RFC6120

来自Jabber/XMPP中文翻译计划

"本文的英文原文来自RFC 6120 (http://xmpp.org/rfcs/rfc6120.html)

互联网工程任务组(IETF)	P. Saint-Andre
申请讨论: 6120	Cisco
取代: 3920	2011年3月
类别: 标准跟踪	
ISSN: 2070-1721	

可扩展的消息和出席信息协议 (XMPP): 核心协议

摘要

可扩展的消息和出席信息协议(XMPP)是一个XML应用，它让任意两个或多个网络实体可以进行结构化和可扩展的准实时信息交流. 本文定义了XMPP的核心方法: XML流的配置和拆除, 通道加密, 验证, 错误处理, 以及基本的消息通讯, 网络可用性 ("presence"), 和 请求-应答 交互模式. 本文取代了 RFC 3920.

本文的状态

这是一个互联网标准跟踪文档.

本文是互联网工程工作组(IETF)的一个成果. 它代表了IETF社区的一致意见. 它已经公开审核并由互联网工程控制组(IESG)批准发布了. 更多关于互联网标准的信息请参见RFC 5741第2章.

关于本文当前状态的信息, 任何错误, 以及如何对它提出反馈，请到 http://www.rfc-editor.org/info/rfc6120 .

版权声明

目录

序论

概述

可扩展的消息和出席信息协议(XMPP)是一个可扩展标记语言XML应用，让任何两个或多个网络实体之间进行结构化和可扩展的准实时信息交流. 本文定义了XMPP的核心协议方法: XML流的配置和解除, 通道加密, 验证, 错误处理, 以及消息通讯基础, 网络可用性 ("presence"), 和 请求–应答 交互.

历史

XMPP的基本语法和语义最开始是由Jabber开源社区开发的, 主要是在1999年. 2002年, 根据IMP-REQS ， XMPP工作组被允许基于Jabber协议开发一个适合IETF的即时消息和出席信息技术. 到了2004年10月, 发布了 RFC3920 和 RFC3921 , 意味着那时候XMPP的主要定义完成了.

从2004年开始，互联网社区已经获得了广泛的XMPP实现和布署经验, 包括XMPP标准基金会(XSF)主持下开展的正式的互操作性测试. 本文全面整合了从软件开发者和XMPP服务提供者得到的反馈, 包含了一系列向后兼容的修改，见 附录D . 结果是, 本文反映了互联网社区对于XMPP1.0核心功能的初步共识, 因此废止了RFC 3920.

功能汇总

这个非正规的章节提供了一个方便开发者的XMPP功能汇总; 接下来的其他章节则是XMPP的规范定义.

XMPP的目标是允许两个(或多个)实体通过网络来交换相关的小件结构化数据(所谓"XML节"). XMPP典型地使用分布式的 客户端–服务器 体系结构来实现, 这里客户端需要连接到一个服务器以获得对网络的访问，从而被允许和其他实体(可能在其他服务器上)交换XML节. 一个客户端连接到一个服务器，交换XML节，以及结束连接，这样的流程如下:

1. 确定要连接的IP地址和端口号, 典型的做法是对一个合格的域名做出解析(3.2)
2. 打开一个传输控制协议 TCP 连接
3. 通过TCP打开一个XML流 4.2
4. 握手最好使用传输层安全性 TLS 来进行通道加密(5)
5. 使用简单验证和安全层 SASL 机制来验证 (6)
6. 绑定一个资源到这个流上 (7)
7. 和其他网络上的实体交换不限数量的XML节(8)
8. 关闭XML流 (4.4)
9. 关闭TCP连接

在XMPP中, 一个服务器可以选择性地连接到另一个服务器以激活域间或服务器间的通讯. 这种情形下, 两个服务器需要在他们自身之间建立一个连接然后交换XML节; 这个过程所做的事情如下:

1. 确定要连接的IP地址和端口号, 典型的做法是对一个合格的域名做出解析(3.2)
2. 打开一个TCP连接
3. 打开一个XML流 4.2
4. 握手最好使用TLS来进行通道加密(5)
5. 使用简单验证和安全层 SASL 机制来验证 (6) *
6. 交换不限数量的XML节，可以服务器之间直接交换，也可以代表每台服务器上的相关实体来交换，例如那些连到服务器上的客户端 (8)
7. 关闭XML流 (4.4)
8. 关闭TCP连接

- 互操作性提示: 在本文写就的时候, 大多数已布署的服务器仍使用服务器回拨协议 XEP-0220 来提供弱身份验证，而不是使用SASL的 PKIX证书来提供强验证, 特别在这些情况下，SASL握手无论如何将不会得到强验证 (例如, 因为TLS握手没有被对方服务器强制要求, 或因为当TLS握手时对方服务器提供的PKIX证书是自签名的并且之前没有被接受过); 细节请见 XEP-0220 . 本文的解决方案显然提供了一个更高级别的安全性 (参见 13.6).

本文指定了客户端如何连接到服务器以及基本的XML节语义. 然而, 本文不定义一个连接成功建立之后可能用来交换的XML节的"载荷"; 反之, 那些载荷被定义在各种XMPP扩展之中. 例如, XMPP-IM 定义了基本的即时消息和出席信息功能的扩展. 另外, XSF创造了各种扩展协议，即XEP系列 XEP-0001 ,也为广泛的应用程序定义了扩展.

术语

本文中的关键字 "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", 和 "OPTIONAL" 的解释参见RFC 2119 关键字 .

特定的安全相关的术语的含义参见 安全术语 ; 这些术语包括但不限于, "assurance", "attack", "authentication", "authorization", "certificate", "certification authority", "certification path", "confidentiality", "credential", "downgrade", "encryption", "hash value", "identity", "integrity", "signature", "self-signed certificate", "sign", "spoof", "tamper", "trust", "trust anchor", "validate", and "verify".

特定的和证书，域名，应用服务身份相关的术语参见 TLS-证书 ; 这包括但不限于, "PKIX certificate", "source domain", "derived domain", 以及身份类型 "CN-ID", "DNS-ID", 和 "SRV-ID".

其他安全相关的术语定义于参考协议中 (例如, "denial of service" (拒绝服务)定义于 DOS 或 "end entity certificate" (终端实体证书)定义于 PKIX).

术语 "whitespace" (空格) 用于指代 XML 中任何匹配"S"的字符或字符串, 也就是说, 一个或多个满足 ABNF 定义的SP, HTAB, CR, 或 LF 规则的实例.

术语 "localpart" (本地部分), "domainpart" (域部分), 以及 "resourcepart" (资源部分)定义于 XMPP地址 .

术语 "bare JID" (纯JID) 指代一个格式为 <localpart@domainpart> (对于一个位于某个服务器上的帐户而言) 或 <domainpart> (对于一个服务器而言) 的XMPP地址.

术语 "full JID" (全JID) 指代一个格式为 <localpart@domainpart/resourcepart> (对一个典型的已授权客户端或和某个帐号相关的设备而言) 或 <domainpart/resourcepart> (对于一个典型的资源或和某个服务器相关的文字)的XMPP地址.

术语 "XML stream" (也称为 "stream" (流)) 定义于 4.1 .

术语 "XML stanza" (也称为 "stanza" (节)) 定义于 4.1 . 有三种 stanzas（节）: message, presence, 和 IQ ("Info/Query"的简称). 这些通讯原语分别定义于 8.2.1 , 8.2.2 , 和 8.2.3 .

术语 "originating entity" (原实体)指的是第一次生成一个发送到XMPP网络的stanza(节)的实体(例如, 一个已连接的客户端, 一个附加的服务, 或一个服务器). 术语 "generated stanza" (生成的节)值的是生成的节那个节.

术语 "input stream" (输入流)指定这样一个XML流，服务器通过这个流从一个已连接的客户端或远端服务器接收数据, 而术语 "output stream" (输出流)指定这样一个流，服务器通过这个流发送数据到一个已连接的客户端或远程服务器. 以下术语指定一些动作，处理从输入流收到的数据时服务器可以执行这些动作:

```
route(路由):
    传递数据到一个远端服务器让它自行处理或最终递送到一个和远端服务器关联的客户端
deliver(递送):
    传递数据到一个已连接的客户端
ignore(忽略):
    丢弃数据不做任何处理或返回一个错误给发送者sender
```

当术语 "ignore" (忽略)用于客户端处理收到的数据时, 短语 "without acting upon it" (不做任何处理)明确的包括不展示任何数据给使用者(人).

接下来的 "XML符号" 被 IRI 用于展示无法用仅用ASCII码呈现的字符, 本文的一些例子使用了类似 "&#x...." 的格式来表现 UNICODE 字符串 (例如, 字符串 "ř" 表示Unicode字符 LATIN SMALL LETTER R WITH CARON); 这个格式在XMPP系统中绝对不会在网络上被发送.

和 URI 展现统一资源定位符的规则一样, XMPP地址文本也是用 '<' 和 '>' 括起来的(尽管基本上它们不属于 URIs).

例如, 被括起来的行是用来提高可读性的, "[...]" 表示省略, 并且还是用了以下预定义字符串 (这些预定义的字符串不会通过网络发送出去):

- C: = 客户端
- E: = 任何XMPP实体
- I: = 发起实体
- P: = 对端服务器
- R: = 接收实体
- S: = 服务器
- S1: = 服务器1
- S2: = 服务器2

读者需要注意这些例子不包括细节, 并且例子里的一些协议流程中, 展示的备用步骤不一定是由前一个步骤发送的确切的数据触发的; 本文或常用参考文档中的协议规范所用到的所有用例里面提供的例子都遵从上述规则. 所有例子都是虚构的并且交换的信息 (例如, 用户名和密码) 不代表任何现存的用户和服务器.

体系结构

XMPP提供一种异步的端到端的结构化数据交换技术，在一个分布式的可全球寻址和出席信息感知的客户端和服务器的网络中使用直接的持久XML流。这种体系结构形式包含了普遍的网络可用性的知识，以及在给定的客户端–服务器和服务器–服务器会话的时候，不限数量的并发信息交易的概念，所以我们把它称为 "并发交易可用性" ("Availability for Concurrent Transactions") (简称ACT) 来把它和来自WWW的 "Representational State Transfer" REST 体系结构形式区别开. 尽管XMPP的体系结构很大程度上类似于 email (参见 EMAIL-ARCH, 它引入了一些变化以便于准实时通讯. ACT体系结构形式的独特特性如下.

全局地址

和email一样, 为了通过网络路由和递送消息,XMPP使用全球唯一地址(基于DNS). 所有XMPP实体可以在网络上被寻址, 大部分客户端和服务器以及很多外部服务可以被客户端和服务器访问. 通常, 服务器地址的格式为 <域部分> (例如, <im.example.com>), 属于某台服务器的帐号的格式为 <本地部分@域部分> (例如, <juliet@im.example.com>, 称为 "纯JID"), 而连接到一个特定的设备或资源并且已经被(服务器)授权可以和外部交互的客户端的格式为 <本地部分@域部分/资源部分> (例如, <juliet@im.example.com/balcony>, 称为 "全JID"). 因为历史原因, XMPP地址常被称为Jabber IDs 或 JIDs. 因为XMPP地址格式的正式规范依赖于国际化技术（本文撰写时正在制定中），这个格式定义于 XMPP-ADDR 而非本文之中. 术语 "localpart"(本地部分), "domainpart"（域部分）, 和 "resourcepart"（资源部分） 正式定义于 XMPP-ADDR .

出席信息

XMPP让一个实体能够向其他实体声明它的网络可用性或者 "presence"（出席信息） . 在XMPP中, 这种可通讯状态是用端到端的专用通讯元素来标识的: 即 <presence/> 节. 尽管网络可用性对于XMPP消息交换并不是必需的, 它还是可以促进实时交互，因为消息发起者可以在发消息之前知道接收者在线并处于可通讯状态. 端到端的出席信息定义于 XMPP-IM .

持久流

每个点对点的一"跳"都建立了基于TCP长连接的持久XML流来保持可通讯状态. 这些 "always-on" 客户端-服务器 和 服务器-服务器 流使得任何时间每方都能够推送数据到另一方并且立即路由和递送. XML流定义于 4 .

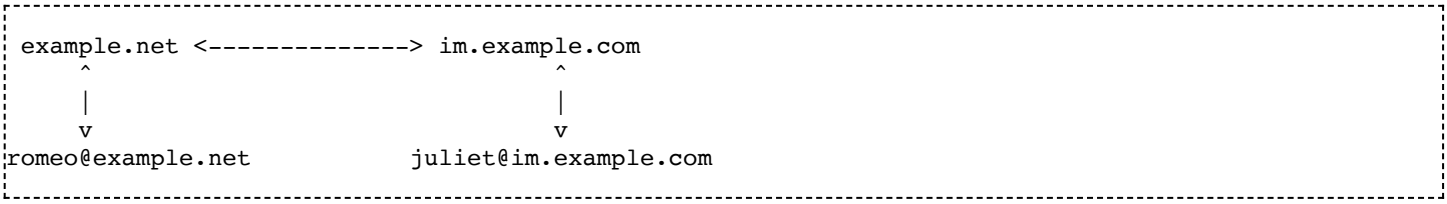
结构化数据

XMPP中基本的协议数据单元不是一个XML流 (它只是为点对点通讯提供传输层) 而是一个 XML 节 ("stanza"), 它是一个通过流发送的XML片段. 一个节的根元素包括路由属性 (类似 "from" 和 "to" 地址), 而节的子元素包含了递送给目标接收者的载荷. XML节定义于 8 .

客户端和服务器的分布式网络

在实践之中, XMPP是一个包含了很多互相通讯的客户端和服务器的网络(当然, 任何两个给定的布署服务器之间的通讯都是严格谨慎的并且也和本地服务策略有关). 因此, 例如, 与服务器 <im.example.com> 关联的用户 <juliet@im.example.com> 能够和服务器 <example.net> 关联的用户 <romeo@example.net> 交换消息，出席信息和其他结构化数据. 这个模式对使用全局地址的消息协议是很常见的, 例如email网络 (见 SMTP 和 EMAIL-ARCH . 结果, 在XMPP中端到端的通讯是逻辑上的点对点，而物理结构则是 客户端-服务器-服务器-客户端, 如下图所示.

图1: 分布式客户端-服务器 体系结构



> 参考文献: 体系结构使用 XML流 和 XML节 ，但是两个客户端之间直接建立端到端的连接则使用基于 LINKLOCAL 的技术, 不过那个体系结构没有定义在本协议之中，它只能说是 "类XMPP"; 详见 XEP-0174 . 另外, XML流可以在任何可靠的传输层上建立端到端的连接, 包括XMPP本身的扩展; 无论如何, 这些方法没有包含在本文之中.

以下段落描述客户端和服务器们各自在网络中负责什么.

一个客户端就是一个实体，它先和它的注册帐号所在服务器建立XML流 (通过 SASL握手) , 然后完成资源绑定 , 这样就能通过建好的流在客户端和服务器之间递送XML节. 客户端使用 XMPP 来和它的服务器, 其他客户端以及任何其他网络上的实体通讯, 这里服务器负责递送节到同一台服务器上其他已连接的客户端，或把它们路由到远程服务器上. 一个服务器上的注册帐号可以同时使用多个客户端连接到一台服务器上, 这里每个客户端的XMPP地址的 *资源部分* 是不同的 (例如, <juliet@im.example.com/balcony> 和 <juliet@im.example.com/chamber>), 定义于 XMPP-ADDR 和 7

一个服务器是一个实体，主要负责以下事项:

- 管理已连接客户端的 XML流 并通过建好的流递送 XML节 到那些通过客户端; 这也包括负责确保客户端在被授权访问XMPP网络之前的客户端身份验证工作.
- 遵循本地服务对服务器之间通讯的策略, 管理和远程服务器之间的 XML流 并通过建好的流路由 XML节 到那些服务器.

取决于服务器的不同, 一个XMPP服务器的次要责任可能包括:

- 存储客户端使用的数据 (例如, 用户的基于 XMPP–IM 的联系人; 在这种情况下, 相关的XML节直接由服务器本身代替客户端来处理而不用路由到远程服务器或递送到一个已连接的客户端.
- 托管的额外服务也使用XMPP作为通讯的基础，但是提供超出本文范围的额外的或 XMPP–IM 定义的功能; 例子包括多用户会议服务(定义于 XEP-0045) 和 发布–订阅 服务 (定义于 XEP-0060).

TCP绑定

范围

如本文定义的XMPP所述, 一个发起方实体在 (客户端或服务器) 和接收方实体协商XML流之前必须 (MUST) 打开一个到接收方实体 (服务器) 的 TCP 连接. 然后在使用XML流期间双方一直保持那个TCP连接. 这个规则在下面章节中的TCP绑定要用到.

> 参考文献: 不一定要把XML流建立在TCP上, 其他传输协议也是可以的. 例如, 两个实体可以通过 HTTP 互相连接（定义于 XEP-0124 和 XEP-0206 . 无论如何, 本协议只定义把XMPP绑定到 TCP.

合格的全域名解析

因为XML流是是通过TCP发送的, 发起方实体在尝试打开一个XML流之前需要确定接收方实体的IPv4或IPv6地址(以及端口). 一般来说这是通过解析接收方实体的合格的全域名(简称FQDN,参见 DNS概念)来实现的.

首选流程:SRV查询

FQDN解析的首选流程是如下使用 DNS-SRV 记录:

1. 发起方实体构造一个 DNS SRV 查询，参数如下:
    1. 一个 "xmpp–client" (用于 客户端–服务器 连接) 或 "xmpp–server" (用于 服务器–服务器 连接)服务
    2. 一个 "tcp" 协议
    3. 一个对应发起方实体希望连接的XMPP服务的 "原有域" （ TLS-CERTS ） 的名字 (例如, "example.net" 或 "im.example.com")
2. 得到一个类似 "_xmpp–client._tcp.example.net." 或 "_xmpp–server._tcp.im.example.com." 的查询.
3. 如果收到应答, 它将包含一个或多个FDQN和端口的组合, 每个都拥有权重和优先级 (如 DNS-SRV 所述). (无论如何, 如果SRV查询的结果是一个单独的资源记录 ".", 即根域名, 那么发起方实体必须(MUST)在这时终止SRV处理，因为根据 DNS-SRV ,这样一个结果意味着那个服务在本域中是不可用的)
4. 发起方实体至少选择返回的FQDNs记录中的一个来解决 (根据 DNS-SRV 规则,对FDQN执行 DNS "A" 或 "AAAA" 查询; 这将返回一个 IPv4 或 IPv6 的地址.
5. 成功解析FDQN(包括SRV查询返回的相应的端口号)之后,发起方实体就使用IP地址来连接接收方实体.
6. 如果发起方实体使用那个IP地址连接失败，而的 "A" 或 "AAAA" 记录查询返回了不止一个IP地址, 那么发起方实体使用那个FDQN的下一个解析好的IP地址作为连接地址.

7. 如果发起方实体用给定的FDQN的所有解析出来的IP地址都无法连接, 那么它重复解析过程并使用基于优先级和权重的SRV查询(定义于 DNS SRV)返回的下一个FQDN来连接.
8. 如果发起方实体接收到它的SRV应答但是无法使用接收到的应答数据来建立一个XMPP连接, 它不应该(SHOULD NOT)尝试下面描述的备用流程(这有助于防止入站和出站连接状态不匹配).
9. 如果发起方实体不能从它的SRV查询接收到应答, 它应该(SHOULD)尝试下一节描述的备用流程.

后备流程

后备流程应该(SHOULD)是一个常规的 "A" 或 "AAAA" 地址记录解析以决定原始域的IPv4或IPv6地址, 而端口则为 "xmpp-client" 端口(5222)用于客户端-服务器连接或 "xmpp-server" 端口(5269)用于服务器-服务器连接 (这些是在IANA注册的缺省端口，14.7 .

如果通过TCP连接不成功, 发起方实体可能尝试找到并使用替代连接方法例如HTTP绑定 (见 XEP-0124 和 XEP-0206 , 它可能使用 DNS-TXT 记录(参见 XEP-0156) 来搜索.

什么时候不用SRV

如果发起方实体已经被显式地配置为一个关联到接收实体的原始域的一个特定的FQDN (以及潜在的端口) (比如,一个特定的原始域 example.net "写死" 到一个配置好的 apps.example.com 的 FQDN), 鼓励初始方实体使用配置好的名字而不是建议的对原始域的SRV解析流程.

附加服务使用SRV记录

很多XMPP服务器以可托管附加服务 (超出本文和 XMPP-IM 定义范围) 这种方式来实现，附加服务的DNS域名典型的形式是主XMPP服务的 "子域名" (例如, conference.example.net 用于 XEP-0045 服务,相关的XMPP主服务为 example.net) 或 底层服务的一级域名的 "子域名" (例如, muc.example.com 用于 XEP-0045 服务，相关的XMPP主服务为 im.example.com). 如果一个和远程XMPP服务关联的实体希望连接到这样一个附加服务上, 它将生成一个适当的XML节，而远程服务器将尝试通过一个SRV查询资源记录类似 "_xmpp-server._tcp.conference.example.net." 或 "_xmpp-server._tcp.muc.example.com." 来解析该附加服务的DNS域名. 所以, 如果一个XMPP服务的管理员希望让远程服务器相关的实体能访问这样的附加服务, 除了用于他们的主XMPP服务的 "_xmpp-server" 记录之外, 他们还需要声明适当的 "_xmpp-server" SRV 记录. 当 SRV 记录不可用的时候, 可使用后备方法 3.2.2 来为附加服务解析域名.

重连

XMPP服务器可能在会向连接的客户端和远端服务器提供TCP连接服务的时候意外掉线. 因为这些连接的数量可能非常大, 实体的重连机制寻求解决重连可能导致的对软件性能的冲剂和网络堵塞. 如果实体选择重连, 它:

- 应该把重连之前等待的秒数设置为0到60之间 (这有助于确保不会所有实体在掉线的同一个时间间隔之后同时尝试重连).
- 如果第一次尝试重连没有成功则随后的尝试重连的时间间隔应该越来越长 (例如, 按照 ETHERNET 描述的 "动态二进制指数后退算法").

建议重连的时候使用TLS会话恢复 TLS-RESUME . 本文未来的某个版本, 或某个独立的协议, 可能会提供更多详细的关于加速重连过程的方法的指南.

可靠性

在XMPP使用常连的TCP连接意味着通过XML流发送的XML节可能不可靠, 因为长连的TCP的各方可能无法及时地了解掉线情况. 在XMPP应用层, 长连接掉线可能导致无法发送节. 尽管本文定义的核心XMPP技术未包括克服这一可靠性缺陷的特性, 有一个XMPP扩展在做这件事 (例如, XEP-0198).

