inclusion of that attribute as a way to signal support for the stream features (authentication, encryption, etc.) defined under Version Support (Section 4.4.1) herein.

贡献者

XMPP 的大部分核心方面是由 1999 年开始的 Jabber 开源社区开发的. 这个社区是由 Jeremie Miller 建立的, 他于 1999 年 1 月发布了最初版的 jabber server 源代码. 主要的基础协议的早期贡献者还包括 Ryan Eatmon, Peter Millard, Thomas Muldowney,和 Dave Smith. XMPP 工作组的工作主要集中在安全性和国际化方面; 在这些领域, 使用 TLS 和 SASL 的协议最初是由 Rob Norris 贡献的, stringprep profiles 最初是由 Joe Hildebrand 贡献的. The error code syntax 是由 Lisa Dusseault 建议的.

致谢

感谢许多在贡献者名单之外的人们. 尽管很难提供一个完整的名单, 以下个人对于定义协议或评论标准提供了很多帮助: Thomas Charron, Richard Dobson, Sam Hartman, Schuyler Heath, Jonathan Hogg, Cullen Jennings, Craig Kaes, Jacek Konieczny, Alexey Melnikov, Keith Minkler, Julian Missig, Pete Resnick, Marshall Rose, Alexey Shchepin, Jean-Louis Seguineau, Iain Shigeoka, Greg Troxel, and David Waite. 也感谢 XMPP 工作组的成员和 IETF 社区在本文的成文过程中一直提供的评论和反馈。

作者地址

Peter Saint-Andre (editor) Jabber Software Foundation

EMail: stpeter@jabber.org

Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the IETF's procedures with respect to rights in IETF Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf- ipr@ietf.org.

感谢

目前为 RFC 编辑活动提供资金的 Internet Society.