XML流

流基础

两个基本概念，使得XMPP实体之间的小的结构化信息有效载荷能快速地进行异步交换：XML流和XML节。这些术语的定义如下。

XML流的定义:

> XML流是一个容器，用于任何两个实体通过网络进行XML元素的交换. XML流的开始明确表达为一个打开的 "流头" (即, 一个包含了适当树形和命名空间声明的 XML <stream> 标签), 而这个XML流的结尾明确表达为一个关闭的XML </stream> 标签. 在流的生存期间, 发起方实体可以通过这个流发送不限数量的XML元素, 这些元素或用来协商这个流 (例如, 完成TLS协商 或 SASL协商) 或用于 XML节. "发起流" 是从发起方实体 (通常是一个客户端或服务器) 到接收方实体 (通常是一个服务器), 也可视为对应发起方 "连接到" 或 "和.....开启会话" 接收方实体. 发起流允许从发起方实体到接收方实体的单向通讯; 为了让接收方实体能够向发起方实体发送节, 接收方实体必须(MUST) 协商一个相反的流 ("应答流").

XML节的定义:

> XML节是一个XMPP中的基本语义单位. 一个节就是一个第一层元素 (在流的深度=1)，它的元素名是 "message", "presence", 或 "iq" ， 而它的合格命名空间是 'jabber:client' 或 'jabber:server'. 相比之下, 任何其他命名空间限定的第一层元素都不是一个XML节 (stream errors, stream features, TLS相关的元素, SASL相关的元素, 等等.), 由'jabber:client' 或 'jabber:server' 命名空间限定的 <message/>, <presence/>, 或 <iq/> 元素但不在第一层 (例如, 包含在一个扩展元素中的 <message/> 元素 (做报告用的 8.4)也不是一个XML节, 不是命名空间 'jabber:client' 或 'jabber:server'限定的 <message/>, <presence/>, 或 <iq/> 元素也不是一个XML节. 一个XML节典型的包含一个或多个必要的子元素 (以及相关的属性, 元素, 和 XML 字符串数据) 来传达所需的信息, 子元素可以(MAY)使用任何XML命名空间 (见 XML-NAMES 和本协议的 8.4).

有三种节: message, presence, 和 IQ ("Info/Query"的缩写). 这些节类型提供三种不同的通讯原语: 一个 "推送" 机制用于已生成的消息, 一个特定的 "发行–订阅" 机制用于广播网络可用性信息, 和一个 "请求–应答" 机制用于更结构化的数据交换 (类似 HTTP . 更多解释分别位于 8.2.1 , 8.2.2 , 和 8.2.3 .

考虑一个客户端连接到一个服务器的例子. 客户端通过发送一个流头来发起一个XML流到服务器, 最好在前面加上一个XML声明来指定XML版本和支持的字符串编码 (见 11.5 和 11.6). 遵循本地策略和服务设置, 该服务器接着以第二个XML流应答回客户端, 最好再次在前面加上一个XML声明. 一旦客户端完成 SASL协商 和 资源绑定 , 该客户端就能通过这个流来发送不限数量的XML节. 当客户端想要关闭这个流的时候, 它只要简单的发送一个关闭 </stream> 标签给服务器，如 4.4 .

于是, 从本质上讲, 一个XML流作为会话期间发送的XML节的信封, 而另一个XML流作为会话期间接收的XML节的信封. 我们可以用如下的简化模型做一个展示.

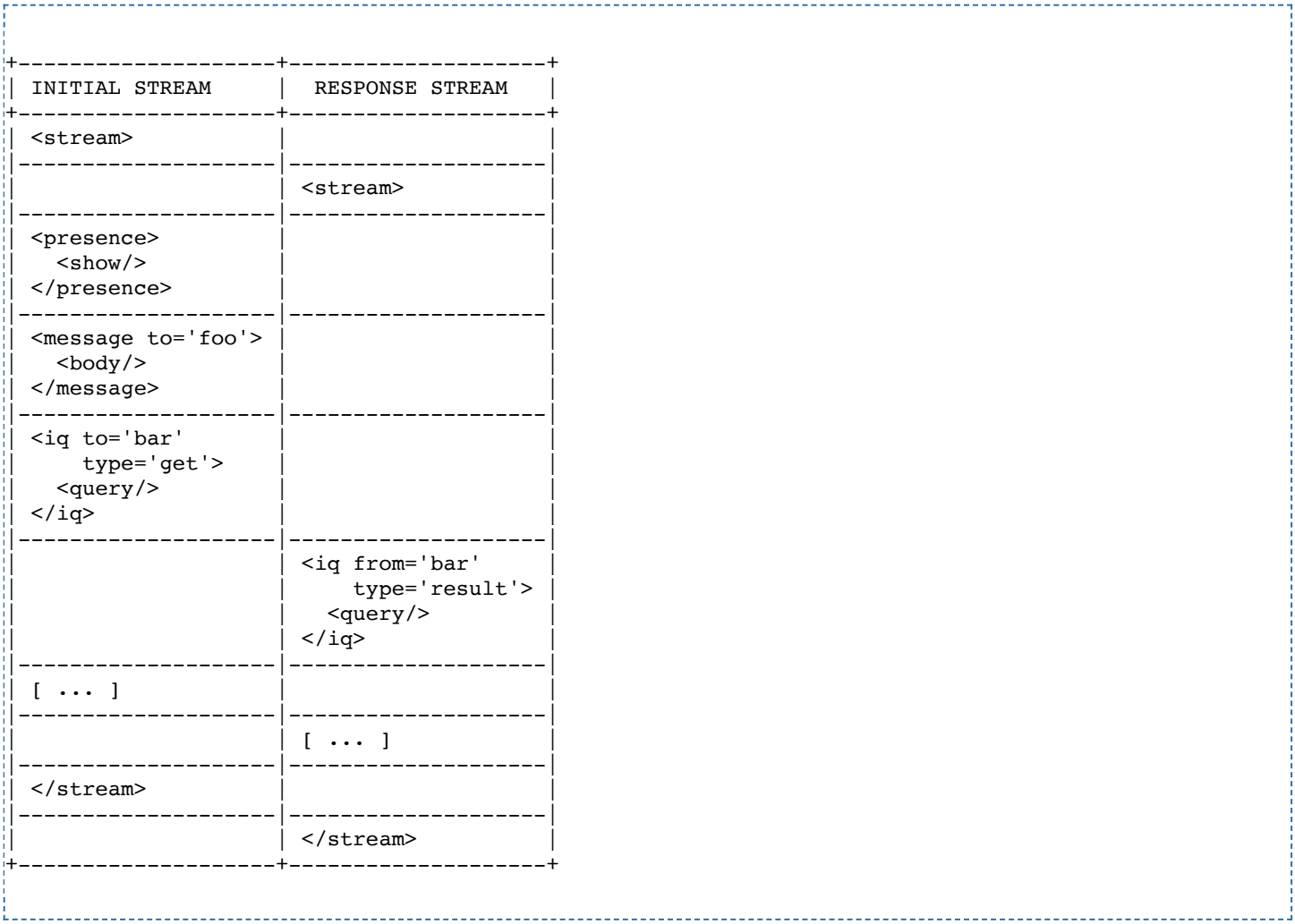


图2: 两路XML流的简化观点

那些习惯于以文档为中心的方式看待XML的人可能会发现下面的类比是有益的:

- 两个XML流像两个 "文档" (相当于 XML 的"文档" 产生)，它们是通过积累XML节来建立的.
- 根 <stream/> 元素像每个 "文档" 的 "文档实体" (详见 XML 的4.8章).
- 通过流发送的XML节像该 "文档"的 "片段" (详见 XML-FRAG).

无论如何, 这些描述只是类比, 因为XMPP不处理文档和片段而是处理流和节.

本节的其余部分定义XML流（连同相关主题）的以下几个方面：

- 如何打开流（ 4.2 ）
- 流协商过程（ 4.3 ）
- 如何关闭流（ 4.4 ）
- XML流的方向性（ 4.5 ）
- 如何处理沉默的对端（ 4.6 ）
- 流的XML属性（ 4.7 ）
- 流的XML命名空间（ 4.8 ）
- 和XML流相关的错误处理（ 4.9 ）

打开流

连接到接收方实体的适当的IP地址和端口之后, 发起方实体通过发送一个流头 ("发起流头") 来打开到接收方实体的流.

```
I: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

然后接收方实体通过发送一个它自己的流头 ("应答流头") 来回复发起方实体.

```
R: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

接着实体们就可以再进行流协商过程的剩余步骤了.

流协商

基本概念

因为流的接收方实体可以说是它所服务的域的看门人, 它对客户端或对端服务器的发起的连接有一定的条件要求. 最低程度, 发起方实体在接收方实体被允许发送节之前需要验证接收方实体, (对客户–服务器流来说这意味着使用 6 中描述的SASL). 无论如何, 接收方实体可以考虑其他验证条件来强制协商, 例如使用 5 描述的TLS加密. 接收方实体通过交流"stream features"以通知发起方实体这些条件: 发起方需要在接收方接受它发送的XML节之前完成的一系列特别的协议交互, 以及任何自愿协商但是可以提高XML流处理的协议交互 (例如, XEP-0138 描述的建立应用层压缩).

连接条件的存在意味着流需要协商. 层的顺序 (TCP, 然后是TLS, 然后是SASL, 然后是XMPP. 这些顺序如13.3描述) 意味着流协商是一个多阶段的过程. 进一步的结构由两个因素来施加: (1) 一个给定的流特性可以仅对特定的实体提供, 或只在特定的其他特性已经被协商之后提供 (例如, 资源绑定仅在SASL验证之后提供), 和 (2) 流特性可能是强制协商也可能是自愿协商. 最后, 基于安全的原因一个流的参与者们在成功地完成用于特定特性的协议交互之后需要丢弃它们在协商过程中获得的知识 (例如, 在所有情况下的TLS和当可能建立一个安全层的情况下的SASL, 如有关SASL机制的规范所述). 通过刷新旧的流上下文和在现有的TCP连接上交换新的流头就可以做到这一点.

流特性格式

如果发起方实体包含在发起流头里的 'version' 属性值设为不低于 "1.0" (见 4.7.5), 接收方实体在发送应答流头之后必须发送一个<features/> 子元素 (通常使用 4.8.5 描述的流命名空间前缀作为前缀) 给发起方实体以声明使流协商过程继续下去的任何条件. 每个条件用 <features/> 元素的子元素的格式, 由一个不同于流命名空间和内容命名空间的命名空间来限定. <features/> 元素可以包含一个子元素, 多个子元素, 或者为空.

> 实现备注: 包含在任何给定的<features/>元素内的子元素的顺序不重要.

如果一个特殊的流特性是或者可以是强制协商的, 那个特性的定义需要做以下几件事之一:

1. 声明这个特性总是强制协商的 (例如, XMPP客户端的资源绑定就是这样的); 或
2. 为接收方实体指定一个方法来标记这个特性在这次交互中为强制协商 (例如, 对于STARTTLS, 这是通过包含一个空的 <required/> 元素到流特性广告中来实现的, 但这不是对所有流特性的通用格式); 建议用于新的强制协商特性的流特性定义如 STARTTLS 所做的那样通过包含一个空的 <required/> 元素来实现.

> 参考文献: 因为没有通用格式来说明一个特性是强制协商的, 有可能会出现一个发起方实体不能理解的特性而被接收方认为是强制协商的, 而导致流协商过程失败. 尽管这样一个结果是不可取的, 本工作组认为不需要通用格式的情况是很罕见的.

基于安全性的原因, 在某些流特性协商成功之后， 发起方有必要发送一个新的发起流头 (例如, 任何情况下的TLS和当建立了安全层的情况下的SASL). 如果一个给定的流特性出现在这种情况下, 那个特性的定义需要指定该特性协商之后的流重启.

一个包含至少一个强制协商特性的<features/>元素表明了流协商没有完成, 发起方实体必须进行进一步的特性协商.

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

一个<features/>元素可以包含不止一个强制协商特性. 这意味着发起方实体能在流协商过程的这个阶段中对强制协商特性进行选择. 举例来说, 可能一个将来的技术将执行和TLS一样的功能, 所以接收方实体可能在这次流协商过程中的同一个阶段声明同时支持TLS和这个将来的技术. 无论如何, 这只适用于流协商过程中的给定阶段而不适用于不同阶段的强制协商特性 (例如, 接收方实体不会声明同时支持STARTTLS和SASL作为强制性协商, 或同时声明SASL和资源绑定为强制协商, 因为TLS需要在SASL之前协商并且SASL需要在资源绑定之前协商).

一个同时包含强制协商和自愿协商特性的<features/>元素表明协商未完成, 发起方实体可以在它尝试协商强制协商特性之前完成自愿协商特性.

```
R: <stream:features>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
     <compression xmlns='http://jabber.org/features/compress'>
       <method>zlib</method>
       <method>lzw</method>
     </compression>
   </stream:features>
```

一个只包含自愿协商特性的<features/>元素表明流协商已经完成, 发起方实体可以开始发送XML节了. 但是如果发起方实体愿意可以协商更多特性.

```
R: <stream:features>
     <compression xmlns='http://jabber.org/features/compress'>
       <method>zlib</method>
       <method>lzw</method>
     </compression>
   </stream:features>
```

一个空的<features/>元素表明流协商已经完成, 发起方实体可以开始发送XML节了.

```
R: <stream:features/>
```

重启

在对一个需要流重启的特性成功协商之后, 双方都必须考虑前一个流将被取代, 但是必须不能发送一个关闭的</stream>标签并且必须不能终止底层的TCP连接; 反之, 双方必须重用现有的连接, 它可以处于一个新的状态(例如, 作为一个TLS协商的结果被加密). 然后发起方实体必须发送一个新的发起流头, 它之前应该放一个如 11.5 所述的XML声明. 当接收方实体接收到新的发起流头, 它必须在发送一个新的应答流头(它之前应该放一个如 11.5 所述的XML声明) 之前生成一个新的流ID(而不是重用旧的流ID).

重发特性

接收方实体必须在一个流重启之后发送一个流特性的更新列表给发起方实体. 如果没有更多的特性要被声明, 这个更新的流特性列表可以是空的, 也可以包含任何特性的组合.

完成流协商

接收方实体通过发送一个空的<features/>元素或只包含自愿协商特性的<features/>元素来表明流协商过程的完成. 这样做之后, 接收方实体可以发送一个空的<features/> 元素(例如, 在这些自愿协商特性协商完成之后) 但是必须不能发送额外的流特性给发起方实体(如果接收方实体有新的特性提供, 最好仅限于强制协商或安全关键的特性, 它可以简单地以一个<reset/>流错误(4.9.3.16)来关闭流并且等发起方重新连接的时候声明新的特性, 最好以一个交错的方法来关闭现有的流, 这样不会让所有的发起方同时进行重连). 一旦流协商完成, 发起方就可以一直通过这个流来发送XML节, 只要双方都维持着这个流.

> 参考文献: 在下面 第7章 定义的资源绑定是一个前述规则的一个历史性的例外, 因为对于客户端来说它是强制协商的但使用XML节来达成协商.

在流协商完成之前, 发起方实体不能(MUST NOT)尝试发送 XML节 给非自身的实体(也就是说, 客户端的已连接资源或客户端帐号的任何其他已验证的资源) 或给它连接的服务器. 即使发起方尝试这么做, 接收方实体也不能(MUST NOT)接受这些节并且必须以一个<not-authorized/>流错误(4.9.3.12)来关闭流. 这个规则只适用于XML节(也就是说, 由内容命名空间限定的 <message/>, <presence/>, 和 <iq/> 元素) 而不是用于流协商的XML元素(例如, 完成 TLS协商 或 SASL协商 的元素).

确定地址

在一个XML流的双方已经完成了流协商的适当步骤之后, 流的接收方实体必须确定发起方实体的JID.

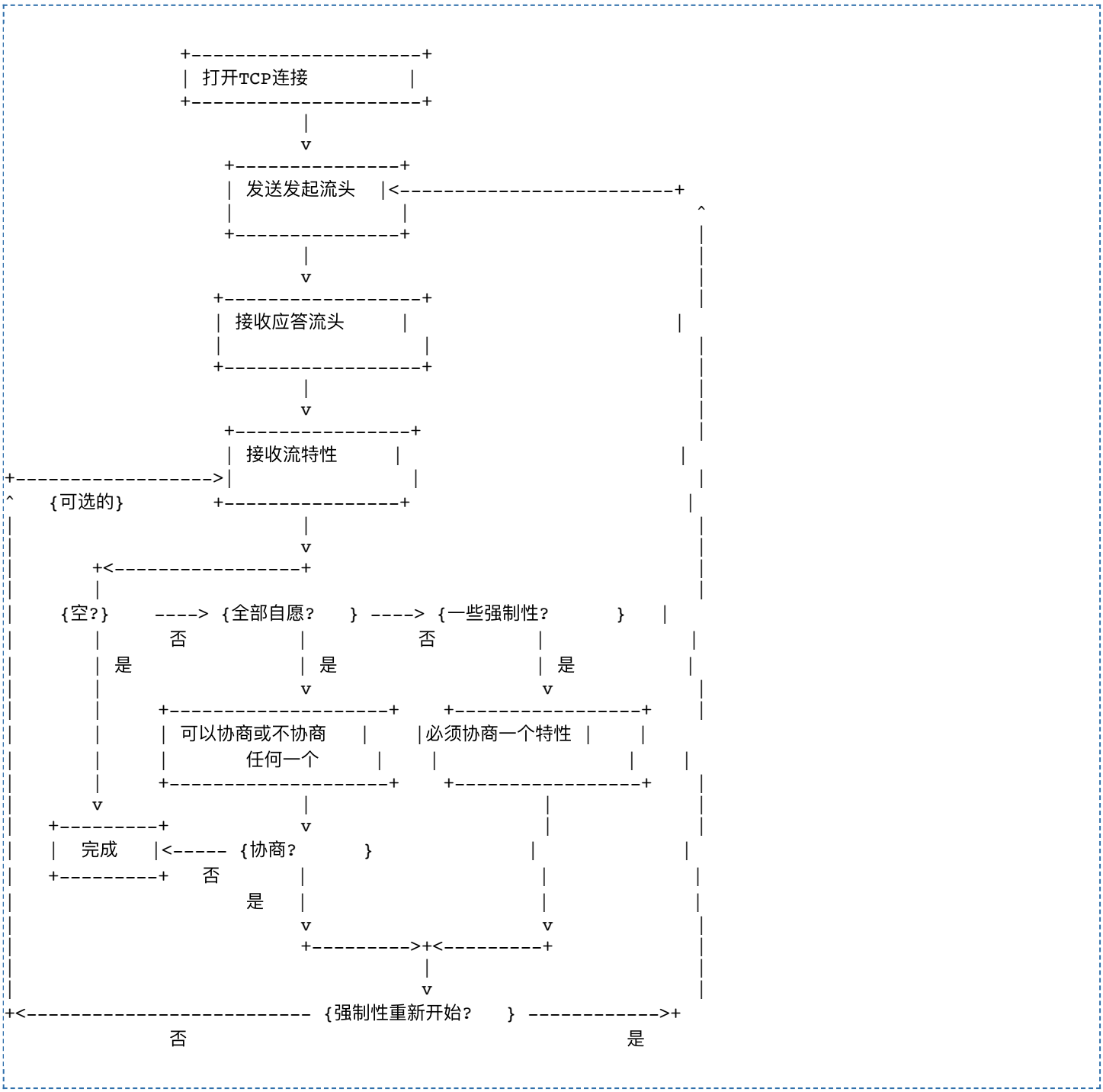
对于客户端-服务器的通讯, 在服务器能确定客户端的地址之前, SASL协商 和 资源绑定 必须完成. 客户端的纯JID (<localpart@domainpart>) 必须是授权身份 (如 SASL 所定义的, 要么 (1) 是 SASL协商期间客户端直接与之通讯的身份,要么 (2) 如果SASL协商期间没有指定授权身份, 则是从服务器的验证身份指定的. 全JID(<localpart@domainpart/resourcepart>)的资源部分(resourcepart)必须是客户端和服务器在 资源绑定 期间协商得来的那个资源. 客户端必须不去尝试猜测它的JID而是必须相信在资源绑定期间服务器返回给它的JID. 服务器必须确保返回的这个JID (包含 本地部分(localpart), 域部分(domainpart), 资源部分(resourcepart), 以及分隔符) 遵循定义于 XMPP-ADDR 的XMPP地址规范格式; 为了满足这个限定, 服务器可以把客户端的发来的JID替换成服务器确定的规范JID并在资源绑定期间使用那个JID来和客户端通讯.

对于服务器–服务器通讯, 发起方服务器的纯JID (<domainpart>) 必须是授权的身份 (定义于 SASL), 要么 (1) 是SASL协商 期间发起方服务器直接与之通讯的身份,要么 (2) 如果SASL协商期间没有指定授权身份， 则是从接收方服务器的验证身份指定的. 在缺少SASL协商的情况下, 接收方服务器可以认为授权身份是一个和相关确认协议协商的身份(例如, 在 服务器回拨XEP-0220 中 <result/> 元素的'from'属性.

> 安全警告: 因为可能会有第三方在一个安全层例如TLS成功协商之前篡改流上发送的数据, 建议接收服务器谨慎对待这些未得到保护的信息; 这特别适用于由发起方实体发送的第一个发起流头中的'from'和'to'地址.

流程图

我们在接下来的非规范性的流程图里为流协商过程总结前述的规则, 以发起方实体的视角来展示.



图表3: 流协商流程图

关闭流

从一个实体到另一个实体的XML流可以在任何时候关闭, 可以是因为发生了一个特定的流错误 (4.9章)) 也可以没有任何错误(例如, 当客户端简单地结束它的会话).

通过发送一个关闭的</stream>标签来关闭流.

```
E: </stream:stream>
```

如果双方正在使用一个TCP连接上两个流, 或者两个TCP连接上两个流的通信方式, 发送关闭流标签的实体必须按以下步骤进行:

1. 在终止底层的TCP连接之前等待另一方也关闭它的出站流; 这让另一方有机会在终止TCP连接之前发完任何到正在关闭的实体的出站数据.
2. 避免通过它的出站流发送任何更多的数据到另一方, 但是继续处理从另一方实体已经接收到的数据(并且, 如果必要, 处理这些数据).
3. 如果另一方没有在一个合理的时间(这里 "合理" 的定义取决于实现或布署)内发送它的关闭流标签则认为两个流都失效了.
4. 从另一方接收到一个反向的关闭流标签之后, 或在等待一段合理的时间之后未收到应答, 终止底层的TCP连接.

   安全警告: 根据 TLS 的7.2.1章节, 为了帮助防止截断攻击， 正在关闭流的那一方在终止底层TCP连接之前必须发送一个 TLS close_notify警告, 并且必须从另一方接收到应答的 close_notify警告.

如果双方通过多重TCP连接使用多重流, 那么没有已定义的配对流, 因此其行为取决于实现.

方向性

一个XML流总是单向的, 这意味着只能从一个方向通过流来发送XML节(要么从发起方实体到接收方实体， 要么从接收方实体到发起方实体).

取决于已协商的会话的类型以及涉及的实体的性质, 该实体可以使用:

- 在一个TCP连接上跑两个流, 这里第一个流协商的安全性上下文也适用于第二个流. 这对客户端–服务器是典型的情况, 并且服务器必须允许客户端为两个流使用同一个TCP连接.
- 在两个TCP连接上跑两个流, 这里每个流是独立保障安全的. 在这种方法下, 一个TCP连接用于发起方实体发送节到接收方实体的那个流, 另一个TCP连接用于接收方实体发送节到发起方实体的那个流. 这对服务器–服务器会话是典型的情况.
- 在两个或更多TCP连接上跑多个流, 这里每个流是独立保障安全的. 这个方法有时用于两个大的XMPP服务提供商之间的服务器–服务器通讯; 无论如何, 在 10.1 描述的情况下这将导致难于维护从多个流接收到的数据的一致性, 这是为什么如果远程服务器尝试如4.9.3.3 所述来协商多个流， 本服务器可能以一个<conflict/>流错误(4.9.3.3)来关闭流.

这个方向性的概念只适用于节, 并且明确地不适用于根流(stream root)的第一级子元素, 那些根流是被用来启动或管理流的(例如, 用于TLS协商, SASL协商, 服务器回拨 XEP–0220 , 流管理 XEP–0198 的第一级元素).

上述的考虑暗示当完成 STARTTLS协商 和 [RFC6120#SASL协商|SASL协商]] 的时候，两个服务器将使用同一个TCP连接, 但是在流协商过程完成之后，原始的那个TCP连接将仅用于发起方服务器发送XML节到接收方服务器. 为了让接收方服务器能发送XML节给发起方服务器, 接收方服务器将需要反转角色并通过一个单独的TCP连接从接收方服务器向发起方服务器协商一个XML流. 然后这个单独的TCP连接用新一轮的 TLS 和/或 SASL 协商来保证安全性.

> 实现备注: 基于历史原因, 一个服务器–服务器会话总是使用两个TCP连接. 这个方法仍然在本文描述的标准行为之内, 类似 XEP-0288 的扩展允许服务器们使用单一的TCP连接协商来进行双向的节交换.

> 参考文献: 尽管XMPP开发者们有时对底层TCP连接使用了"单向"和"双向"的概念 (例如, 把用于客户端–服务器会话的TCP连接称为"双向的" 而用于服务器–服务器会话的TCP连接称为"单向的"), 严格来讲一个流总是单向的 (因为发起方实体和接收方实体总是最少有两个流, 每个方向一个) 并且一个TCP连接总是双向的 (因为TCP通讯能从两个方向发送). 方向性应用于TCP连接的应用层通讯, 而不是在TCP连接的传输层通讯本身.

无响应对端的处理

当连接某个流的实体在一段时间内没有接收到同样连接到该流的另一对等端发来的任何XMPP信息，那么该对等端可能是无响应的。有几个原因很可能引起这种情况发生：

1. 底层的TCP连接死掉。
2. 尽管当底层的TCP连接仍然是激活的时候，XML流被中断了。
3. 对等端是空闲的，只是没有通过其连接的XML流发送XMPP信息到该实体。

这三个条件最好被分别对待，像下面章节描述的一样。

> 实现注意:为了处理无响应的对等端，我们把两个单向的TCP连接当作一个在概念上相当的双向的TCP连接（见4.5节（方向性））;然而，实现者要知道在两个单向的TCP连接的情况下，在XMPP应用层对等端对通信的响应将从其第二个TCP连接返回。此外，在每个方向上的多个数据流的使用（大型的XMPP服务提供商间的服务器到服务器之间的连接经常这么部署）使得对XMPP流和底层TCP连接的应用级检查进一步复杂化了，因为任何给定的初始流和任何给定的响应流之间没有必然联系。

死连接

如果底层的TCP连接是死的，流级别的检查（例如：[XEP–0199 (http://xmpp.org/rfcs/rfc6120.html#XEP–0199)]和[XEP–0198 (http://xmpp.org/rfcs/rfc6120.html#XEP–0198)]）是无效的。因此，不管有没有流错误都没必要关闭流，恰当的做法是直接终止TCP连接。

检查TCP连接的一个通用方法是在XML节之间发送一个空格符（U+0020）,发送空格在XML流中是被允许的，下面第11.7节中将进行描述。发送这样的一个空格被称作“空格保持激活”（词语“whitespace ping”通常被使用，尽管事实上它不是一个ping,因为“pong”是不可能的）。然而，在TLS认证和SASL认证期间，发送空格符是不允许的，下面第5.3.3节和第6.3.5节将会描述。

中断的流

即使底层的TCP连接仍然是激活的，对等端很可能从来不对实体发出的XMPP通信请求作出响应，不管是正常的节还是例如在[XEP-0199 (http://xmpp.org/rfcs/rfc6120.html#XEP-0199)]中所定义的应用程序级别ping那样的专门流通信检查，或者在[XEP-0198 (http://xmpp.org/rfcs/rfc6120.html#XEP-0198)]中定义的更全面的流管理协议。在这种情况下，对实体来说恰当的做法是发送<connection-timeout/>流错误（第4.9.3.4节）来关闭中断的流。

空闲对端

即使底层的TCP连接仍然激活，并且流没有被中断，对等端很可能在一段时间内都没有发送XML节。在这种情况下，对等端可以（MAY）关闭流（像第4.4节描述的一样），而不是让不再使用的流打开着。如果空闲对等端没有关闭流，那么与该流关联的另一端或者可以（MAY）通过使用第4.4节描述的握手方式来关闭该流，或者发送一个流错误（例如：<resource-constraint/>（第4.9.3.17节），如果实体已经到了打开TCP连接数的限制，或者<policy-violation/>（第4.9.3.14节），如果连接已超出本地超时政策）来关闭该流。然而，层（下面第13.3节指定）的顺序要符合，在得出对等端处于空闲状态的结论前，另一端需要去核实底层TCP连接仍然是激活的并且流没有被中断（前面已描述）。此外，在接受空闲对等端时最好宽容一点，因为经验表明，这样做可以提高在XMPP网络上通信的可靠性，并且保持两个服务器之间的流比积极地去超时一个流通常更有效。

检查方法的使用

建议实现人员支持任何他们认为合适的流检查和连接检查方法, 但是要小心衡量这些方法对网络的冲击和及时发现中断的流和死TCP连接得到的好处. 使用的任何特定检查方法的时间间隔对本地服务策略都是一个大事情，并且严重依赖于网络环境和给定布署和连接类型的使用场景. 在撰写本文的时候, 建议任何这类检查的执行不要超过每5分钟一次, 并且理想的情况是, 这类检查由客户端发起而不是服务器来发起. 鼓励那些实现XMPP软件和布署XMPP服务的人对适当的流检查和连接检查时间间隔寻求其他的意见, 特别是当使用了功率受限的设备的时候 (例如, 在移动环境).

流属性

根<stream/>元素的属性定义在以下章节.

安全警告: 在流的保密性和安全性被 第5章 描述的TLS或一个相当的安全层(类似SASL GSSAPI机制)保护之前, 一个流头提供的属性们可能会被攻击者篡改.

实现备注: 根<stream/>元素的属性不是以一个命名空间前缀来前置是因为, 根据 XML-NAMES 的解释, "[d]缺省的命名空间声明不直接应用用于属性名称; 没有前缀的属性的解释由它们出现的那个节来决定."

from

'from'属性指定发送流元素的实体的XMPP标识。

对客户机-服务器通信的发起流头而言，'from'属性是控制客户端的主要XMPP标识，如格式为<localpart@domainpart>的JID。客户端可能不知道XMPP标识，因为XMPP标识不是在XMPP应用层的等级上被分配的（如通用安全服务应用程序接口[GSS-API (http://xmpp.org/rfcs/rfc6120.html#GSS-API)]中描述），或者从客户端提供的信息由服务器生成（像采用SASL EXTERNAL机制的终端用户证书部署）。此外，如果客户端认为XMPP标识是私人信息，那么在流的保密性和完整性被TLS或等效的安全层保护之前是不提倡包括一个'from'属性的。但是，如果客户端知道XMPP标识，它应该（SHOULD）在流的保密性和完整性被TLS或等效的安全层保护之后包括'from'属性。

```
I: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

对于服务器-服务器通信的发起流头而言，'from'属性是服务器配置的FQDN之一，如，格式如<domainpart>的JID。初始服务器可能有一个以上的XMPP标识，如，在服务器提供虚拟主机的情况下，所以它需要选择一个与此输出流关联的标识（如：基于触发流协商尝试的节的'to'属性）。因为服务器是XMPP网络上的“公共实体”，它必须（MUST）在流的保密性和完整性被TLS或等效的安全层保护之后包含'from'属性。

```
I: <?xml version='1.0'?>
   <stream:stream
       from='example.net'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

对于客户端-服务器和服务器-服务器通信的响应流头，接收实体必须（MUST）包括'from'属性并且必须（MUST）把该属性值设置为接收实体的FQDN之一（可以（MAY）是一个正式域名（FQDN），而不是发起流头中'to'属性指定的值，下面第4.9.1.3节和第4.9.3.6节将描述）。

```
R: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
```

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

无论'from'属性是否被包括，在和其它实体交换XML节之前，每个实体必须（MUST）核实其它实体的身份，下面第13.5节将描述。

> 互操作性备注：基于[RFC3920 (http://xmpp.org/rfcs/rfc6120.html#RFC3920)]的实现在任何流头（即使其保密性和完整性受到保护）中不包括'from'地址是可以的，一个实体在接收这样的流头时应该（SHOULD）宽容些。

to

> 对于客户端-服务器和服务器-服务器通信中的发起流头，发起实体必须（MUST）包含'to'属性，并且必须（MUST）使用发起实体知道或者期望接收实体提供服务的域名来设置'to'属性的值。（在其他方面，如[TLS-EXT (http://xmpp.org/rfcs/rfc6120.html#TLS-EXT)]中所描述，在TLS协商期间的服务器名称显示，也可以提供类似的信息。）

```
I: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

> 对于客户端-服务器通信的响应流头，如果客户端在其发起流头中包含'from'属性，那么服务器在应答流头中必须（MUST）包含'to'属性，并且必须（MUST）使用初始化流头中'from'属性指定的裸JID来设置'to'属性的值。如果客户端在其初始化流头中不包含'from'属性，那么服务器在应答流中不必（MUST NOT）包含'to'属性。

```
R: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

> 对于服务器-服务器通信的响应流头，接收实体在应答流头中必须（MUST）包含'to'属性，并且必须（MUST）使用发起流头中'from'属性指定的域名部分来设置'to'属性的值。

```
R: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='g4qSvGvBxJ+xeAd7QKezOQJFFlw='
       to='example.net'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

无论是否包含'to'属性，每个实体在与其它实体交换XML节之前必须（MUST）先核实其它实体的身份，如下面第13.5节所描述。

互操作性备注：基于[RFC3920 (http://xmpp.org/rfcs/rfc6120.html#RFC3920)]的实现在任何流头中不包括'to'地址是可以的，一个实体在接收这样的流头时应该（SHOULD）宽容些。

id

'id'属性是流的唯一标识符，称为“流ID”。流ID必须（MUST）由接收实体在发送应答流头时生成，并且在接收的应用（一般是一个服务器）内部，所生成的流ID必须（MUST）是唯一的。

安全警告：流ID都必须（MUST）是不可预测的和非重复的，因为身份验证机制重用它时可能引起安全隐患，如服务器回拨[XEP-0220 (http://xmpp.org/rfcs/rfc6120.html#XEP-0220)]和“XMPP 0.9”身份验证机制，“XMPP 0.9”机制是RFC 3920在XMPP使用SASL机制之前所使用的身份认证机制;出于安全目的 的随机性建议,见[RANDOM (http://xmpp.org/rfcs/rfc6120.html#RANDOM)]。

对于初始流头，初始实体不必（MUST NOT）包含'id'属性;但是，如果包含'id'属性，接收实体必须（MUST）忽略它。

对于响应流头，接收实体必须（MUST）包括'id'属性。

```
R: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

互用性备注：在RFC 3920中，列入'id'属性的文字含糊不清的，导致一些实现的id属性脱离了响应流头。

xml:lang

'xml:lang'属性指定一个实体在该流上发送的任何可读XML字符串数据的首选或缺省语言(XML节也可以拥有'xml:lang'属性, 定义在 8.1.5). 这个属性的语法定义于 XML 的2.12节; 特别是, 'xml:lang'属性必须符合 NMTOKEN 数据类型 (定义于 XML 的2.3节) 同时必须符合定义于 LANGTAGS 的语言标识符.

对于发起流头, 发起方实体应该包含 'xml:lang' 属性.

```
I: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

对于应答流头, 接收方实体必须包含 'xml:lang' 属性. 应用以下规则:

- 如果发起方实体在它的发起流头包含了'xml:lang'属性，而接收方实体在它生成和发送给发起方实体的可读XML字符串数据(例如, 流和节错误中的<text/>元素)中支持那个语言, 'xml:lang'属性值必须是发起方实体首选语言的标识符(例如, "de-CH").
- 如果接收方实体根据定义于LANGMATCH 3.4节的 "lookup scheme"(例如, "de" 而不是 "de-CH")支持一种和发起方实体首选语言匹配的语言, 那么'xml:lang'属性值应该是匹配语言的标识符.
- 如果接收方实体不支持发起方实体的首选语言或根据查找方案匹配的语言(或者如果发起方实体未在其发起流头中包含'xml:lang'属性), 那么'xml:lang'属性值必须是接收方实体的缺省语言的标识符(例如, "en").

```
R: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

如果发起方实体在它的发起流头包含了'xml:lang'属性, 接收方实体应该记住那个值并以此作为发起方实体通过当前流发送的所有节的缺省 xml:lang . 如下面的 8.1.5 所述, 发起方实体可以在它通过这个流发送的任何XML节中包含'xml:lang'属性. 如果发起方实体在任何这类节中未包含'xml:lang'属性, 接收方实体在路由它到远程服务器或递送它到一个已连接的客户端时应该加上'xml:lang'属性, 而这个属性的值必须是发起方实体的首选语言的标识符(即使接收方实体生成和发送给发起方实体的可读XML字符串数据不支持该语言, 类似流或节错误等). 如果发起方实体在任何这类节中包含了'xml:lang'属性, 接收方实体在路由它到远程服务器或递送到已连接的客户端时必须不能修改和删除它.

version

包含值至少为“1.0”的版本属性传达了对本文所定义的流相关的协议支持的信号，这些流相关的协议包括TLS协商，SASL协商，流特征，流错误。

本文规定的XMPP版本是“1.0”；尤其，XMPP 1.0 封装了流相关协议以及三个定义的XML节类型的基本语义（<message/>, <presence/>,和 <iq/>，下面8.2.1节，8.2.2节，8.2.3节分别描述）。

XMPP版本编号方案是“<主版本>.<次版本>（‘<major>.<minor>’）”。主版本和次版本号必须（MUST）视为独立的整数，并且每个数可能（MAY）会以高于一个单一的数字的方式递增。因此，“XMPP 2.4”版本将低于“XMPP 2.13”，同理，“XMPP 2.13”版本将低于“XMPP 12.3”。接受方必须（MUST）忽略前导零（例如，“XMPP 6.01”），并且不能（MUST NOT）发送。

只有重大的新功能已被添加到核心协议（如：对message, presence, 或者IQ节新定义一个‘TYPE’属性值），次要版本号才递增。具有较小次版本号的实体必须（MUST）忽略版本比它大的次版本号，但是具有较大次版本号的实体为了一些信息目的可以使用版本比它小的次版本号（如：具有较大次版本号的实体将仅仅关注其通信器将不能理解‘TYPE’属性的值，因此不发送它）。

下述规则适用于流头内“version”属性的生成和处理：

1. 初始化实体必须（MUST）在初始化流头中设置‘version’属性的值为它所支持的最高版本号（如：如果它支持的最高版本号是本文中定义的，它必须（MUST）设置该值为“1.0”）。
2. 接收实体必须（MUST）在应答流头中设置‘version’属性的值或者为初始化实体提供的值，或者为接收实体所支持的最高版本号，以较低者为准。接收实体必须（MUST）执行主要和次要版本号的数值比较，而不是关于“<主版本>.<次版本>（‘<major>.<minor>’）”的字符串匹配。
3. 如果在应答流头中包含的版本号至少在主版本号上比在初始化流头中包含的版本号低，那么较新版本的实体不能与旧版本的实体互操作，初始化实体应该（SHOULD）通过发送一个包含<unsupported-version/>元素的流错误（第4.9.3.25节）来关闭流。
4. 如果任何一个实体接收到一个没有‘version’属性的流头，该实体必须（MUST）认为其他实体支持的版本是“0.9”，并且不应该在响应流头中包含‘version’属性。

流属性总结

下表总结了根元素<stream/>的属性。

	初始实体 到 接收实体	接收实体 到 初始实体
to	接收实体JID	初始实体JID
from	初始实体JID	接收实体JID
id	忽 略	流标识
xml:lang	默认语言	默认语言
version	XMPP 1.0+ supported	XMPP 1.0+ supported

图4：流属性

XML命名空间

读者可以参考 XML-NAMES 来完整地理解本章的概念, 特别是本协议的第三章和第6.2节的 "缺省命名空间" 的概念.

流命名空间

<stream/> 根元素 ("流头") 必须由 'http://etherx.jabber.org/streams' (即 "流命名空间") 命名空间来限定. 如果违反了这个规则, 接受到该流头的实体必须以一个流错误来关闭流, 这个流错误应该是 <invalid-namespace/> (4.9.3.10), 尽管一些现存的实现发送的是 <bad-format/> (4.9.3.1) .

内容命名空间

实体可以声明一个"内容命名空间" 作为缺省的命名空间用于在流上发送的数据 (也就是那些不属于由流命名空间限定的元素的数据). 如果这样做, (1) 内容命名空间必须不同于流命名空间, 并且 (2) 内容命名空间对于发起流和应答流必须是相同的, 使得两个流的限定是一致的. 内容命名空间应用所有从流上发送的一级子元素,除非被显式地由另一个命名空间来限定 (即, 内容命名空间是缺省命名空间).

另外 (也就是说, 不定义内容命名空间作为缺省命名空间), 实体可以显式地为流的每个一级子元素限定命名空间, 使用所谓 "自由前缀规范". 这两种方式在下面的例子中展示.

当声明了一个内容命名空间作为缺省命名空间时, 大致来说一个流看起来类似下面的例子.

```
<stream:stream
    from='juliet@im.example.com'
    to='im.example.com'
    version='1.0'
    xml:lang='en'
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'>
  <message>
    <body>foo</body>
  </message>
</stream:stream>
```

当没有定义内容命名空间作为缺省命名空间而使用所谓"自由前缀规范"时, 大致一个流看起来像以下例子.

```
<stream
    from='juliet@im.example.com'
    to='im.example.com'
    version='1.0'
    xml:lang='en'
    xmlns='http://etherx.jabber.org/streams'>
  <message xmlns='jabber:client'>
    <body>foo</body>
  </message>
</stream>
```

传统上, 大多数XMPP实现在流头使用了 把内容命名空间作为缺省命名空间 的方式, 而不是 自由前缀规范 的方式; 无论如何, 两种方式都是可以接受的, 因为它们在语义上是等价的.

XMPP内容命名空间

本协议定义的XMPP使用两种内容命名空间: 'jabber:client' 和 'jabber:server'. 这些命名空间差不多相同但是用于不同的上下文 ('jabber:client'用于 客户端-服务器 通讯而'jabber:server' 用于 服务器-服务器 通讯). 两者之间唯一的不同是在通过'jabber:client'命名空间限定的XML流发送的节中 'to' 和 'from' 属性是可选的, 而在'jabber:server'限定的XML流发送的节中它们是必需的. 支持这些内容命名空间意味着支持常规属性和所有三个核心节类型 (message, presence, 和IQ)的基本语义.

一个实现可以支持不同于 'jabber:client' 或 'jabber:server' 的内容命名空间. 然而, 因为这些命名空间将定义不同于XMPP的应用, 它们将在单独的协议中定义.

一个实现可以拒绝支持任何与缺省命名空间不同的其他内容命名空间. 如果一个实体不支持它接收到的一个一级子元素的内容命名空间, 它必须以 <invalid-namespace/> 流错误(4.9.3.10)关闭这个流 .

客户端实现必须支持'jabber:client'内容命名空间作为缺省命名空间. 'jabber:server'内容命名空间超出了XMPP客户端的范畴, 并且客户端不能(MUST NOT)发送'jabber:server'命名空间限定的节.

服务器实现必须同时支持'jabber:client'命名空间(当这个流用于客户端和服务器之间通讯的时候)和'jabber:server'命名空间(当这个流用于两个服务器之间的通讯的时候)作为缺省命名空间. 和一个已连接的客户端通讯时,该服务器不能(MUST NOT)发送由'jabber:server'命名空间限定的节; 和一个对端服务器通讯时, 该服务器不能(MUST NOT)发送由'jabber:client'命名空间限定的节.

> 实现备注: 因为一个客户端通过一个内容命名空间为'jabber:client'的流发送节, 如果一个服务器路由一个它从一个已连接的客户端收到的节到另一个对端服务器,那么它需要为这个节 "重新界定范围(re-scope)" ， 所以它的命名空间是'jabber:server'. 类似的, 如果一个服务器递送一个它从别的对端服务器收到的节到一个已连接的客户端， 那么它需要给这个节 "重新界定范围" ， 所以它的内容命名空间是'jabber:client'. 这个规则适用于4.1定义的XML节(即, 一个由'jabber:client'或'jabber:server'命名空间限定的顶级的 <message/>, <presence/>, 或 <iq/> 元素), 并且命名空间继承到一个节的所有子元素. 然而, 这个规则不适用于非'jabber:client'和'jabber:server'限定的元素以及它们的任何子元素(例如, 一个包含了用于报告的扩展元素(8.1)的 <message/>元素). 尽管不可能禁止一个实体在一个扩展元素中生成的节的子元素由'jabber:client'或'jabber:server'命名空间来限定, 现有的实现处理这类节的方法是不一致的; 因此, 建议实现者自己权衡选择降低互操作性还是损失这些节的功能. 最后, 建议服务器对于使用其他流连接方法和替代的XMPP连接方法而来的节重新界定范围, 例如那些定义在 XEP-0124 , XEP-0206 , XEP-0114 , 和 XEP-0225 中的节.

其他命名空间

参与一个流的双方都可以发送非内容命名空间和流命名空间限定的数据. 例如, 和TLS协商以及SASL协商相关的数据如何交换, 以及类似 流管理XEP-0198和服务器回拨XEP-0220的XMPP扩展.

> 互操作性备注: 由于历史的原因, 一些服务器实现预期有一个'jabber:server:dialback'命名空间的声明用于 服务器-服务器 流, 详见XEP-0220.

无论如何, 一个XMPP服务器不能(MUST NOT)路由或递送这样一个从入站流中接收到的数据， 如果那个数据 (a) 是由其他命名空间限定 并且 (b) 地址指向的实体不是一个服务器, 除非从出站流发送数据出来的另一方服务器显式地协商或声明支持从该服务器接收任意数据. 制定这个规则是因为XMPP是设计用来做XML节交换的(而不是任意XML数据), 也因为允许实体发送任意数据到其他实体可能会导致恶意数据交换显著增加. 作为本规则的例子, example.net 域主机将不会从 <romeo@example.net> 到 <juliet@example.com> 的路由一级XML元素:

```
<ns1:foo xmlns:ns1='http://example.org/ns1'
           from='romeo@example.net/resource1'
           to='juliet@example.com'>
   <ns1:bar/>
 </ns1:foo>
```

这个规则也适用于看起来像节但是命名空间不正确所以并不是真的节的一级元素(参见 4.8.5), 例如:

```
<ns2:message xmlns:ns2='http://example.org/ns2'
              from='romeo@example.net/resource1'
              to='juliet@example.com'>
   <body>hi</body>
 </ns2:message>
```

在从一个入站流中接收到任意一级XML元素之后, 服务器必须要么忽略该数据要么以一个流错误关闭这个流, 这个流错误应该是 <unsupported-stanza-type/> (4.9.3.24).

命名空间声明和前缀

因为内容命名空间不同于流命名空间, 如果把一个内容命名空间声明为缺省命名空间，那么以下论断为真:

1. 流头需要包含一个同时用于内容命名空间和流命名空间的命名空间声明.
2. 流命名空间的声明需要包含用于流命名空间的前缀.

互操作性备注: 由于历史原因, 一个实现可能只包含前缀'stream'用于流命名空间(包含了前缀的结果就像<stream:stream>和<stream:features>); 这个来自RFC3920的规范保留下来用于向后兼容. 实现如果使用不同于'stream'的前缀用于流命名空间将导致互操作性问题. 如果实体接收到一个流头而该流头的流命名空间前缀是不可接受的, 它必须以一个流错误来关闭这个流, 这个流错误应该是 <bad-namespace-prefix/> (4.9.3.2), 虽然一些现有的实现发送的是 <bad-format/> (4.9.3.1) .

一个实现不能(MUST NOT)为由内容命名空间限定的元素生成命名空间前缀(即, 在流上发送的数据的缺省命名空间)，如果内容命名空间是'jabber:client'或'jabber:server'. 例如, 以下是非法的:

```
<stream:stream
    from='juliet@im.example.com'
    to='im.example.com'
    version='1.0'
    xml:lang='en'
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'>

  <foo:message xmlns:foo='jabber:client'>
    <foo:body>foo</foo:body>
  </foo:message>
```

XMPP实体不应该接受违反这一规则的数据(特别是, XMPP服务器如果不首先纠正这个错误，就不能(MUST NOT) 路由这类数据到另一个实体); 替代方案是，它应该要么忽略这个数据要么以流错误来关闭这个流, 这个流错误应该是 <bad-namespace-prefix/> (4.9.3.2).

在一个流头中对命名空间的声明必须只应用于那个流(例如, 'jabber:server:dialback' 命名空间用于服务器回拨 XEP-0220). 特别是, 因为用于通过流路由或递送到其他实体的XML节将丢失在那些生成节的原始流头中的命名空间的上下文, 这些节中的扩展内容的命名空间不能(MUST NOT)在那个流头中声明(参见 8.4). 如果流的参与双方声明了这样的命名空间, 这个流的其他参与方应该以<invalid-namespace/>流错误关闭该流(4.9.3.10). 在任何情况下, 实体必须确保当从入站流路由或递送节到出战流的时候，这些命名空间(根据本章)被正确地声明 .

流错误

流的根元素可以包含一个由流命名空间限定的<error/>子元素. 这个错误子元素将由兼容的实体来发送，如果它发现发生了一个流错误.

规则

以下规则适用于流这一级的错误.

流错误是不可恢复的

流级别的错误是不可恢复的. 所以, 如果一个错误发生在流这个级别, 检测到这个错误的实体必须发送一个<error/>元素，其中包含适当的子元素以指明错误情况，并如4.4所述立刻关闭这个流.

```
C: <message><body>No closing tag!</message>

S: <stream:error>
     <not-well-formed
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

接着收到这个流错误的实体将如4.4所述关闭这个流.

```
C: </stream:stream>
```

流错误可能发生在安装过程中

如果该错误是被初始化流头触发的, 那么接收方实体必须仍然发送打开的<stream>标签, 把这个<error/>元素作为该流元素的子元素, 并且发送关闭流的</stream>标签(最好在同一个TCP包里).

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://wrong.namespace.example.org/'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <invalid-namespace
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

当主机未定义或未知时会发生流错误

如果初始化实体未提供'to'属性或在'to'属性里提供了一个未知的主机并且这个错误发生在流安装的时候, 在接收方实体关闭这个流之前，由接收方实体返回给初始方实体的流头的'from'属性必须要么是接收方实体的可靠的完整域名要么是空字符串.

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='unknown.host.example.com'
       version='1.0'
```

```
      xml:lang='en'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
      from='im.example.com'
      id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
      to='juliet@im.example.com'
      version='1.0'
      xml:lang='en'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <host-unknown
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

流错误发到哪

当在初始方实体和接收方实体之间使用了两个TCP连接(每个方向使用一个连接)而不是使用单独的双向连接时, 适用以下规则:

- 和初始方流相关的流级别错误由接收方实体在应答流中通过同一个TCP连接返回.
- 从初始方实体用同一个TCP连接通过初始流发送的出站节所触发的节错误(区别于流级别的错误), 接收方实体应通过另一个("返回")TCP连接的应答流中返回, 因为从初始方实体的角度来看这个(返回错误的)节是入站节.

语法

流错误的语法如下所示, 显示在中括号 '[' 和 ']' 的XML数据是可选的.

```
<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>
```

"defined-condition" 必须对应 4.9.3 中定义的流错误条件之一. 然而, 因为将来可能定义额外的错误条件, 如果实体接收到一个它不理解的流错误条件, 那么它必须把未知的条件当作 <undefined-condition/> (4.9.3.21). 如果一个XMPP扩展的设计者或XMPP实现的开发者需要使用未在本协议中定义的流错误条件来通讯, 他们可以定义一个由应用层命名空间限定的应用特有的错误条件元素来达到这个目的.

<error/>元素:

- 必须包含一个对应已定义错误条件之一的子元素; 这个元素必须由 'urn:ietf:params:xml:ns:xmpp-streams' 命名空间限定.
- 可以包含一个内有XML字符串数据的 <text/> 子元素用来描述错误细节; 这个元素必须由 'urn:ietf:params:xml:ns:xmpp-streams' 命名空间限定并且应该拥有一个 'xml:lang' 属性来指定XML字符串数据的自然语言.

- 可以包含一个子元素用于应用特有的错误条件; 这个元素必须由一个应用定义的命名空间来限定，并且它的结构也由该命名空间来定义 (见 4.9.4).

<text/> 元素是可选的. 如果有它, 它必须只用于提供描述性或诊断性的信息，用来补充一个已定义的条件或应用特有的条件的含义. 它不能(MUST NOT)被应用程序解释执行. 它不能(MUST NOT)被用作向自然人用户展示的错误消息, 但是可以被用作和已定义条件元素(以及, 可选地, 应用特有的条件元素)相关的错误消息的额外信息.

已定义的流错误条件

以下是已定义的流级别的错误条件.

bad-format

实体发送了无法处理的XML.

(在以下例子中, 客户端发送了一个非XML的XMPP消息, 它也可能会触发一个 <not-well-formed/> 流错误(4.9.3.13).)

```
C: <message>
     <body>No closing tag!
   </message>

S: <stream:error>
     <bad-format
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

> 这个错误可以用更多特定的XML相关的错误来替代, 例如 <bad-namespace-prefix/>, <invalid-xml/>, <not-well-formed/>, <restricted-xml/>, 和 <unsupported-encoding/>. 无论如何, 建议使用更多特定的错误.

bad-namespace-prefix

实体发送了不被支持的命名空间前缀, 或在一个需要这样的前缀的元素中没有发送命名空间前缀 (见 11.2).

(在以下例子中, 客户端指定了一个命名空间前缀 "foobar" 用于XML流命名空间.)

```
C: <?xml version='1.0'?>
   <foobar:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xmlns='jabber:client'
       xmlns:foobar='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
```

```
  <stream:error>
    <bad-namespace-prefix
        xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  </stream:error>
  </stream:stream>
```

conflict

服务器要么 (1) 关闭这个实体现存的流(如果和现有流冲突的新流已经被初始化了), 要么 (2) 拒绝这个实体的新流(如果允许这个新的流将导致和现有的流冲突(例如, 服务器限制来自同一IP地址的连接数量或对于给定的域对只允许一个 服务器-服务器 流，以确保10.1所述的顺序处理)).

```
C: <?xml version='1.0'?>
  <stream:stream
      from='juliet@im.example.com'
      to='im.example.com'
      version='1.0'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
  <stream:stream
      from='im.example.com'
      id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
      to='juliet@im.example.com'
      version='1.0'
      xml:lang='en'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>
  <stream:error>
    <conflict
        xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  </stream:error>
  </stream:stream>
```

如果客户端收到一个<conflict/>流错误(4.9.3.3), 它尝试重新连接的时候绑定的资源不能(MUST NOT)和前一个会话的资源相同，而是必须选择一个不同的资源; 详见 第7章.

connection-timeout

如果一方有理由相信另一方永久地失去了通过某个流进行通讯的能力，它可以关闭这个流. 有很多办法可以查觉到对方丧失通讯能力, 类似 4.4 所述的空格符保持连接, 定义于 XEP-0199 的XMPP级的ping, 以及定义于XEP-0198 的XMPP流管理.

```
P: <stream:error>
    <connection-timeout
        xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  </stream:error>
  </stream:stream>
```

> 互操作性备注: RFC 3920指出，如果在一段时间内对端在某个流上没有产生任何流量，则使用<connection-timeout/>流错误(4.9.3.4). 那种行为已经不推荐了; 替代办法是, 只在已连接客户端或对端服务器不响应该流发送的数据时才使用该错误.

host-gone

在初始化流头中提供的'to'属性的值对应的完整合法域名(FQDN)不再由接收方实体提供服务.

(以下例子中, 当连接到"im.example.com"服务器时,对端指定了一个'to'地址"foo.im.example.com", 但是这个服务器不再为那个地址提供服务了.)

```
P: <?xml version='1.0'?>
   <stream:stream
       from='example.net'
       to='foo.im.example.com'
       version='1.0'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='g4qSvGvBxJ+xeAd7QKezOQJFFlw='
       to='example.net'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <host-gone
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

host-unknown

初始流头中提供的'to'属性的值不对应接收方实体所服务的合法域名(FQDN).

(在下例中, 对端连接到"im.example.com"服务器的时候指定了一个'to'地址"example.org", 但是该服务器不知道这个地址.)

```
P: <?xml version='1.0'?>
   <stream:stream
       from='example.net'
       to='example.org'
       version='1.0'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='g4qSvGvBxJ+xeAd7QKezOQJFFlw='
       to='example.net'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <host-unknown
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

improper-addressing

在两服务器之间发送的节缺少'to'或'from'属性, 这个'from'或'to'属性没有值, 或它的值违反了XMPP地址XMPP-ADDR的规则 .

(在下例中, 对端在服务器-服务器流中发送了一个不包含'to'地址的节.)

```
P: <message from='juliet@im.example.com'>
     <body>Wherefore art thou?</body>
   </message>

S: <stream:error>
     <improper-addressing
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

internal-server-error

服务器配置错误或其他内部错误导致它无法服务于该流.

```
S: <stream:error>
     <internal-server-error
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

invalid-from

当 (1) 在两个服务器使用SASL或服务器回拨, 或 (2) 在客户端和服务器通过SASL验证和资源绑定的时候,协商时的'from'属性提供的数据不匹配已授权的JID或合法的域名.

(在下例中, 一个仅被授权为"example.net"的对端尝试以地址"example.org"发送节.)

```
P: <message from='romeo@example.org' to='juliet@im.example.com'>
     <body>Neither, fair saint, if either thee dislike.</body>
   </message>

S: <stream:error>
     <invalid-from
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

invalid-namespace

流命名空间的名字不是"http://etherx.jabber.org/streams" (见 11.2) 或不支持把内容命名空间声明为缺省命名空间 (例如, 不同于"jabber:client"或"jabber:server").

(下例中, 客户端为流指定了一个命名空间'http://wrong.namespace.example.org/'.)

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
```

```
      version='1.0'
      xmlns='jabber:client'
      xmlns:stream='http://wrong.namespace.example.org/'>

S: <?xml version='1.0'?>
   <stream:stream
      from='im.example.com'
      id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
      to='juliet@im.example.com'
      version='1.0'
      xml:lang='en'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <invalid-namespace
        xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

invalid-xml

实体通过该流发送了非法的XML到一个执行验证的服务器上(见 11.4).

(下例中, 对端尝试发送一个类型为"subscribe"的IQ节, 但是XML schema没有这个'类型'的属性.)

```
P: <iq from='example.net'
      id='l3b1vs75'
      to='im.example.com'
      type='subscribe'>
     <ping xmlns='urn:xmpp:ping'/>
   </iq>

S: <stream:error>
     <invalid-xml
        xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

not-authorized

实体尝试在流被验证之前发送XML节或其他出站数据, 或没有被授权执行一个和流协商有关的动作; 接收方实体在发送流错误之前不能(MUST NOT)处理这些数据.

(下例中, 客户端尝试在被服务器验证前发送XML节.)

```
C: <?xml version='1.0'?>
   <stream:stream
      from='juliet@im.example.com'
      to='im.example.com'
      version='1.0'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
      from='im.example.com'
      id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
      to='juliet@im.example.com'
      version='1.0'
      xml:lang='en'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'>
```

```
C: <message to='romeo@example.net'>
     <body>Wherefore art thou?</body>
   </message>

S: <stream:error>
     <not-authorized
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

not-well-formed

初始化实体发送了违反XML或XML-NAMES的"良好格式"规则的XML.

(下例中, 客户端发送一个命名空间格式错误的XMPP消息.)

```
C: <message>
     <foo:body>What is this foo?</foo:body>
   </message>

S: <stream:error>
     <not-well-formed
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

> 互操作性备注: 在RFC 3920中, 这个错误条件的名字是 "xml-not-well-formed" 而不是 "not-well-formed". 改名是因为元素名 <xml-not-well-formed/> 违反了XML第三章的约束 "以 (('X'|'x')('M'|'m')('L'|'l')) 打头的名字是保留给本协议的这个或下个版本的标准化用的".

policy-violation

实体违反一些本地服务策略 (例如, 一个节超出了配置的大小限制); 服务器可以选择在 <text/> 元素里或在一个应用特有的条件元素中指定这个策略.

(下例中, 客户端发送了一个据服务器的本地服务策略看来过大的XMPP消息.)

```
C: <message to='juliet@im.example.com' id='foo'>
     <body>[ ... the-emacs-manual ... ]</body>
   </message>

S: <stream:error>
     <policy-violation
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
     <stanza-too-big xmlns='urn:xmpp:errors'/>
   </stream:error>

S: </stream:stream>
```

remote-connection-failed

服务器不能正确地连接到一个需要验证或授权的远程实体 (例如, 在和服务器回拨XEP-0220相关的特定场景); 当发生这个错误的是XMPP服务提供商有管理员权限的域时候, 这个条件不被使用, 那种情况下更适合使用 <internal-server-error/> 条件.

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <remote-connection-failed
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

reset

服务器在下列情况下关闭流，如提供了新的流特性(特别是关系安全的), 如为流建立安全上下文的密钥或者证书过期或在流的声明周期中被收回([[RFC6120#检查长连接流的证书|13.7.2.3), 如TLS序列号已经封装了(5.3.5), 等等. reset适用于流以及该流(例如, 通过TLS和 SASL)建立的的任何安全上下文, 这意味着新的流的加密和验证需要再次协商(例如, 不能使用TLS会话恢复了).

```
S: <stream:error>
     <reset
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

resource-constraint

服务器缺乏必要系统资源来服务于这个流.

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <resource-constraint
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
```

```
   </stream:error>
   </stream:stream>
```

restricted-xml

实体尝试发送受限的XML特性，例如 注释, 处理指令, DTD子集, 或XML实体参考(见 11.1).

(下例中, 客户端发送一个包含XML注释的XMPP消息.)

```
C: <message to='juliet@im.example.com'>
     <!--<subject/>-->
     <body>This message has no subject.</body>
   </message>

S: <stream:error>
     <restricted-xml
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

see-other-host

服务器将不提供服务给初始化实体，但是重定向到同一个服务提供商管理控制下的另一台主机. 服务器返回的 <see-other-host/> 元素的XML字符串数据必须指定用来连接的替代的合法域名(FQDN)或IP地址, 它必须是一个合法的域部分或一个域部分加上一个端口号(通过"域名:端口号"中的':'字符来区分). 如果域部分和源域相同,或和派生域相同, 或解析出来的IPv4或IPv6地址和初始化实体原先连接的地址相同(只是端口号不同), 那么初始化实体应该简单地重新连接那个地址. (IPv6地址的格式必须遵循 IPv6-ADDR, 它包括如 URI 所定义的，把IPv6地址封闭在方括号'[' 和 ']'里.) 否则, 初始化实体必须解析 <see-other-host/>元素中指定的合格域名(FQDN),如 **解析合格域名3.2**所述.

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <see-other-host
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'>
       [2001:41D0:1:A49b::1]:9222
     </see-other-host>
   </stream:error>
   </stream:stream>
```

当协商一个已经被重定向的流时, 初始化实体必须应用和它应用于初次连接时相同的策略 (例如, 一个必须使用TLS的策略), 必须在初始化流头中指定相同的'to'地址, 而且必须使用和初次尝试连接时相同的参考标识符来检查新地址的标识符 (符合 TLS-CERTS). 即使接收方在流的保密和信任关系建立之前返回

一个<see-other-host/>错误(从而产生 拒绝服务 攻击的可能性), 事实上初始化实体需要基于相同的参考标识符来检查XMPP服务的标识符， 这意味着初始化实体将不会连接到一个恶意的实体. 为了避免 拒绝服务 攻击, (a) 在流的保密和信任关系由TLS或相当的安全层(例如 SASL GSSAPI 机制)保护起来立之前, 接收方实体不应该以<see-other-host/>流错误关闭流, 并且 (b) 只有当它已经被接收方实体验证了, 接收方才可以有一个下述的重定向策略. 另外, 初始化实体在特定数量的成功重定向之后应该放弃尝试连接(例如, 最少2次但不超过5次).

system-shutdown

服务器正在关闭且所有活跃的流正在被关闭.

```
S: <stream:error>
     <system-shutdown
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

undefined-condition

这个错误条件不是预定义的条件列表中的一个; 这个错误应该不被使用， 除非结合一个应用特有的条件.

```
S: <stream:error>
     <undefined-condition
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
     <app-error xmlns='http://example.org/ns'/>
   </stream:error>
   </stream:stream>
```

unsupported-encoding

初始化实体给流使用的编码不被服务器支持(见 11.6) 或没有正确地对流进行编码 (例如, 违反了 UTF-8 编码规则).

(下例中, 客户端试图使用UTF-16编码而不是UTF-8.)

```
C: <?xml version='1.0' encoding='UTF-16'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <unsupported-encoding
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
```

```
   </stream:error>
   </stream:stream>
```

unsupported-feature

接收方实体声明了一个 强制协商 的流特性，但是初始化实体不支持它, 并且提不出其他等价于不支持特性的 强制协商 特性.

(下例中, 接收方实体要求一个example特性, 但是初始化实体不支持这个特性.)

```
Rs: <stream:features>
     <example xmlns='urn:xmpp:example'>
       <required/>
     </example>
   </stream:features>

I: <stream:error>
     <unsupported-feature
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

unsupported-stanza-type

初始化实体发送的流的顶级子元素不被服务器支持, 要么是因为接收方实体不理解命名空间要么因为接收方实体不理解适用的命名空间的元素名(可能是声明为缺省命名空间的内容命名空间).

(下例中, 客户端尝试发送一个由'jabber:client'命名空间限定的顶级子元素 <pubsub/> , 但是那个命名空间的schema没有定义这个元素.)

```
C: <pubsub xmlns='jabber:client'>
     <publish node='princely_musings'>
       <item id='ae890ac52d0df67ed7cfdf51b644e901'>
         <entry xmlns='http://www.w3.org/2005/Atom'>
           <title>Soliloquy</title>
           <summary>
To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them?
           </summary>
           <link rel='alternate' type='text/html'
                 href='http://denmark.example/2003/12/13/atom03'/>
           <id>tag:denmark.example,2003:entry-32397</id>
           <published>2003-12-13T18:30:02Z</published>
           <updated>2003-12-13T18:30:02Z</updated>
         </entry>
       </item>
     </publish>
   </pubsub>

S: <stream:error>
     <unsupported-stanza-type
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

unsupported-version

由初始化实体在流头中提供的'version'属性指定的XMPP版本不被服务器支持.

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='11.0'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>
   <stream:error>
     <unsupported-version
         xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
   </stream:error>
   </stream:stream>
```

应用特有的条件

大家知道, 应用可以在错误元素中包含一个适当的命名空间子元素来提供应用特有的错误信息. 应用特有的元素应该补充或进一步限定一个已定义的元素. 因此, <error/> 元素将包含两个或三个子元素.

```
C: <message>
     <body>
       My keyboard layout is:

       QWERTYUIOP{}|
       ASDFGHJKL:"
       ZXCVBNM<>?
     </body>
   </message>

S: <stream:error>
     <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='http://example.org/ns'/>
   </stream:error>
   </stream:stream>
```

简化的流示例

这一仗包含两个客户端和服务器之间基于流的连接的高度简化的例子; 这些例子的目的是说明迄今为止介绍的那些概念, 但是读者需要注意这些例子省略了一些细节 (更完整的例子见 第9章).

一个基本的连接:

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

[ ... stream negotiation ... ]

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

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

C: </stream:stream>

S: </stream:stream>
```

连接坏了:

```
C: <?xml version='1.0'?>
   <stream:stream
       from='juliet@im.example.com'
       to='im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

S: <?xml version='1.0'?>
   <stream:stream
       from='im.example.com'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

[ ... stream negotiation ... ]

C:   <message from='juliet@im.example.com/balcony'
              to='romeo@example.net'
              xml:lang='en'>
       <body>No closing tag!
     </message>

S: <stream:error>
    <not-well-formed
```

```
      xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  </stream:error>
  </stream:stream>
```

更多详细的例子请到第9章 .

STARTTLS协商

STARTTLS基础

XMPP包含了一个方法来保护流的安全使其免于被篡改和窃听. 这个通道加密的方法使用传输层安全 TLS 协议, 特别是"STARTTLS"扩展， 这个扩展是以 USINGTLS中描述的 IMAP, POP3, 和 ACAP 协议中的类似扩展为蓝本的. STARTTLS扩展的XML命名空间是 'urn:ietf:params:xml:ns:xmpp-tls'.

支持

在XMPP客户端和服务器的实现中必须支持STARTTLS. 一个给定布署的管理员可以指定 客户端-服务器通讯 和/或 服务器-服务器通讯 中TLS是强制协商的. 一个初始化实体应该在开始SASL验证之前使用TLS保护和接收方之间的流的安全.

流协商规则

强制协商

如果接收方实体只声明了STARTTLS特性或接收方实体包含了5.4.1所述的<required/>子元素, 双方必须确保TLS是强制协商的. 如果TLS是强制协商的, 在流协商过程的初始化阶段接收方实体应该不(SHOULD NOT)声明支持任何STARTTLS以外的流特性, 因为在XMPP的层顺序中更多流特性可能依赖TLS的预先协商 (例如, 由接收方实体提供的特定的SASL机制将依赖于TLS是否完成协商).

重启

在TLS协商之后, 双方必须重启这个流.

数据格式

当STARTTLS协商时, 实体们不能(MUST NOT)在XML元素之间发送任何空格符号 (即, 由初始化实体发送的从'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的顶级<starttls/>元素的最后的字符, 到由接收方实体发送的'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的顶级<proceed/>元素的最后的字符). 这个禁令帮助确保适当的安全字节精度. 任何出现在本文提供的STARTTLS例子中的空格只是为了提高可读性.

TLS和SASL协商的顺序

如果初始化实体选择使用TLS, STARTTLS协商必须在SASL协商之前完成; 这个协商顺序对于帮助保护SASL协商期间发送的验证信息是必要的, 同时尽可能使用基于预先的TLS协商中提供的证书(或其他证件)的SASL EXTERNAL机制.

TLS重协商

TLS协议允许双方在一个 受TLS保护 的通道里初始化一个新的握手来建立新的加密参数(见 TLS-NEG). 最常提及的案例如下:

1. 刷新密钥
2. 如TLS的6.1节所述封装TLS序列号.
3. 在受保护的通道上先完成服务器验证再完成客户端验证以保护客户端证书.

因为在XMPP中建立一个流的代价相对低廉, 对于前面两个案例推荐使用XMPP流重置(如 4.9.3.16) 而不是执行TLS重协商.

第三个案例在TLS客户端(也可能是一个XMPP服务器)递交TLS证书给服务器时提高了安全特性. 如果和一个未验证的TLS服务器交换这类证书可能泄露隐私信息, 先完成让TLS客户端验证TLS服务器的TLS协商，再完成让TLS服务器验证TLS客户端的TLS协商，是适当的. 然而, 这个案例极为罕见，因为由一个扮演TLS客户端角色的XMPP服务器或XMPP客户端对外展现的证书几乎总是公开的(即, PKIX 证书), 所以在验证作为TLS服务器的XMPP服务器之前提供那些证书通常将不会泄露隐私信息.

作为结果, 鼓励实现者在他们的软件中支持TLS重协商之前小心地权衡它的开销和好处, 不鼓励扮演TLS客户端的XMPP实体尝试TLS重协商，除非已知要在TLS协商中发送的证书(或其他证件信息)是私有的.

对TLS重协商的支持是严格可选的. 然而, 支持TLS重协商的实现们必须实现和使用 TLS重协商扩展 TLS-NEG.

如果一个不支持TLS重协商的实体察觉到一个重协商尝试, 那么它必须立刻关闭相关的TCP连接而不要返回任何流错误(因为这个违规可能发生在TLS层, 而不是XMPP层, 详见 13.3).

如果一个支持TLS重协商的实体察觉到一个未使用 TLS重协商扩展 TLS-NEG 的TLS重协商尝试 , 那么它必须立刻关闭相关的TCP连接而不要返回任何流错误(因为这个违规可能发生在TLS层, 而不是XMPP层, 详见 13.3).

TLS扩展

一个流的双方可以在它自己的TLS协商时包含任何TLS扩展. 这是TLS层的事情, 不是XMPP层.

过程

流头和流特性交换

初始化实体如第三章所述解析接收实体的合格域名(FQDN), 打开一个到解析的IP地址和声明的端口的TCP连接, 并发送一个初始化流头给接收方流头.

```
I: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

接收方实体必须通过初始化实体打开的那个TCP连接发送一个应答流头给初始化实体.

```
R: <stream:stream
     from='im.example.com'
     id='t7AMCin9zjMNwQKDnplntZPIDEI='
     to='juliet@im.example.com'
```

```
    version='1.0'
    xml:lang='en'
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'>
```

接着接收方实体必须发送流特性给初始化实体. 如果接收方实体支持TLS, 流特性必须包含一个支持STARTTLS协商的声明, 即, 一个由'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的<starttls/>元素.

如果接收方实体认为STARTTLS协商是强制协商的, <starttls/>元素必须包含一个空的<required/>子元素.

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

STARTTLS协商的初始化

STARTTLS命令

为了开始STARTTLS协商, 初始化实体发出STARTTLS指令(即, 一个由'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的<starttls/>元素)来指示接收方实体它希望开始一次STARTTLS协商以保护流.

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

接收方实体必须以由'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的<proceed/>元素(继续进行的情况下)或<failure/>元素(失败的情况下)回复.

失败的情况

如果发生失败的情况, 接收方实体必须返回一个由'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的<failure/>元素, 关闭这个XML流, 并且终止当前的TCP连接.

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

R: </stream:stream>
```

导致失败的情况包含但不限于以下几种:

1. 初始化实体发送了一个异常的STARTTLS命令.
2. 接收方实体在它的流特性中不提供STARTTLS特性 .
3. 接收方实体因为内部错误而无法完成STARTTLS协商.

参考文献: STARTTLS失败不会由TLS错误触发，例如 坏证书(bad_certificate)或 握手失败(handshake_failure), 它是由TLS协商本身生成和处理的，参见TLS.

如果发生了失败的情况, 初始化实体可以尝试重连，参见 3.3.

继续进行的情况

如果发生继续进行的情况, 接收方实体必须返回一个由'urn:ietf:params:xml:ns:xmpp-tls'命名空间限定的<proceed/>元素.

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

接收方实体在发送了<proceed/>元素的关闭字符'>'之后必须认为TLS协商已经立刻开始了. 初始化实体在从接收方实体接收到<proceed/>元素的关闭字符'>'之后必须认为TLS协商已经立刻开始了.

实体现在继续进行TLS协商， 如下一节所述.

TLS协商

规则

为了在TCP连接上完成TLS协商, 实体们必须跟随以下定义于TLS的过程 .

以下规则适用于:

1. 实体们在TLS协商完成之前不能(MUST NOT)发送任何其他XML数据.
2. 当使用定义于13.8的任何强制实现(MTI)的密码组时, 接收方实体必须出示一个证书.
3. 所以证书相互验证是有可能的, 接收方实体应该发送一个证书请求给初始化实体, 而初始化实体应该发送一个证书给接收方实体(但是由于隐私的原因可能选择在接收方实体已经被初始化实体验证之后才发送自己的证书).
4. 接收方实体应该基于包含在初始化流头中的'to'属性中的域部分选择哪个证书来展示(本质上, 这个域部分功能上等同于服务器名称指示 定义于TLS-EXT).
5. 为了确定TLS协商是否成功, 初始化实体必须尝试根据13.7.2定义的证书验证程序来验证接收方实体的证书.
6. 如果初始化实体出示了一个证书, 接收方实体也必须尝试根据13.7.2定义的证书验证程序来验证初始化实体的证书.
7. 随着TLS协商成功, 所有双方传送的更多的数据必须被协商的算法， 密钥和秘密来保护(即, 加密, 完整性保护, 或都依赖于使用的密码组).

安全警告: 关于13.8提到的密码组必须被TLS所支持; 自然的, 其他密码组也可以被支持.

TLS失败

如果TLS协商结果是失败, 接收方实体必须终止该TCP连接.

在终止该TCP连接之前,接收方实体不能(MUST NOT)发送关闭标签</stream>(因为失败可能发生在TLS层, 而不是XMPP层， 参见13.3所述).

初始化实体可以如3.3所述尝试重连, 尝试使用或不使用TLS协商(依照本地服务策略, 用户配置的偏好, 等等).

TLS成功

如果TLS协商是成功的, 那么实体们必须继续如下步骤.

1. 初始化实体必须忽略TLS生效之前在TCP上的不安全情况下从接收方实体收到的任何信息(例如, 接收方实体的'from'地址或从接收方实体收到的流ID以及流特性).
2. 接收方实体必须忽略TLS生效之前在TCP上的不安全情况下从初始化实体收到的任何信息(例如, 初始化实体的'from'地址).
3. 初始化实体必须通过加密的连接发送一个新的初始化流头给接收方实体(如4.3.3定义的, 在发送新的初始化流头之前初始化实体必须发送一个关闭标签</stream>, 因为接收方实体和初始化实体必须确定旧的流在TLS协商成功之后被替代了).

```
I: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

4. 接收方实体必须通过加密连接以一个新的应答流头来应答(为此它必须生成一个新的流ID而不是重用旧的流ID).

```
R: <stream:stream
     from='im.example.com'
     id='vgKi/bkYME8OAj4rlXMkpucAqe4='
     to='juliet@im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

5. 接收方实体也必须发送流特性给初始化实体, 不能(MUST NOT)包含STARTTLS特性但是应该包含SASL流特性, 如第六章(特别是6.4.1中关于为什么不在这里提供SASL流特性的新的原因).

```
R: <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>EXTERNAL</mechanism>
       <mechanism>SCRAM-SHA-1-PLUS</mechanism>
       <mechanism>SCRAM-SHA-1</mechanism>
       <mechanism>PLAIN</mechanism>
     </mechanisms>
   </stream:features>
```

SASL协商

SASL基础

XMPP包含了一个某种意义上XMPP特有的简单验证和安全层协议(见SASL)用于验证一个流. SASL提供一个一般化的方法来给基于连接的协议添加验证支持, 而XMPP遵照SASL的解析要求来使用SASL的XML命名空间解析. SASL扩展的XML命名空间名称是'urn:ietf:params:xml:ns:xmpp-sasl'.

支持

XMPP客户端和服务器必须支持SASL协商.

流协商规则

强制协商

一个流双方bxu确认SASL是强制协商的.

重启

在SASL协商之后, 双方必须重启该流.

机制推荐

任何将要扮演SASL客户端或SASL服务器的实体必须对于该客户端或该服务器维护一个它推荐的SASL机制的有序列表, 这个列表的顺序是根据本地策略或用户配置来的(它的顺序应该是根据验证能力越强排在越靠前). 初始化实体必须独立于接收方实体的推荐顺序来维护它自己的推荐顺序. 客户端必须以它自己的推荐顺序来尝试SASL机制. 例如, 如果服务器提供的顺序列表是"PLAIN SCRAM-SHA-1 GSSAPI" 或 "SCRAM-SHA-1 GSSAPI PLAIN" 而客户端的顺序列表是 "GSSAPI SCRAM-SHA-1", 客户端必须首先尝试 GSSAPI 然后尝试 SCRAM-SHA-1 而不能(MUST NOT)尝试 PLAIN (因为 PLAIN 不在它的列表中).

机制提供

如果接收方实体在它接受特定的SASL机制之前确定TLS协商是强制协商, 它不能(MUST NOT)在完成TLS协商之前在它的可用SASL机制列表中声明那个机制.

如果发生以下两种情况，接收方实体应该提供 SASL EXTERNAL 机制，:

1. 当初始化实体在TLS协商中出示了一个证书，这个证书被接收方实体接受了，接收方根据本地服务策略把它用于强身份验证(例如, 因为证书没有过期，没有撤销，并且被锚定到一个接收方实体信任的root账户).
2. 接收方实体期望初始化实体能够验证和授权这个证书所提供的身份; 在服务器-服务器流的情形下, 接收方实体可能有这样一个预期，因为初始化实体的证书所展示的DNS域名和初始化流头中相应的'from'属性是匹配的, 这里使用TLS-CERTS的匹配规则; 在客户端-服务器流的情形下, 接收方实体可能有这样一个预期，是因为在初始化实体的证书中展示的纯JID和在这个服务器上注册的一个用户帐号匹配，或者因为其他包含在初始化实体证书中的信息和被允许使用该服务器访问XMPP网络的某个实体相匹配.

无论如何, 接收方实体在其他情况下也一样可以提供 SASL EXTERNAL 机制.

当接收方实体提供 SASL EXTERNAL 机制, 接收方实体应该首先列出它提供的SASL机制的 EXTERNAL 机制列表，而初始化实体应该尝试首先使用EXTERNAL机制来进行SASL协商(这个选择往往会增加双方相互进行证书验证的可能性).

13.8定义了必须支持的SASL机制; 自然的, 也一样可以支持其他的SASL机制.

> 参考文献: 在XMPP的上下文中使用SASL的最佳实践，使用ANONYMOUS机制请参考XEP-0175，使用EXTERNAL机制请参考XEP-0178.

数据格式

以下数据格式规则适用于SASL协商:

1. 当SASL协商时, 实体不能(MUST NOT)在XML元素之间发送任何空格符号(即, 从初始化实体发送的'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<auth/>顶级元素的最后一个字符, 到接收方实体发送的'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<success/>顶级元素的最后一个字符). 这个禁令帮助确保正确的安全层字节精确度. 本文中任何在SASL例子中出现的这类空格只是为了增加可读性.
2. 任何包含在XML元素中的XML字符串数据必须使用base 64编码, 这里的编码要坚持BASE64第四章的定义,并且填充位设为零.
3. 作为附录A.4下的XML schema中正式指定的 'urn:ietf:params:xml:ns:xmpp-sasl' 命名空间, 接收方实体可以在<mechanisms/>元素中包含一个或多个应用特有的子元素来提供初始化实体使用提供的一个或多个机制进行成功的SASL协商而可能需要的信息; 无论如何, 所有这类元素的语法和语义超出了本协议的范围(见XEP-0233的例子).

安全层

涉及安全层协商的SASL协商成功后, 初始化实体和接收方实体都必须丢弃任何应用层状态(即, 来自XMPP层的状态, 不包括来自TLS协商或SASL协商的状态).

简单用户名

一些SASL机制(例如, CRAM-MD5, DIGEST-MD5, 和 SCRAM) 指定了在这些机制的上下文中使用的验证身份是一个"简单用户名" (见 SASL 的第二章以及 SASLPREP). 在任何特定的机制或部署中简单用户名的准确格式都是一个本地事务, 并且简单用户名不需要映射到一个应用身份例如JID或JID部件(例如, 本地部分). 无论如何, 在缺乏由服务器提供的本地信息的情况下, 一个XMPP客户端应该假定一个SASL机制的验证身份是等于该用户的JID的本地部分的简单用户名.

授权身份

授权身份是一个由初始化实体提供的可选的身份， 用来定义它扮演的身份(见SASL第二章). 在客户端-服务器流中, 它大部分被管理员用于代表另一个用户来执行一些管理任务, 而在服务器-服务器流它大部分被用于在XMPP服务上指定一个特定的附加服务(例如, 一个多用户聊天服务器conference.example.com寄宿在example.com的XMPP服务上). 如果初始化实体希望代表另一个实体并且所选择的SASL机制支持授权身份的传输, 该初始化实体必须在SASL协商时提供一个授权身份. 如果初始化实体不希望代表另一个实体的身份, 它不能(MUST NOT)提供授权身份.

在客户端-服务器通讯的情况下, 授权身份的值必须是一个纯JID(<本地部分@域部分>) 而不是一个全JID(<本地部分@域部分/资源部分>).

在服务器-服务器通讯的情况下, 授权身份的值必须且只能是一个域部分(<域部分>).

如果初始化实体在SASL协商时提供一个授权身份, 接收方实体负责验证初始化实体是否事实上被允许承担指定的授权身份; 如果不是, 接收方实体必须返回一个<invalid-authzid/> SASL错误， 如 6.5.6 所述.

领域

在以特定SASL机制协商的时候接受方实体可以包含一个领域(例如, GSSAPI 和 DIGEST-MD5 机制都允许验证交换的信息中包含领域, 不过其他方式, 如 EXTERNAL, SCRAM, 和 PLAIN 机制不支持这个特性). 如果接受方实体不以一个领域来通讯, 则初始化实体不能(MUST NOT)假定任何领域的存在. 领域必须只被用于验证的目的; 特别是, 初始化实体不能(MUST NOT)尝试从接受方实体提供的领域信息来派生出一个XMPP域部分.

回合

SASL 规定，一个使用中的协议(例如XMPP)可以定义两个方法，这样协议可以节省批准SASL机制的回合:

1. 当SASL客户端(XMPP "初始化实体") 请求一个验证交换时, 如果使用了适当的SASL机制, 它可以在它的请求中包含 "初始化应答" 数据. 在XMPP中, 要实现这一点，就把初始化应答作为XML字符串数据包含在<auth/>元素中.
2. 在验证交换的结尾, 如果使用了适当的SASL机制，SASL服务器(XMPP "接受方实体") 可以包含 "成功附带的额外数据". 在XMPP中, 要实现这一点，就把额外数据作为XML字符串数据包含在<success/>元素中.

为了协议的效率, 要求客户端和服务器必须支持这些方法并且建议使用它们; 无论如何, 客户端和服务器也必须支持低效的模式.

过程

SASL协商过程如下.

流头和流特性交换

如果SASL协商紧跟在成功的STARTTLS协商之后, 那么SASL协商发生在已经协商过的受保护的流上. 否则, 初始化实体如第三章所述解析接受方实体的完整域名(FQDN), 打开一个到已解析的IP地址的已声明的端口的TCP连接, 并发送一个初始化流头给接受方实体. 在两种情况下, 接受方实体都将从初始化实体接收到一个初始化流.

```
I: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

当接受方实体处理来自初始化实体的初始化流的时候, 它必须发送一个应答流头给初始化实体(为此它必须生成一个唯一的流ID. 如果TLS协商已经成功, 那么这个流ID必须不同于TLS协商成功之前发送的那个流ID).

```
R: <stream:stream
     from='im.example.com'
     id='vgKi/bkYME8OAj4rlXMkpucAqe4='
     to='juliet@im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

接受方实体也必须发送流特性给初始化实体. 流特性应该包含一个声明用来支持SASL协商, 即, 一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<mechanisms/>元素. 典型的，只有三种情况下对SASL协商的支持不需要在这里声明:

- TLS协商需要在提供SASL之前发生(即, TLS是必需的并且接受方实体已经在接收到的这次连接尝试的初次的初始化流头中应答过了).
- SASL协商不可能发生在一个 服务器-服务器 连接中(即, 初始化服务器未提供一个证书以进行验证并且因而接受方实体回滚到使用服务器回拨协议XEP-0220进行弱身份验证).
- SASL已经协商过了(即, 接受方实体在成功进行SASL协商之后应答一个以流重启的方式发送的初始化流头).

<mechanisms/>元素必须为接受方实体提供给初始化实体的每个验证机制包含一个<mechanism/>子元素. 大家知道, 在XML中的<mechanism/>元素的顺序表示来自接受方实体的SASL机制的优先顺序(它不一定是来自初始化实体的优先顺序).

```
R: <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>EXTERNAL</mechanism>
       <mechanism>SCRAM-SHA-1-PLUS</mechanism>
       <mechanism>SCRAM-SHA-1</mechanism>
       <mechanism>PLAIN</mechanism>
     </mechanisms>
   </stream:features>
```

初始化

为了开始SASL协商, 初始化实体发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<auth/>元素并在'mechanism'属性包含一个适当的值, 从而开始使用特定的验证机制进行握手. 这个元素可以包含XML字符串数据(用SASL术语来说, 就是"初始化应答"), 如果这个机制支持或必须要它的话. 如果初始化实体需要发送一个长度为零的初始化应答, 它必须以单个等号字符("=")来传输这个应答, 这表示那个应答是当前的但是不包含数据.

```
I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='PLAIN'>AGp1bGlldAByMG0zMG15cjBtMzA=</auth>
```

如果初始化实体后来发送另一个<auth/>元素而正在进行的验证握手还没完成, 接收方实体必须丢弃正在进行的握手而必须为后来请求的SASL机制处理新的握手.

挑战-应答序列

如果必要, 接收方实体通过发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<challenge/>元素来挑战初始化实体; 这个元素可以包含XML字符串数据(它必须根据被初始化实体选择的SASL机制的定义来生成).

初始化实体通过发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<response/>元素来应答这个挑战; 这个元素可以包含XML字符串数据(它必须根据被初始化实体选择的SASL机制的定义来生成).

如果必要y, 接收方实体发送更多挑战而初始化实体发送更多应答.

这一系列的 挑战/应答 对 一直持续直到发生以下三件事情之一:

- 初始化实体退出这个验证机制的握手.
- 接收方实体报告握手失败.
- 接收方实体报告握手成功.

这些场景具体描述在接下来的章节.

放弃

初始化实体通过发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<abort/>元素放弃为这个验证机制所做的握手.

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

在接收到一个<abort/>元素之后, 接收方实体必须返回一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<failure/>元素并在其中包含一个<aborted/>子元素.

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

SASL失败

接收方实体通过发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<failure/>元素来汇报这个验证机制握手失败(特定的失败原因必须放进<failure/>元素的适当子元素，如 6.5定义的).

```
R: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     <not-authorized/>
   </failure>
```

为了选择适当的SASL机制, 接收方实体应该允许一个可配置的但是合理的重试次数(至少2次但不超过5次); 这让初始化实体能(例如, 一个终端用户客户端)容忍不正确的凭证(例如, 一个输错的密码)而不要强制重新连接(如果接收方实体立刻返回SASL失败并关闭流).

如果初始化实体对同一个SASL机制尝试了合理的重试次数并且都失败了, 它可以回滚到顺序列表中的下一个机制，只要发送一个新的<auth/>请求给接收方实体, 从而开始那个机制的新的握手. 如果所有握手都失败了并且在初始化实体支持和可接受的机制列表里没有剩余的机制了, 初始化实体应该简单地关闭这个流，如4.4所述(而不是等待这个流超时).

如果初始化实体超出了重试次数, 接收方实体必须以一个流错误关闭流, 它应该是<policy-violation/>(4.9.3.14), 不过一些现有的实现发送的是<not-authorized/>(4.9.3.12).

> 实现备注: 对于 服务器-服务器 流, 如果接收方实体不能提供SASL EXTERNAL机制或其他基于TLS协商期间简历的安全上下文的SASL机制, 接收方实体可以尝试使用服务器回拨协议XEP-0220来完成弱身份验证; 无论如何, 如果根据本地策略弱身份验证不足够的话，那么接收方实体应该以<policy-violation/>流错误(4.9.3.14)关闭这个流而不是等待这个流超时.

SASL成功

在确定SASL握手成功之前, 如果初始化实体在一个其保密和诚信得到TLS或同等的安全层(例如SASL GSSAPI机制)保护的初始化流头提供了一个'from'属性，那么接收方实体应该把这个验证身份结果关联到来自SASL协商的'from'地址; 如果这两个身份不匹配，那么接收方实体应该终止连接尝试(然而, 接收

方实体可以有合法的理由不终止这个连接尝试, 例如, 因为它覆盖了一个连接的客户端的地址来纠正JID格式或根据终端用户的证书授予一个JID).

接收方实体通过发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<success/>元素来汇报握手成功; 这个元素可以包含XML字符串数据(在SASL 属于中, 是"成功的附加数据"), 如果选择的SASL机制支持或者要求它. 如果接收方实体需要发送零长度的附加数据, 它必须传送一个单独的等号字符("=")数据.

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

> 参考文献: 对于 客户端-服务器 流, 在SASL协商时通讯的授权身份是用来根据接收方服务器为初始化客户端确定的权威地址, 如4.3.6所述.

一旦接收到<success/>元素, 初始化实体必须在现有的TCP连接上发送一个新的初始化流头到接收方实体来初始化一个新的流(如4.3.3所述, 在发送新的初始化流头之前,初始化实体不能(MUST NOT)发送一个关闭</stream>标签, 因为接收方实体和初始化实体必须确定原始的流被替换成SASL协商成功之后的流).

```
I: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

一旦从初始化实体接收到新的初始化流头, 接收方实体必须发送一个新的流头给初始化实体来应答(为此它必须生成一个新的流ID而不是重用旧的流ID).

```
R: <stream:stream
     from='im.example.com'
     id='gPybzaOzBmaADgxKXu9UClbprp0='
     to='juliet@im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

接收方实体也必须发送流特性, 包含任何更多的可用特性或不包含特性(通过一个空的<features/>元素).

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

SASL错误

SASL错误的语法如下, 那些用方括号 '[' 和 ']' 括起来的XML数据是可选的.

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <defined-condition/>
  [<text xml:lang='langcode'>
      OPTIONAL descriptive text
  </text>]
</failure>
```

"defined-condition" 必须是在接下来的章节里的定义的SASL相关的错误条件之一. 然而, 因为将来可能定义额外的错误条件, 如果一个实体收到一个它不理解的SASL错误条件，那么它必须把这个未知的条件视为一个通用的验证错误, 即, 等同于 <not-authorized/> (6.5.10).

内含的<text/>元素是可选的, 并且可被用于提供关于这个错误条件的应用特有的信息, 这个信息可以显示给人看但只是作为已定义的条件的补充.

因为XMPP本身定义了一个SASL应用范本并且不期望有更多专门的XMPP应用建立在SASL之上, 所以SASL错误格式不会像在XML流(4.9.4)和XML节(8.3.4)的做法一样，为应用特有的错误提供扩展性.

aborted

接收方实体确认验证握手已经被初始化实体放弃; 对<abort/>元素发送应答.

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

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

account-disabled

初始化实体的帐号已经被暂时禁用; 对<auth/>元素或<response/>元素发送应答(可以包含或不包含初始化应答数据).

```
I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='PLAIN'>AGp1bGlldAByMG0zMG15cjBtMzA=</auth>

R: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     <account-disabled/>
     <text xml:lang='en'>Call 212-555-1212 for assistance.</text>
   </failure>
```

credentials-expired

因为初始化实体提供的证书过期而验证失败; 对<response/>元素或<auth/>元素发送包含初始化应答数据的应答.

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

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

encryption-required

初始化实体请求的机制不能使用，出非当前的流的保密性和完整性收到保护(典型的是通过TLS); 对<auth/>元素发送应答(包含或不包含初始化应答数据).

```
I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='PLAIN'>AGp1bGlldAByMG0zMG15cjBtMzA=</auth>

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

incorrect-encoding

初始化实体提供的数据无法被处理，因为 base 64 编码不正确(例如, 因为编码没有遵循BASE64的第四章的定义); 对<response/>元素或<auth/>元素发送包含初始化应答数据的应答.

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

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

invalid-authzid

初始化实体提供的authzid是非法的, 要么因为它格式不正确要么因为初始化实体没有权限授权那个ID; 对<response/>元素或<auth/>元素发送包含初始化应答数据的应答.

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

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

invalid-mechanism

初始化实体没有指定一个机制, 或请求的机制不被接收方实体支持; 对<auth/>元素发送应答.

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

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

malformed-request

请求是不良的(例如, <auth/>元素包含了初始化应答数据但是机制不允许这个, 或被发送的数据违反了指定的SASL机制的语法); 对<abort/>, <auth/>, <challenge/>, 或 <response/> 元素发送应答.

(下例中, <auth/>元素的XML字符串数据包含了多于255个UTF-8编码的Unicode字符，所以违反了定义于ANONYMOUS的SASL ANONYMOUS的"token"生产.)

```
I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='ANONYMOUS'>[ ... some-long-token ... ]</auth>

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

mechanism-too-weak

初始化实体请求的机制弱于服务器策略允许初始化实体使用的机制; 对<auth/>元素发送应答(包含或不包含初始化应答数据).

```
I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='PLAIN'>AGp1bGlldAByMG0zMG15cjBtMzA=</auth>

R: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     <mechanism-too-weak/>
   </failure>
```

not-authorized

验证失败，因为初始化实体没有提供正确的证书, 或因为发生了一些普通的验证失败而接收方实体不希望泄露失败原因的特定信息; 对<response/>元素或<auth/>元素发送包含初始化应答数据的应答.

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

R: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     <not-authorized/>
   </failure>
```

> 安全警告: 这个错误条件包含但不限于不正确的证书或不存在的用户名的情形. 为了组织目录获取攻击, 不正确的证书和不存在的用户名两者没有区别.

temporary-auth-failure

验证失败，因为接收方实体的临时性错误, 可以建议初始化实体晚点再试; 对<auth/>元素或<response/>元素发送应答.

```
I: <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     [ ... ]
```

```
   </response>

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

SASL定义

SASL的范本需求里面要求使用中的协议定义必须提供以下信息.

服务名:

"xmpp"

初始化序列:

在初始化实体提供一个打开的XML流头之后，接收方实体以同样的方式应答, 接收方实体提供一个可接受的验证方法的列表. 初始化实体从列表里选择一个方法并在放在<auth/>元素的'mechanism'属性的值里发送给接收方实体, 可选择包含一个初始化应答以避免多一个来回.

交换序列:

挑战和应答是通过从接收方实体发给初始化实体的<challenge/>元素和从初始化实体发送给接收方实体的<response/>元素. 接收方实体通过发送<failure/>元素来报告失败，通过发送<success/>元素来报告成功; 初始化实体通过发送<abort/>元素来放弃交换. 在成功协商之后, 双方确认原始的XML流被关闭而双方发送新的流头.

安全层协商:

安全层，对于接收方实体来说，发送完<success/>元素的'>'字符之后立刻生效, 对于初始化实体来说，在接收到<success/>元素的'>'字符之后立刻生效. 层的顺序是先 TCP, then TLS, then SASL, 然后 XMPP.

授权身份的使用:

授权身份在XMPP中可被用于指示客户端的非缺省的<localpart@domainpart>; 空字符串等于缺少授权身份.

资源绑定

原理

在客户端从一个服务器验证之后, 它必须绑定一个特定的资源到这个流，这样服务器才能正确地对客户端寻址. 就是说, 必须有一个XMPP资源关联到客户端的纯JID (<localpart@domainpart>), 所以在那个流上使用的地址是一个全JID，格式为<localpart@domainpart/resource> (包含资源部分). 这确保服务器可以向客户端相关的实体而不是服务器本身或客户端的帐号递送XML节和从客户端相关的实体而不是服务器本身或客户端的帐号接收XML节, 详见 第十章.

参考文献: 在绑定资源之前，客户端可以和服务器本身或客户端帐号交换数据，因为全JID只在被这个客户端和服务器之间已协商好的流的上下文的外部寻址时需要用到, 但这不是常规做法.

在客户端已经绑定了一个资源到该流之后, 它被视为一个 "已连接的资源". 服务器应该允许一个实体同时维持多个已连接资源, 每个已连接的资源关联到一个唯一的XML流并且和其他已连接的资源的资源部分是不同的.

> 安全警告: 服务器应该允许一个XMPP服务的管理员限制已连接资源的数量，为了防止特定的拒绝服务攻击，详见13.12.

如果, 在完成资源绑定步骤之前, 客户端尝试发送一个XML节给另一个不是服务器本身或客户端的帐号的实体, 服务器不能(MUST NOT)处理这个节而必须以<not-authorized/>流错误(4.9.3.12)关闭这个流.

资源绑定扩展的XML命名空间是 'urn:ietf:params:xml:ns:xmpp-bind'.

支持

在XMPP客户端和服务器实现中，对于资源绑定的支持是必需的.

流协商规则

强制协商

流的双方必须确保资源绑定是强制协商的.

重启

在资源绑定之后, 双方不能(MUST NOT)重启该流.

声明支持

在SASL协商成功之后，服务器发送一个新的应答流头给客户端, 这时服务器必须在它展示给客户端的流特性中包含一个由'urn:ietf:params:xml:ns:xmpp-bind'命名空间限定的<bind/>元素.

服务器不能(MUST NOT)包含资源绑定流特性，直到客户端验证之后, 通常就是SASL协商成功之后.

```
S: <stream:stream
       from='im.example.com'
       id='gPybzaOzBmaADgxKXu9UClbprp0='
       to='juliet@im.example.com'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'>

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

在得到资源绑定是强制协商的通知之后, 客户端必须绑定一个资源到流上，如下面章节所述.

资源标识符的生成

最低限度，资源部分在<localpart@domainpart>已连接的资源中必须是唯一的. 这个强制性策略是由服务器来负责的.

安全警告: 资源部分可能是关乎安全的. 例如, 如果一个恶意的实体猜测一个客户端的资源部分然后它能确定该客户端(也就是控制的主体)是在线还是离线, 所以导致如13.10.2所述的联机状态泄漏. 为了防止那种可能性, 客户端可以要么 (1) 它自己生成一个随机的资源部分，要么(2) 请求服务器帮它生成一个资源部分. 一个确保资源部分随机性的方法是生成一个通用唯一标识符(UUID)，如 UUID 所述.

服务器生成的资源标识符

一个服务器必须能代替客户端生成XMPP资源部分. 由服务器生成的资源部分必须是随机的(参见RANDOM).

成功情形

客户端，通过发送一个类型为"set"并包含了一个由'urn:ietf:params:xml:ns:xmpp-bind'命名空间限定的空的<bind/>元素的IQ节(见8.2.3)来请求一个服务器生成的资源部分.

```
C: <iq id='tn281v37' type='set'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
   </iq>
```

一旦服务器为该客户端生成了一个XMPP部分, 它必须返回一个类型为"result"的IQ节给该客户端, 这个节里面必须包含一个<jid/>元素来指定服务器决定的已连接资源的全JID.

```
S: <iq id='tn281v37' type='result'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <jid>
         juliet@im.example.com/4db06f06-1ea4-11dc-aca3-000bcd821bfb
       </jid>
     </bind>
   </iq>
```

错误情形

当一个客户端在资源绑定时请求服务器生成一个资源部分, 定义了以下节错误条件:

- 该帐号已经达到了被允许的并发资源连接数量限制.
- 该客户端不被允许绑定一个资源到该流.

自然的, 可能有这里没定义的错误条件发生, 如8.3所述.

资源约束

如果帐号已经达到被允许的并发连接资源数限制, 服务器必须返回一个<resource-constraint/>节错误(8.3.3.18).

```
S: <iq id='tn281v37' type='error'>
     <error type='wait'>
       <resource-constraint
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

不允许

如果客户端不被允许绑定一个资源到该流, 服务器必须返回一个<not-allowed/>节错误(8.3.3.10).

```
S: <iq id='tn281v37' type='error'>
     <error type='cancel'>
       <not-allowed
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

客户端提交的资源标识符

不同于请求服务器代替自己生成一个资源部分, 一个客户端可以尝试提交一个它自己生成的或受控制的用户已经提供的资源部分.

成功情形

客户端发送类型为"set"包含<bind/>元素以及拥有非空XML字符串数据的<resource/>子元素的IQ节, 以请求它的服务器接受一个客户端提交的资源部分.

```
C: <iq id='wy2xa82b4' type='set'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <resource>balcony</resource>
     </bind>
   </iq>
```

该服务器应该接受这个客户端提交的资源部分. 它返回一个类型为"result"的IQ节给该客户端, 其中包含一个<jid/>子元素来为已连接的资源指定全JID并包含未修改的客户端提交的文本.

```
S: <iq id='wy2xa82b4' type='result'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
      <jid>juliet@im.example.com/balcony</jid>
    </bind>
   </iq>
```

或者, 基于本地服务策略, 该服务器可以拒绝客户端提交的资源部分并以服务器生成的资源部分覆盖它.

```
S: <iq id='wy2xa82b4' type='result'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
      <jid>
   juliet@im.example.com/balcony 4db06f06-1ea4-11dc-aca3-000bcd821bfb
      </jid>
    </bind>
   </iq>
```

错误情形

当一个客户端在资源绑定期间尝试提交它自己的XMPP资源部分, 除了7.6.2还定义了以下节错误条件:

- 所提供的资源部分无法被服务器处理.
- 所提供的资源部分已经被使用.

自然的, 有一些未在这里定义的错误条件可能发生, 如8.3所述.

坏请求

如果提供的资源部分无法被服务器处理(例如, 因为它长度为零或因为它违反了定义于XMPP-ADDR的资源部分的其他规则), 该服务器可能返回一个<bad-request/>节错误(8.3.3.1)而不应该处理这个资源部分， 这样就保持了一致性.

```
S: <iq id='wy2xa82b4' type='error'>
     <error type='modify'>
       <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

冲突

如果当前有一个已连接的客户端的会话被新连接的客户端请求了, 服务器必须做以下事情之一(该服务器做的这些事情之一对于实现或者本地服务策略是一个麻烦, 尽管下面提供了一些建议).

1. 以一个服务器生成的资源部分覆盖新连接的客户端提供的资源部分. 这一行为是被提倡的, 因为对于客户端实现来说它简化了资源绑定过程.
2. 不允许新连接的客户端的资源绑定并保持当前已连接客户端的会话. 这一行为既不提倡也不反对, 尽管实际上它在RFC 3920中被隐性地提倡; 然而, 请注意对<conflict/>错误的处理并不总是被现有的客户端实现支持的, 它经常被当成一个验证错误并且当收到这个错误的时候丢弃缓存的凭证.
3. 中止当前已连接的客户端的会话并允许新连接的客户端的资源绑定尝试. 尽管这是早期XMPP服务器实现的传统行为, 现在不提倡这么做了， 因为它可能导致两个客户端互相挂掉多方的无线循环; 无论如何, 注意这个行为在某些布署场景中可能是适当的， 要么如果服务器知道当前已连接的客户端有一个死连接， 要么有一个4.6所述的断裂的流.

如果服务器遵循1号行为, 它返回一个类型为"result"的<iq/>节给新连接的客户端, 这里的<bind/>元素的<jid/>子元素包含XML字符串数据指定该客户端的全JID, 包含服务器生成的资源部分.

```
S: <iq id='wy2xa82b4' type='result'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <jid>
    juliet@im.example.com/balcony 4db06f06-1ea4-11dc-aca3-000bcd821bfb
       </jid>
     </bind>
   </iq>
```

如果服务器遵循2号行为, 它发送一个<conflict/>节错误(8.3.3.2)应答新连接的客户端的资源绑定尝试但是保持这个XML流， 这样新连接的客户端有机会去协商一个不冲突的资源部分(即, 新连接的客户端在做下一次绑定资源的尝试之前需要选择一个不同的资源部分).

```
S: <iq id='wy2xa82b4' type='error'>
     <error type='modify'>
       <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

如果服务器遵循3号行为, 它返回一个<conflict/>流错误(4.9.3.3)给当前的已连接客户端(如4.9.3.3所述)并返回一个类型为"result"的IQ节(表示成功)新连接的应答资源绑定尝试.

```
S: <iq id='wy2xa82b4' type='result'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <jid>
         juliet@im.example.com/balcony
       </jid>
     </bind>
   </iq>
```

重试

如果客户端提交资源部分的时候发生了一个错误, 服务器应该允许可配置的但是合理的重试次数(至少5次且不高于10次); 这让客户端能够不需要被迫重新连接就可以纠正不正确提通的资源部分(例如, 坏的数据格式或重复的文本字符串).

在客户端达到重试次数限制之后, 服务器必须以<policy-violation/>流错误(4.9.3.14)关闭这个流.

XML节

在一个客户端和一个服务器(或两个服务器)完成了流协商之后, 双方就可以发送XML节了. 对于'jabber:client'和'jabber:server'命名空间定义了三种XML节: <message/>, <presence/>, 和 <iq/>. 另外, 这些节类型有五种常见属性. 这些常见属性, 以及这三种节类型的基本语义, 定义于本协议; 更多即时消息和联机状态应用相关以及有关XMPP扩展协议的应用的XML节语法的详细信息在XMPP-IM里提供.

XMPP客户端和服务器实现必须支持本协议所定义的XML节语法和语义.

> 安全警告: 服务器不能(MUST NOT)处理部分的节， 也不能(MUST NOT)针对一个节的任何部分仍在传输时(在承认收到关闭标签之前)猜测其含义.

常见属性

以下五种属性常见于 message, presence, 和 IQ 节.

to

'to'属性指定该节期望的接收者的JID.

```
<message to='romeo@example.net'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

关于基于'to'地址的入站和出站XML节的服务器处理的信息, 参考第十章.

客户端-服务器流

以下规则适用于已连接客户端通过一个'jabber:client'命名空间限定的XML流发送给它的服务器的节中包含的'to'属性.

1. 一个拥有特定接收者(例如, 一个会话伙伴, 一个远程服务, 该服务器本身, 甚至该用户的纯JID的另一个资源)的节必须拥有一个'to'属性，它的值是一个XMPP地址.
2. 一个从客户端发送到服务器的由该服务器直接处理的节(例如, XMPP-IM所述的好友列表处理或发送给服务器用来广播给其他实体的的联机状态信息)不能(MUST NOT)拥有'to'属性.

以下规则适用于服务器通过一个'jabber:client'命名空间限定的XML流发送到已连接客户端的节中包含的'to'属性..

1. 如果该服务器从另一个已连接客户端或从一个对端服务器接收到该节, 在递送该节给该客户端之前该服务器不能(MUST NOT)修改'to'地址.
2. 如果该服务器本身生成了这个节(例如, 对类型为"get"或"set"的IQ节的应答, 即使该节不包含一个'to'地址), 这个节可以包含一个'to'地址, 这个地址必须是该客户端的全JID, 如果这个节不包含'to'地址，那么该客户端必须把'to'地址视为等同于该客户端的全JID.

> 实现备注: 只递送节到客户端的全JID还是用户的纯JID,是服务器的责任; 就是说, 客户端不需要检查收到的节的'to'地址. 然而, 如果客户端不检查'to'地址，那么建议最好检查纯JID部分(不是全JID), 因为'to'地址可能是该用户的纯JID, 该客户端的当前全JID, 或甚至是一个不同资源的全JID(例如, 在XEP-0160所述的的所谓"离线消息"的情况下).

服务器-服务器流

以下规则适用于一个'jabber:server'命名空间限定的XML流(即,服务器-服务器 流)的上下文中包含的'to'属性.

1. 一个节必须拥有'to'属性，它的值是一个XMPP地址; 如果服务器接收到一个不满足这一限定的节, 它必须以一个<improper-addressing/>流错误(4.9.3.7)来关闭这个流.
2. 包含在这个节的'to'属性中的JID的域部分必须和通过SASL协商(见第六章)与之通讯的, 或服务器回拨(见XEP-0220),或类似的接收方服务器(或其中任何有效的域名)的完全合法域名FQDN匹配; 如果服务器接收到的节不满足这个限定, 它必须以<host-unknown/>流错误(4.9.3.6)或<host-gone/>流错误(4.9.3.5)来关闭该流.

from

'from'属性指定发送者的JID.

```
<message from='juliet@im.example.com/balcony'
         to='romeo@example.net'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

客户端-服务器流

以下规则适用于被'jabber:client'命名空间限定的XML流(即, 客户端-服务器 流)上下文中的'from'属性.

1. 当服务器从一个已连接客户端接收到一个XML节, 该服务器必须给这个节添加一个'from'属性或覆盖这个由客户端指定的'from'属性, 这里'from'属性的值必须是服务器针对生成这个节的已连接资源确定的全JID (<localpart@domainpart/resource>) (见4.3.6), 或在和订阅相关的联机状态信息节(见XMPP-IM)的情况下则是纯JID (<localpart@domainpart>).
2. 当服务器为它自己生成一个从服务器本身发送给客户端的节的时候, 这个节必须包含一个'from'属性，它的值是服务器在流协商中同意的纯JID (即, <domainpart>)(例如, 基于初始化流头中的'to'属性).
3. 当服务器生成一个从该服务器递送到已连接客户端的帐号本身的节的时候(例如, 在服务器代表客户端提供的数据存储服务的上下文中), 该节要么 (a) 不包含'from'属性，要么 (b) 包含一个值为该帐号纯JID(<localpart@domainpart>)的'from'属性.
4. 服务器不能(MUST NOT)给客户端发送不包含'from'属性的节，如果该节不是由服务器代表它本身生成的(例如, 如果它是由另一个客户端或对端服务器生成的，而该服务器仅仅递送它到客户端或一些其他的实体); 所以, 当一个客户端接收到一个不包含'from'属性的节的时候, 它必须假定这个节是从该用户帐号所在服务器发出的.

服务器-服务器流

以下规则适用于一个'jabber:server'命名空间限定的XML流(即,服务器-服务器 流)的上下文中包含的'from'属性.

1. 一个节必须拥有'from'属性，它的值是一个XMPP地址; 如果服务器接收到一个不满足这一限定的节, 它必须以一个<improper-addressing/>流错误(4.9.3.7)来关闭这个流.
2. 包含在这个节的'from'属性中的JID的域部分必须和通过SASL协商(见第六章)与之通讯的, 或服务器回拨(见XEP-0220),或类似的发送方服务器(或其中任何有效的域名)的完全合法域名FQDN匹配; 如果服务器接收到的节不满足这个限定, 它必须以<invalid-from/>流错误(4.9.3.9)来关闭该流.

强制执行这些规则有助于组织特定的如13.12所述的拒绝服务攻击.

id

'id'属性是由发起方实体用来跟踪可能从其他实体(类似中间服务器或预期的接收者)收到的和它生成的节有关的任何应答或错误节.

这个'id'属性仅在当前流保持唯一性还是全局保持唯一性，取决于发起方实体本身.

对于<message/>和<presence/>节来说, 建议发起方实体包含一个'id'属性; 对于<iq/>节来说, 它是必需的.

如果生成的节包含一个'id'属性，那么对于其相应的应答或错误节来说，也必须包含一个'id'属性, 这个'id'属性的值必须和生成的节的那个'id'属性值匹配.

IQ节语义强加了额外的约束，参见8.2.3.

type

'type'属性指定该消息,联机状态或IQ节的用途或上下文. 'type'属性的特定的允许值依赖于这个节是一个消息, 联机状态, 还是IQ节. 为消息和联机状态节定义的值用于即时消息和联机状态应用，所以定义于XMPP-IM中, 而为IQ节定义的值指定所有结构化请求-应答交换中的语义部分(无论载荷是什么), 所以它定义于8.2.3. 唯一通用于所有三种节的'type'值是"error"，定义于8.3.

xml:lang

一个节应该拥有'xml:lang'属性(定义于的2.12节XML)，如果这个节包含了XML字符串数据打算展示给用户(如CHARSETS所解释的, "可读国际化"). 'xml:lang'属性的值指定任何这类可读XML字符串数据的缺省语言.

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

'xml:lang'属性的指可以被特定子元素的'xml:lang'属性覆写.

```
<presence from='romeo@example.net/orchard' xml:lang='en'>
  <show>dnd</show>
  <status>Wooing Juliet</status>
  <status xml:lang='cs'>Dvo&#x0159;&#x00ED;m se Julii</status>
</presence>
```

如果一个由客户端生成的出站节不拥有'xml:lang'属性, 该客户端的服务器应该添加一个'xml:lang'属性，其值为客户端的出站流所指定的值，如4.7.4所述.

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

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

如果一个被客户端或服务器接收到的入站节不拥有'xml:lang'属性, 一个实现必须假定缺省语言是该实体的输入流所指定的值，如4.7.4所述.

'xml:lang'属性的值必须遵循 NMTOKEN 数据类型(定义于XML的2.3节) 并且必须遵循定义于LANGTAGS的格式.

服务器不能(MUST NOT)修改或删除它从其他实体收到的节的'xml:lang'属性.

基本语义

消息语义

<message/>节是一个"推送"机制，这里一个实体推送信息到另一个实体, 类似发生在email系统里的通讯一样. 所有消息节将拥有'to'属性用来指定该消息期望的接收者 (见8.1.1和10.3), 除非消息是被一个已连接的客户端帐号的纯JID发送的. 接收到一个带有'to'地址的消息节之后, 服务器应该尝试路由或递送它到期望的接收者那里(见第十章里和XML节相关的通用路由和递送规则).

联机状态语义

<presence/>节是一个特定的"广播"或"发布-订阅"机制, 这里多个实体接收关于他们订阅的一个实体的信息(在这个案例中, 是网络可用性信息). 通常, 发布客户端应该发送一个不带有'to'属性的联机状态节, 这种情况下该客户端连接的那个服务器将广播那个节给所有已订阅的实体. 然而, 发布客户端也可以发送一个带有'to'属性的联机状态节, 这种情况下该服务器将路由或递送那个节到期望的接收者. 尽管<presence/>节大部分情况下是由XMPP客户端使用, 它也可能被服务器, 附加服务, 以及任何其他类型呃XMPP实体使用. 参见第十章中和XML节相关的通用路由和递送规则, 以及XMPP-IM中联机状态应用的特定规则.

IQ语义

信息查询(Info/Query),或IQ, 是一个"请求-应答"机制, 类似某些情况下的超文本传输协议HTTP. IQ的语义允许一个实体对另一个实体做出一个请求, 并接收一个应答. 这个请求和应答的数据内容由schema或其他限定IQ元素的直接子元素的XML命名空间相关的结构化定义(见8.4)来限定, 发出请求的实体使用'id'属性来跟踪交互过程. 所以, IQ交互沿用了结构化数据交换的常见模式，类似 get/result 或 set/result (尽管适当的时候对于某个请求会返回一个error):

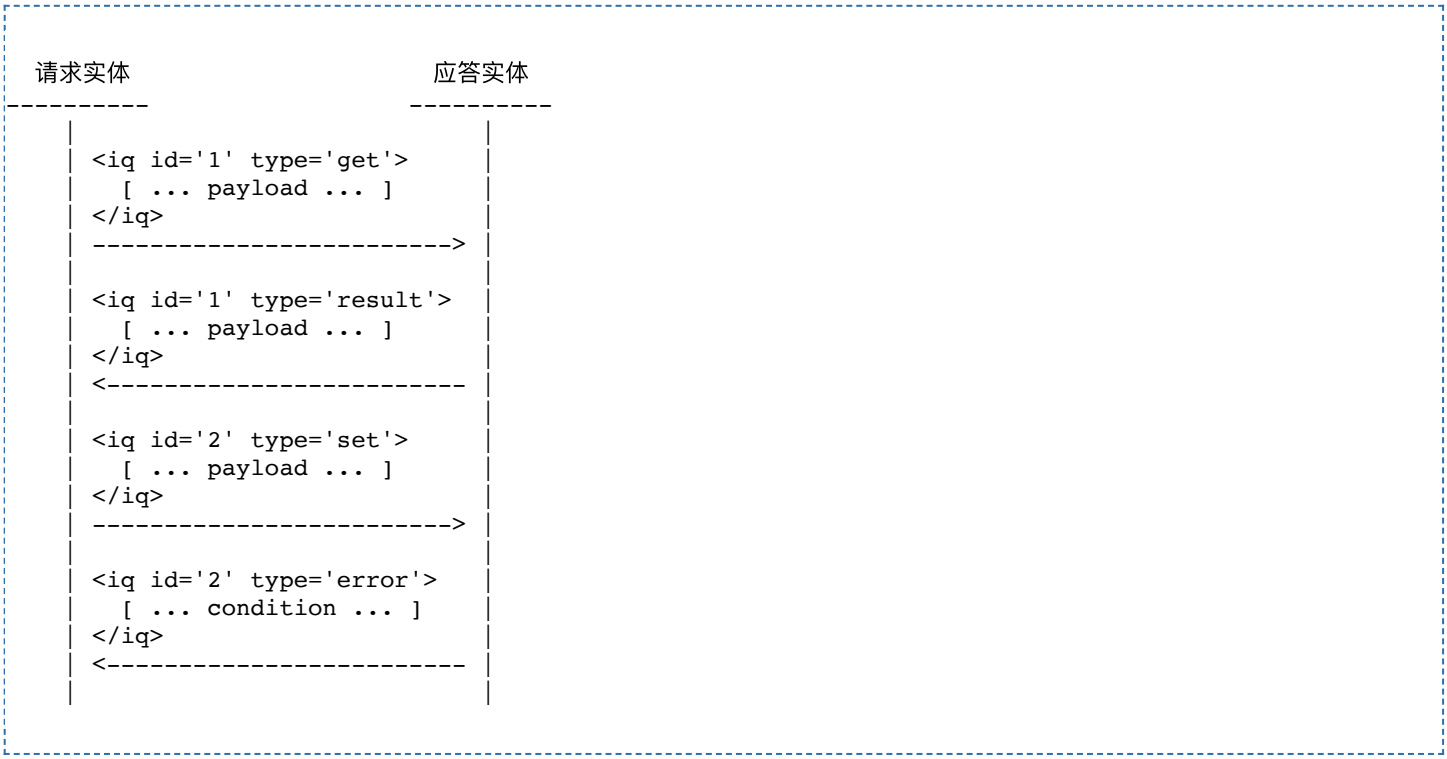


图5: IQ节的语义

为强制这些语义, 以下规则适用:

1. 'id'属性对于IQ节是必需的.
2. 'type'属性对于IQ节是必需的. 这个值必须是以下之一; 如果不是, 接收者或中间路由器必须返回一个<bad-request/>节错误(8.3.3.1).

    - get -- 该节请求信息, 查询需要什么数据以完成更多操作, 等等.
    - set -- 该节为完成某个操作提供需要的数据, 设置新值, 取代旧值, 等等.
    - result -- 该节是对成功的get或set请求的应答.
    - error -- 该节报告关于处理或递送一个get或set请求时发生的错误(见8.3).

3. 接收到类型为"get"或"set"的IQ请求的实体必须返回一个类型为"result"或"error"的IQ应答. 该应答必须保留请求中的'id'属性(或为空，如果生成的节没有包含'id'属性).

4. 接收到类型为"result"或"error"节的实体不能(MUST NOT)发送更多的类型为"result"或"error"的IQ应答来应答; 然而, 请求实体可以发送另一个请求(例如, 一个类型为"set"的IQ对之前在get/result对中查询到的信息提供特定的信息).
5. 类型为"get"或"set"的IQ节必须严格地包含一个子元素, 它定义特定请求的语义.
6. 类型为"result"的IQ节必须包含零或一个子元素.
7. 类型为"error"的IQ节可以包含相关的"get"或"set"子元素并且必须包含一个<error/>子元素; 详见8.3.

节错误

节相关的错误处理的方式类似流错误流错误, 但是不像流错误那样， 节错误是可恢复的; 所以, 他们不会导致XML和当前TCP连接的中止. 反之, 发现错误条件的实体返回一个错误节, 它是一个这样的节:

- 是和触发这个错误的已生成的节同种类型(message, presence, 或 IQ)
- 'type'属性值设为"error"
- 通常是把已生成的节的'from'和'to'互换
- 镜像触发这个错误的已生成的节的'id'属性(如果有的话)
- 包含一个<error/>子元素以指明错误条件并且对发送者可以采取的补救措施提供一个暗示(然而, 不可能总是能够不补救这个错误)

规则

以下规则适用于节错误:

1. 检测到和节相关的错误条件的接收或处理实体应该返回一个错误节(对于IQ节必须这么做).
2. 该错误节应该简单地把生成的节中的'from'和'to'地址互换, 除非这么做将会 (1) 导致信息泄漏(参见[RFC6120#信息泄露|13.10]])或其他违反安全, 或 (2) 强迫错误节的发送者在该错误节的'from'或'to'地址中包含一个异常的JID.
3. 如果生成的节是<message/>或<presence/>并且包含了'id'属性，那么该错误节必须也包含'id'属性. 如果生成的节是<iq/>，那么该错误节必须包含一个'id'属性. 在所有情况下, 'id'属性的值必须和生成节的相同(或者是空，如果生成的节没有包含'id'属性).
4. 错误节必须包含一个<error/>子元素.
5. 返回错误节的实体可以传递它的JID给生成节的发送者(例如, 为了诊断或跟踪的目的)，通过附加一个'by'属性到<error/>子元素.
6. 返回错误节的实体可以包含被发送的原始XML，这样发送者能够检查, 如果必要的话, 并在尝试重发之前纠正该XML(然而, 这只是出于礼貌，并且原实体不能(MUST NOT)依赖接收到的原始载荷). 自然地, 该实体不能(MUST NOT)包含原始数据，如果它不是格式良好的XML, 违反XMPP的XML限制(见11.1), 或反而是有害的(例如, 超出大小限制).
7. 如果'type'属性值不是"error"(或如果没有'type'属性), 不能(MUST NOT)包含一个<error/>子元素.
8. 接收到错误节的实体不能(MUST NOT)以更多的错误节来应答这个节; 这有助于防止死循环.

语法

节相关的错误的语法如下, 这里展示的用方括号'['和']'括起来的XML数据是可选的, 'intended-recipient' 是原始节指定的地址的那个实体的JID, 'sender' 是原始实体的JID, 而 'error-generator' 是检测到错误的并方会错误节的那个实体.

```
<stanza-kind from='intended-recipient' to='sender' type='error'>
  [OPTIONAL to include sender XML here]
  <error [by='error-generator']
         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"必须是 message, presence, 或 iq 之一.

"error-type" 必须是以下之一:

- auth -- 在提供身份之后重试
- cancel -- 不要重试 (该错误不能加以弥补)
- continue -- 继续 (这个条件只是个警告)
- modify -- 在修改发送的数据之后重试
- wait -- 等待之后重试 (该错误是暂时的)

"defined-condition"必须符合8.3.3定义的节错误条件之一. 然而, 因为将来可能会发生额外的错误条件, 如果实体接受到一个它不理解的节错误条件，那么它必须把这个未知的条件当成<undefined-condition/> (8.3.3.21). 如果一个XMPP协议扩展的设计者或一个XMPP实现的开发者需要未在本协议中定义的节错误条件的通讯, 他们可以定义应用特有的命名空间所限定的应用特有的错误条件元素来实现这个目标.

<error/>元素:

- 必须包含一个已定义的条件元素.
- 可以包含一个包含XML字符串数据的<text/>子元素，用来描述错误的详细信息; 这个元素必须由'urn:ietf:params:xml:ns:xmpp-stanzas'命名空间来限定并且应该拥有'xml:lang'属性来指定该XML字符串数据的自然语言.
- 可以包含一个用于应用特有的错误条件的子元素; 这个元素必须由一个应用特有的命名空间来限定，以定义该元素的语法和语义.

<text/>元素是可选的. 如果包含了它, 它仅被用于提供描述和诊断信息以补充说明已定义条件或应用特有的条件的含义. 它不能(MUST NOT)被应用程序当成编程信息来解释. 它不应该被用于向自然人用户展示错误消息, 但是可以被附加在该已定义条件元素(以及, 可选的, 应用特有的条件元素)的错误消息上展示.

> 互操作性备注: 定义于RFC3920的语法包含了一个遗留的'code'属性, 它的语义已经被已定义的条件元素取代; 关于已定义的条件元素和遗留的'code'属性值之间的对应关系，可以在XEP-0086]找到.

已定义的条件

以下条件是已定义好用于节错误的.

error-type 的值是被推荐用于每个已定义的条件通常预期的类型; 无论如何, 在某些情况下不同的类型可能更合适.

bad-request

发送者发送的节里包含的XML不符合适当的schema或不能被拥有(例如, IQ节的'type'属性包含一个不能识别的值, 或一个被已知的命名空间限定的元素但是违反了该元素的已定义的语法); 相关的错误类型应该是"modify".

```
C: <iq from='juliet@im.example.com/balcony'
       id='zj3v142b'
       to='im.example.com'
       type='subscribe'>
     <ping xmlns='urn:xmpp:ping'/>
   </iq>

S: <iq from='im.example.com'
       id='zj3v142b'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error type='modify'>
       <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

conflict

访问未被授权，因为一个现存的资源使用了相同的名字或地址; 相关的错误类型应该是"cancel".

```
C: <iq id='wy2xa82b4' type='set'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <resource>balcony</resource>
     </bind>
   </iq>

S: <iq id='wy2xa82b4' type='error'>
     <error type='cancel'>
       <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

feature-not-implemented

出现在XML节里的特性没有被预定的接收方或中间服务器实现，所以该节无法被处理(例如, 该实体知道该命名空间但是不认识元素名); 相关的错误类型应该是"cancel" 或 "modify".

```
C: <iq from='juliet@im.example.com/balcony'
       id='9u2bax16'
       to='pubsub.example.com'
       type='get'>
     <pubsub xmlns='http://jabber.org/protocol/pubsub'>
       <subscriptions/>
     </pubsub>
   </iq>

E: <iq from='pubsub.example.com'
       id='9u2bax16'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error type='cancel'>
       <feature-not-implemented
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
```

```
      <unsupported
          xmlns='http://jabber.org/protocol/pubsub#errors'
          feature='retrieve-subscriptions'/>
    </error>
  </iq>
```

forbidden

请求的实体没有必要的许可来执行一个只允许特定授权角色或个体来完成的动作(即, 它通常和授权而不是验证有关); 相关的错误类型应该是"auth".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error type='auth'>
       <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </presence>
```

gone

接收者或服务器无法再用这个地址联系到, 通常是永久意义上的(和<redirect/>错误条件相反, 它被用于临时的地址失败); 相关的错误类型应该是"cancel"并且该错误节应该包含一个新的地址(如果可用的话)作为<gone/>元素的XML字符串数据(它必须是一个实体可以联系的唯一资源标识符URI或国际化资源标识符IRI, 典型的是一个XMPP-URI定义的XMPP IRI.

```
C: <message
       from='juliet@im.example.com/churchyard'
       id='sj2b371v'
       to='romeo@example.net'
       type='chat'>
     <body>Thy lips are warm.</body>
   </message>

S: <message
       from='romeo@example.net'
       id='sj2b371v'
       to='juliet@im.example.com/churchyard'
       type='error'>
     <error by='example.net'
            type='cancel'>
       <gone xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
         xmpp:romeo@afterlife.example.net
       </gone>
     </error>
   </message>
```

internal-server-error

服务器发生了错误的配置或其他阻止它处理改节的内部错误; 相关的错误类型应该是"cancel".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error type='cancel'>
       <internal-server-error
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </presence>
```

item-not-found

找不到请求的JID地址或条目; 相关的错误类型应该是"cancel".

```
C: <presence from='userfoo@example.com/bar'
             id='pwb2n78i'
             to='nosuchroom@conference.example.org/foo'/>

S: <presence from='nosuchroom@conference.example.org/foo'
             id='pwb2n78i'
             to='userfoo@example.com/bar'
             type='error'>
     <error type='cancel'>
       <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </presence>
```

> 安全警告: 如果这样做将会提供预定的接收者的网络可用性信息给一个未被授权知道这些信息的实体(关于联机状态信息授权的更多细节讨论，参考XMPP-IM联机状态订阅的讨论)，那么应用不能(MUST NOT)返回这个错误; 相反它应该返回一个<service-unavailable/>错误(8.3.3.19).

jid-malformed

发送的实体所提供(例如, 在资源绑定的时候)或与之通讯(例如, 一个节的'to'地址)的XMPP地址或其中一部分违反了XMPP-ADDR定义的规则; 相关的错误类型应该是"modify".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='ch@r@cters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='ch@r@cters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error by='muc.example.com'
```

```
           type='modify'>
      <jid-malformed
          xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </presence>
```

实现备注: 强制XMPP本地部分的格式主要是相关帐号或实体所在的服务的责任(例如, example.com 服务负责返回所有和格式 <localpart@example.com> 相关的 <jid-malformed/> 错误), 而强制XMPP域部分的格式主要是由路由一个节到域部分所指定的服务的那个服务的责任(例如, example.org 服务负责返回该服务尝试发送的目标JID <localpart@example.com>的格式的 <jid-malformed/> 错误). 无论如何, 任何检测到格式错误的JID的实体可以返回该错误.

not-acceptable

接收者或服务器理解该请求但是不能处理它，因为该请求不符合该接收者或服务器的标准(例如, 请求订阅信息但是未同时包含接收者需要的配置参数); 相关的错误类型应该是"modify".

```
C: <message to='juliet@im.example.com' id='yt2vs71m'>
     <body>[ ... the-emacs-manual ... ]</body>
   </message>

S: <message from='juliet@im.example.com' id='yt2vs71m'>
     <error type='modify'>
       <not-acceptable
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </message>
```

not-allowed

接收者或服务器不允许任何实体执行该动作(例如, 向列入黑名单的域发送消息); 相关的错误类型应该是"cancel".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error type='cancel'>
       <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </presence>
```

not-authorized

发送者在被允许执行某动作之前需要提供凭证, 或已经提供了错误的凭证("not-authorized"的提法, 来源于HTTP的"401 Unauthorized"错误, 可能导致读者认为这个条件是和授权相关的, 但其实它通常用于验证相关的领域); 相关的错误类型应该是"auth".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'>
     <error type='auth'>
       <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </presence>
```

policy-violation

实体违反了一些本地服务策略(例如, 一个消息包含了服务禁止的单词)而服务器可以选择在<text/>元素里或在应用特有的条件元素里指定策略; 相关的错误类型应该是"modify"或"wait", 取决于被违反的策略.

(在下例中, 客户端发送一个包含了根据服务器的本地服务策略被禁止的单词的XMPP消息.)

```
C: <message from='romeo@example.net/foo'
            to='bill@im.example.com'
            id='vq71f4nb'>
     <body>%#&@^!!!</body>
   </message>

S: <message from='bill@im.example.com'
            id='vq71f4nb'
            to='romeo@example.net/foo'>
     <error by='example.net' type='modify'>
       <policy-violation
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </message>
```

recipient-unavailable

预期的接收者暂时不可用, 正在维护, 等等; 相关的错误类型应该是"wait".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'>
     <error type='wait'>
       <recipient-unavailable
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </presence>
```

> 安全警告: 如果这么做将提供关于预期接收者的网络可用性的信息给一个没有被授权可以知道这类信息的实体(对于联机状态授权方面的讨论的更多细节, 参考联机状态信息订阅的讨论XMPP-IM中),应用不能(MUST NOT)返回这个错误; 反之，它必须返回一个<service-unavailable/> 节错误(8.3.3.19).

redirect

接收者或服务器重定向该信息的请求到另一个实体, 典型的临时发生的情形(和<gone/>错误条件相反, 它用于永久性的地址错误); 相关的错误类型应该是"modify"，并且该错误节应该在<redirect/>元素的XML字符串数据中包含替代的地址(它必须是一个发送者可以与之通讯的URI或IRI, 通常是一个XMPP-URI定义的XMPP IRI).

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'
       type='error'>
     <error type='modify'>
       <redirect xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
         xmpp:characters@conference.example.org
       </redirect>
     </error>
   </presence>
```

> 安全警告: 接收到一个节级别的重定向的应用应该向自然人用户警告这个重定向尝试并且在和包含在该<redirect/>元素的XML字符串数据中的地址代表的实体通讯之前请求批准, 因为那个实体可能有不同的身份或可能强制执行不同的安全策略. XMPP节的点对点验证或签名有助于减轻这个风险, 因为它将允许发送者来决定是否被重定向到的实体和开始尝试联系的实体是同一个身份. 应用可以有一个策略规定只有已经验证过接收实体才能重定向. 另外, 在成功地重定向了一定次数之后应用应该中止通讯尝试(例如, 至少2次但不超过5次).

registration-required

请求的实体没有被授权访问请求的服务，因为需要事先注册(提前注册的例子包括XMPP多用户聊天XEP-0045 中仅限会员的房间和到非XMPP即时消息服务的网关, 传统上使用网关XEP-0100是需要注册的); 相关的错误类型应该是"auth".

```
C: <presence
       from='juliet@im.example.com/balcony'
       id='y2bs71v4'
       to='characters@muc.example.com/JulieC'>
     <x xmlns='http://jabber.org/protocol/muc'/>
   </presence>

E: <presence
       from='characters@muc.example.com/JulieC'
       id='y2bs71v4'
       to='juliet@im.example.com/balcony'>
     <error type='auth'>
```

```
      <registration-required
          xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </presence>
```

remote-server-not-found

一个远程服务器或预期的接收者的JID的一部分所代表的服务不存在或不能解析(例如, 没有 _xmpp-server._tcp DNS SRV记录, A记录或AAAA记录解析也失败了, 或A/AAAA查询成功了但是在IANA注册了的端口5269上没有应答); 相关错误类型应该是"cancel".

```
C: <message
       from='romeo@example.net/home'
       id='ud7n1f4h'
       to='bar@example.org'
       type='chat'>
    <body>yt?</body>
  </message>

E: <message
       from='bar@example.org'
       id='ud7n1f4h'
       to='romeo@example.net/home'
       type='error'>
    <error type='cancel'>
      <remote-server-not-found
          xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </message>
```

remote-server-timeout

远程服务器或作为预定的接收者的全JID的一部分(或履行请求所需要的)的服务能被解析但是无法在合理的时间内与之建立通讯(例如, 无法在解析到的IP地址和端口上建立一个XML流, 或可以建立一个XML流但是因为TLS, SASL, Server Dialback的问题导致流协商失败, 等等); 相关的错误类型应该是"wait"(除非该错误是更永久性的, 例如, 远程服务器能被找到但是无法被认证或它违反了安全策略).

```
C: <message
       from='romeo@example.net/home'
       id='ud7n1f4h'
       to='bar@example.org'
       type='chat'>
    <body>yt?</body>
  </message>

E: <message
       from='bar@example.org'
       id='ud7n1f4h'
       to='romeo@example.net/home'
       type='error'>
    <error type='wait'>
      <remote-server-timeout
          xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
  </message>
```

resource-constraint

服务器或接收者忙或缺乏必要的系统资源来服务该请求; 相关的错误类型应该是"wait".

```
C: <iq from='romeo@example.net/foo'
       id='kj4vz31m'
       to='pubsub.example.com'
       type='get'>
     <pubsub xmlns='http://jabber.org/protocol/pubsub'>
       <items node='my_musings'/>
     </pubsub>
   </iq>

E: <iq from='pubsub.example.com'
       id='kj4vz31m'
       to='romeo@example.net/foo'
       type='error'>
     <error type='wait'>
       <resource-constraint
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

service-unavailable

服务器或接收者当前未提供被请求的服务; 相关的错误类型应该是"cancel".

```
C: <message from='romeo@example.net/foo'
            to='juliet@im.example.com'>
     <body>Hello?</body>
   </message>

S: <message from='juliet@im.example.com/foo'
            to='romeo@example.net'>
     <error type='cancel'>
       <service-unavailable
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </message>
```

> 安全警告: 应用程序必须返回一个<service-unavailable/>节错误(8.3.3.19)而不是<item-not-found/>(8.3.3.7)或<recipient-unavailable/>(8.3.3.13), 如果发送后面两个错误将提供关于预期的接收者的网络可用性信息给一个未被授权知道这写信息的实体(详见联机状态信息授权的讨论, 参考XMPP-IM).

subscription-required

提出请求的实体没有被授权访问所请求的服务，因为需要事先订阅(事先订阅的例子包括授权接收XMPP-IM定义的联机状态信息和用于XEP-0060定义的XMPP发布-订阅的 opt-in 数据种子); 相关的错误类型应该是"auth".

```
C: <message
       from='romeo@example.net/orchard'
       id='pa73b4n7'
       to='playwright@shakespeare.example.com'
       type='chat'>
     <subject>ACT II, SCENE II</subject>
     <body>help, I forgot my lines!</body>
   </message>
```

```
E: <message
       from='playwright@shakespeare.example.com'
       id='pa73b4n7'
       to='romeo@example.net/orchard'
       type='error'>
     <error type='auth'>
       <subscription-required
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </message>
```

undefined-condition

该错误条件不在本列表中的其他错误条件之中; 任何错误类型都可能和本条件有关, 并且除非和应用特有的条件联合在一起，它应该不被使用.

```
C: <message
       from='northumberland@shakespeare.example'
       id='richard2-4.1.247'
       to='kingrichard@royalty.england.example'>
     <body>My lord, dispatch; read o'er these articles.</body>
     <amp xmlns='http://jabber.org/protocol/amp'>
       <rule action='notify'
             condition='deliver'
             value='stored'/>
     </amp>
   </message>

S: <message from='example.org'
            id='amp1'
            to='northumberland@example.net/field'
            type='error'>
     <amp xmlns='http://jabber.org/protocol/amp'
          from='kingrichard@example.org'
          status='error'
          to='northumberland@example.net/field'>
       <rule action='error'
             condition='deliver'
             value='stored'/>
     </amp>
     <error type='modify'>
       <undefined-condition
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
       <failed-rules xmlns='http://jabber.org/protocol/amp#errors'>
         <rule action='error'
               condition='deliver'
               value='stored'/>
       </failed-rules>
     </error>
   </message>
```

unexpected-request

接收者或服务器理解这个请求但是不希望它在这个时候出现(即, 该请求顺序错了); 相关的错误类型应该是"wait"或"modify".

```
C: <iq from='romeo@example.net/foo'
       id='o6hsv25z'
       to='pubsub.example.com'
       type='set'>
     <pubsub xmlns='http://jabber.org/protocol/pubsub'>
       <unsubscribe
```

```
            node='my_musings'
            jid='romeo@example.net'/>
      </pubsub>
    </iq>

E: <iq from='pubsub.example.com'
       id='o6hsv25z'
       to='romeo@example.net/foo'
       type='error'>
     <error type='modify'>
       <unexpected-request
           xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
       <not-subscribed
           xmlns='http://jabber.org/protocol/pubsub#errors'/>
     </error>
   </iq>
```

应用特有的条件

大家知道, 一个应用可以提供应用特有的节错误信息，通过在错误元素中包含一个正确的命名空间的子元素. 典型的, 该应用特有的元素补充或进一步限定一个已定义的元素. 从而, 该<error/>元素将包含两个或三个子元素.

```
<iq id='ixc3v1b9' type='error'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <too-many-parameters xmlns='http://example.org/ns'/>
  </error>
</iq>

<message type='error' id='7h3baci9'>
  <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'>
      [ ... application-specific information ... ]
    </text>
    <too-many-parameters xmlns='http://example.org/ns'/>
  </error>
</message>
```

一个接收到它不理解的应用特有的错误条件的实体必须忽略那个条件但适当处理该错误节的其他部分.

扩展内容

尽管 message, presence, 和 IQ 节为消息,可用性,和 请求–应答 交互 提供了基本的语义, XMPP还使用XML命名空间 (见 XML-NAMES) 扩展基本的节语法来提供附加的功能.

一个 message 或 presence 节可以包含一个或多个可选的子元素来指定扩展消息含义的内容(例如, XEP-0071所述的的消息主体的XHTML格式版本), 并且一个类型为"get"或"set"的IQ节必须包含一个这样的子元素. 这样一个子元素可以使用任何名称并且必须拥有一个命名空间声明(不同于 "jabber:client", "jabber:server", 或 "http://etherx.jabber.org/streams") 来定义子元素中的数据. 这样一个子元素被成为一个 "扩展元素". 扩展元素可能被包含在节的直属子元素中，也可能包含在任何混合的层级里面.

类似的, "扩展属性" 也是允许的. 表示说: 一个节本身 (即, 一个 <iq/>, <message/>, 或 <presence/> 元素，并由 "jabber:client" 或 "jabber:server" 内容命名空间限定) 或这样一个节的任何子元素 (一个由内容命名空间限定的扩展元素或子元素) 也可以包含一个或多个由不同于内容命名空间或保留的

"http://www.w3.org/XML/1998/namespace" 命名空间的其他命名空间(包括所谓 "空命名空间"，如果属性没有如XML-NAMES所说的那样做前缀的话)限定的属性.

> 互操作性备注: 为了向后兼容和最大化互操作性, 生成节的实体应该不在节本身或被内容命名空间"jabber:client" 或 "jabber:server" 限定的节的子元素(例如, <message/> 节的<body/> 子元素)中包含这类属性.

一个扩展元素或扩展属性被称为 "扩展内容" 并且限定这样一个元素或属性的命名空间被称为"扩展命名空间".

> 参考文献: 尽管用于XMPP的扩展命名空间通常由XMPP标准化基金会(XSF)和IETF定义, 对于定义扩展命名空间来说，没有协议或IETF标准行为是必需的, 任何个体或组织都可以自由地定义XMPP扩展.

为了说明这些概念, 下面有些例子.

以下的节包含一个直接子元素，它的扩展命名空间是 'jabber:iq:roster':

```
<iq from='juliet@capulet.com/balcony'
    id='h83vxa4c'
    type='get'>
 <query xmlns='jabber:iq:roster'/>
</iq>
```

以下节包含两个不同扩展命名空间的子元素.

```
<presence from='juliet@capulet.com/balcony'>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='http://code.google.com/p/exodus'
     ver='QgayPKawpkPSDYmwT/WM94uAlu0='/>
  <x xmlns='vcard-temp:x:update'>
    <photo>sha1-hash-of-image</photo>
  </x>
</presence>
```

以下节包含两个子元素, 其中一个被 "jabber:client" 或 "jabber:server" 内容命名空间限定，另一个被一个扩展命名空间限定; 而该扩展元素包含了一个由另一个扩展命名空间限定的子元素.

```
<message to='juliet@capulet.com'>
  <body>Hello?</body>
  <html xmlns='http://jabber.org/protocol/xhtml-im'>
    <body xmlns='http://www.w3.org/1999/xhtml'>
      <p style='font-weight:bold'>Hello?</p>
    </body>
  </html>
</message>
```

实现不为扩展命名空间限定的元素生成命名空间前缀在XMPP社区是常见的(在XML社区, 这个惯例有时被称为"免前缀标准化"). 无论如何, 如果一个实现生成了这类命名空间前缀那么它必须在该节本身或该节的一个子元素中包含该命名空间的声明, 而不是在流头声明(见 4.8.4).

路由实体(典型的是服务器)处理序列化XML节的时候应该尝试保持前缀, 但是接收实体不能(MUST NOT)依赖该前缀字符串来获取任何特定的值(对于'stream'前缀的许可, 如4.8.5所述, 是这个规则的一个例外, 尽管它是用于流而不是节的).

在任何实现的特定部分对任何给定的扩展命名空间的支持都是可选的. 如果一个实体不理解这样一个命名空间, 该实体的的预期行为依赖于该实体是 (1) 接收者 还是 (2) 一个路由或递送该节给接收者的服务器.

如果一个接收者收到一个包含了不理解的元素或属性的节, 它不能(MUST NOT)尝试处理那个XML数据，而必须按以下方式处理.

- 如果一个预定的接收者受到一个message节，它的唯一子元素由不理解的命名空间限定, 那么根据XMPP应用它必须要么忽略整个节要么返回一个节错误, 节错误应该是 <service-unavailable/> (8.3.3.19).
- 如果预定的接收者接收到一个presence节，它的唯一子元素由它不理解的命名空间限定, 那么它必须忽略该子元素，把它当成没有子元素的联机状态信息节.
- 如果预定的接收者收到一个message或presence节，它包含的XML数据由它不理解的命名空间来限定, 那么它必须忽略节的由未知的命名空间限定的那部分.
- 如果预定的接收者接收到一个类型为"get"或"set"的IQ节，它包含的一个子元素由它不理解的命名空间限定, 那么实体必须返回一个类型为"error"且错误条件为<service-unavailable/>的IQ节.

如果一个服务器持有一个节，是用来递送到另一个实体的，并且这个节包含了一个该服务器不理解的子元素, 它必须不加修改地路由或递送该节到一个和本地账户关联的已连接客户端.

详细示例

本章的详细示例用来进一步说明本标准所定义的协议.

客户端-服务器示例

以下例子展示客户端和服务器协商XML流, 交换XML节, 和关闭已协商的流的XMPP数据流. 服务器是"im.example.com", 该服务器要求使用TLS, 客户端验证使用SASL SCRAM-SHA-1机制，客户端帐号是<juliet@im.example.com>而密码是"r0m30myr0m30", 并且客户端在这个流上提交了一个资源绑定请求. 我们假设在发送初始化流头之前, 客户端已经解析了_xmpp-client._tcp.im.example.com的SRV记录并已经打开一个TCP连接到已解析的IP地址和声明的端口上.

TLS

第一步: 客户端初始化流到服务器:

```
C: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

第二步: 服务器发送一个应答流头给客户端来应答:

```
S: <stream:stream
     from='im.example.com'
     id='t7AMCin9zjMNwQKDnplntZPIDEI='
     to='juliet@im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

第三步: 服务器发送流特性给客户端(在这个点上只有STARTTLS扩展, 它是强制协商的):

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

第四步: 客户端发送STARTTLS命令给服务器:

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

第五步: 服务器通知客户端允许继续:

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

第五步(替代): 服务器通知客户端STARTTLS协商失败, 关闭XML流, 并中止TCP连接(所以, 流协商处理以不成功而结束并且双方不再进入下一步):

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

第六步: 客户端和服务器尝试通过现有的TCP连接完成TLS协商(详见TLS).

第七步: 如果TLS协商成功, 客户端通过TLS保护的TCP连接初始化一个新的流到服务器:

```
C: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

第七步(替代): 如果TLS协商不成功, 服务器关闭TCP连接(所以, 流协商处理以不成功而结束并且双方不再进入下一步):

SASL

第八步: 服务器发送流头给客户端并带上任何可用的流特性来应答:

```
S: <stream:stream
     from='im.example.com'
     id='vgKi/bkYME8OAj4rlXMkpucAqe4='
     to='juliet@im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>

S: <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>SCRAM-SHA-1-PLUS</mechanism>
       <mechanism>SCRAM-SHA-1</mechanism>
       <mechanism>PLAIN</mechanism>
     </mechanisms>
   </stream:features>
```

第九步: 客户端选择一个验证机制(在这个场景中, 是 SCRAM-SHA-1), 包含初始化应答数据:

```
C: <auth xmlns="urn:ietf:params:xml:ns:xmpp-sasl"
         mechanism="SCRAM-SHA-1">
     biwsbj1qdWxpZXQscj1vTXNUQUF3QUFBQU1BQUFBTlAwVEFBQUFBQUJQVTBBQQ==
   </auth>
```

解码之后的 base 64 数据是 "n,,n=juliet,r=oMsTAAwAAAAMAAAANP0TAAAAAABPU0AA".

第十步: 服务器发送挑战:

```
S: <challenge xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
     cj1vTXNUQUF3QUFBQU1BQUFBTlAwVEFBQUFBQUJQVTBBQWUxMjQ2OTViLTY5Y
     TktNGRlNi05YzMwLWI1MWIzODA4YzU5ZSxzPU5qaGtZVE0wTURndE5HWTBaaT
     AwTmpkbUxUa3hNbVV0TkRsbU5UTm1ORE5rTURNeixpPTQwOTY=
   </challenge>
```

解码后的 base 64 数据是 "r=oMsTAAwAAAAMAAAANP0TAAAAAABPU0AAe124695b-69a9-4de6-9c30-b51b3808c59e,s=NjhkYTM0MDgtNGY0Zi00NjdmLTkxMmUtNDlmNTNmNDNkMDMz,i=4096" (实际数据中是没有换行的).

第十一步: 客户端发送一个应答:

```
C: <response xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
     Yz1iaXdzLHI9b01zVEFBd0FBQUFNQUFBQU5QMFRBQUFBQUFCUFUwQUFlMTI0N
     jk1Yi02OWE5LTRkZTYtOWMzMC1iNTFiMzgwOGM1OWUscD1VQTU3dE0vU3ZwQV
     RCa0gyRlhzMFdEWHZKWXc9
   </response>
```

解码后的 base 64 数据是 "c=biws,r=oMsTAAwAAAAMAAAANP0TAAAAAABPU0 AAe124695b-69a9-4de6-9c30-b51b3808c59e,p=UA57tM/ SvpATBkH2FXs0WDXvJYw=" (实际数据中是没有换行的).

第十二步: 服务器通知客户端成功了, 并且包含了额外的数据:

```
S: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     dj1wTk5ERlZFUXh1WHhDb1NFaVc4R0VaKzFSU289
   </success>
```

解码后的 base 64 数据是 "v=pNNDFVEQxuXxCoSEiW8GEZ+1RSo=".

第十二步(替代): 服务器返回一个SASL错误给客户端(所以, 流协商处理以不成功结束并且双方不再进行下一步):

```
S: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     <not-authorized/>
   </failure>
   </stream>
```

第十三步: 客户端初始化一个新的流到服务器:

```
C: <stream:stream
     from='juliet@im.example.com'
     to='im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>
```

资源绑定

第十四步: 服务器发送一个流头到客户端并带上支持的特性(在这个场景中, 是资源绑定)来应答:

```
S: <stream:stream
     from='im.example.com'
     id='gPybzaOzBmaADgxKXu9UClbprp0='
     to='juliet@im.example.com'
     version='1.0'
     xml:lang='en'
     xmlns='jabber:client'
     xmlns:stream='http://etherx.jabber.org/streams'>

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

在被通知资源绑定是强制协商之后, 客户端需要绑定一个资源到流; 这里我们假定客户端提交了一个自然人可读的文本字符串.

第十五步: 客户端绑定一个资源:

```
C: <iq id='yhc13a95' type='set'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <resource>balcony</resource>
     </bind>
   </iq>
```

第十六步: 服务器接受提交的资源部分并通知客户端资源绑定成功:

```
S: <iq id='yhc13a95' type='result'>
     <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
       <jid>
         juliet@im.example.com/balcony
       </jid>
     </bind>
   </iq>
```

第十六步(替代): 服务器返回错误给客户端(所以, 流协商处理以不成功结束并且双方不再进入下一步):

```
S: <iq id='yhc13a95' type='error'>
     <error type='cancel'>
       <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
     </error>
   </iq>
```

节交换

现在客户端被允许通过协商好的流发送XML节了.

```
C: <message from='juliet@im.example.com/balcony'
            id='ju2ba41c'
            to='romeo@example.net'
            type='chat'
            xml:lang='en'>
     <body>Art thou not Romeo, and a Montague?</body>
   </message>
```

如果必要, 发送者的服务器和预定的接收者的服务器协商XML流(见9.2).

预定的接收者应答, 并且消息被递送到客户端.

```
E: <message from='romeo@example.net/orchard'
            id='ju2ba41c'
            to='juliet@im.example.com/balcony'
            type='chat'
            xml:lang='en'>
     <body>Neither, fair saint, if either thee dislike.</body>
   </message>
```

客户端随后可以通过这个流继续发送和接收不限数量的XML节.

关闭

不想发送更多的消息, 客户端关闭它到服务器的流,不在等待来自服务器的入站数据了.

```
C: </stream:stream>
```

和4.4一致, 服务器可能发送额外的数据给客户端然后才关闭到该客户端的流.

```
S: </stream:stream>
```

客户端现在发送一个 TLS close_notify 警告, 从服务器接收到一个 close_notify 警告应答, 然后中止当前的TCP连接.

服务器-服务器示例

以下示例展示一个服务器和对端服务器协商XML流， 交换XML节, 和关闭已协商的流的数据流. 初始化服务器("Server1")是im.example.com; 接收服务器("Server2")是example.net 并且要求使用TLS; im.example.com递交一个证书并通过SASL EXTERNAL机制验证. 假定在发送初始化流头之前, Server1已经解析了一个SRV记录_xmpp-server._tcp.example.net并且已经打开了一个TCP连接到已解析的IP地址的声明的端口上. 注意Server1怎样声明内容命名空间"jabber:server"作为缺省的命名空间并为流相关的元素使用前缀, 反之Server2使用免前缀标准.

TLS

第一步: Server1初始化流到Server2:

```
S1: <stream:stream
      from='im.example.com'
      to='example.net'
      version='1.0'
      xmlns='jabber:server'
      xmlns:stream='http://etherx.jabber.org/streams'>
```

第二步: Server2发送一个应答流头到Server1来应答:

```
S2: <stream
      from='example.net'
      id='hTiXkW+ih9k2SqdGkk/AZi0OJ/Q='
      to='im.example.com'
      version='1.0'
      xmlns='http://etherx.jabber.org/streams'>
```

第三步: Server2发送流特性给Server1(在这个点上只有STARTTLS扩展, 它是强制协商的):

```
S2: <features xmlns='http://etherx.jabber.org/streams'>
      <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
        <required/>
```

```
      </starttls>
    </features>
```

第四步: Server1发送STARTTLS指令给Server2:

```
S1: <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

第五步: Server2通知Server1它被允许继续:

```
S2: <proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

第五步(替代): Server2通知Server1 STARTTLS协商失败了, 关闭流, 并中止TCP连接(于是, 流协商过程以不成功结束并且双方不再进行下一步):

```
S2: <failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
    </stream>
```

第六步: Server1和Server2尝试通过TCP完成TLS协商(详见TLS).

第七步: 如果TLS协商成功了, Server1在受TLS保护的TCP连接上初始化一个新流到Server2:

```
S1: <stream:stream
      from='im.example.com'
      to='example.net'
      version='1.0'
      xmlns='jabber:server'
      xmlns:stream='http://etherx.jabber.org/streams'>
```

第七步(替代): 如果TLS协商不成功, Server2关闭TCP连接(所以, 流协商过程以不成功结束并且双方不再进行下一步).

SASL

第八步: Server2发送一个应答流头给Server1并带上可用的流特性(包括优先的SASL EXTERNAL机制):

```
S2: <stream
      from='example.net'
      id='RChdjlgj/TIBcbT9Keu31zDihH4='
      to='im.example.com'
      version='1.0'
      xmlns='http://etherx.jabber.org/streams'>

S2: <features xmlns='http://etherx.jabber.org/streams'>
      <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
        <mechanism>EXTERNAL</mechanism>
      </mechanisms>
    </features>
```

第九步: Server1选择EXTERNAL机制(包含一个"="的空应答):

```
S1: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
          mechanism='EXTERNAL'>=</auth>
```

第十步: Server2返回成功:

```
S2: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

第十步(替代): Server2通知Server1验证失败了(所以, 流协商过程以不成功结束并且双方不再进行下一步):

```
S2: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <not-authorized/>
    </failure>
    </stream>
```

第十一步: Server1初始化一个新流到Server2:

```
S1: <stream:stream
      from='im.example.com'
      to='example.net'
      version='1.0'
      xmlns='jabber:server'
      xmlns:stream='http://etherx.jabber.org/streams'>
```

第十二步: Server2发送一个流头给并带上任何附加的特性(或, 在这个例子中, 一个空的特性元素)来应答:

```
S2: <stream
      from='example.net'
      id='MbbV2FeojySpUIP6J91qaa+TWHM='
      to='im.example.com'
      version='1.0'
      xmlns='http://etherx.jabber.org/streams'>

S2: <features xmlns='http://etherx.jabber.org/streams'/>
```

节交换

现在Server1被允许通过已协商的从im.example.com到example.net的流发送XML节给Server2; 这里我们假定被传输的节就是前面演示的那些客户端-服务器通讯的节, 尽管是在一个服务器-服务器由'jabber:server'命名空间限定的流上.

Server1发送XML节给Server2:

```
S1: <message from='juliet@im.example.com/balcony'
             id='ju2ba41c'
```

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

关闭

想不再发送更多消息, Server1关闭它到Server2的流但是等待从Server2的入站数据. (实践中, 流大部分时候保持打开一段时间, 因为Server1和Server2不是立刻知道流是否需要更多通讯.)

```
S1: </stream:stream>
```

和建议的流关闭握手一致, Server2同样关闭流:

```
S2: </stream>
```

Server1现在发送一个TLS close_notify警告, 从Server2接收一个close_notify警告应答, 然后中止当前的TCP连接.

处理XML节的服务器规则

每个服务器实现将包含它自己的处理接受的节的逻辑. 这写逻辑决定服务器是需要路由一个给定的节到其他域, 还是把它递送到一个本地实体(典型的是一个和本地帐号相关联的已连接客户端), 或者直接由服务器本身处理它. 本章提供处理XML节的通用规则. 然而, 特殊的XMPP应用可以定义递送规则来修改或补充以下规则(例如, 定义于XMPP-IM的用于即时消息和联机状态信息应用的一系列递送规则).

顺序处理

一个XMPP服务器必须确保它从一个已连接的客户端或远端服务器的给定的入站流上收到的节和其他XML元素的顺序处理.

顺序处理适用于 (a) 任何用来协商和管理XML流的XML元素, 和 (b) 所有XML节, 包括但不限于以下情况:

1. 由一个客户端发送到它的服务器或它自己的纯JID的由服务器直接处理的节(例如, XMPP-IM所述的获取好友名册和初始化出席信息的顺序处理).
2. 由一个已连接的客户端发送的节并且是用来递送到另一个和该服务器相关的实体(例如, 从地址 <juliet@im.example.com> 发到 <nurse@im.example.com> 的节). 服务器必须确保按照从发送的客户端的入站流上接收到节的顺序来递送那些到预期接收者的节, 对递送目的接收者地址为纯JID和全JID的节等同对待.
3. 由一个已连接的客户端发送的节并且是用来递送到一个位于远端服务器上的实体(例如, 从地址 <juliet@im.example.com> 发到 <romeo@example.net>). 路由的服务器必须确保按照从发送的客户端的入站流上接收到节的顺序来路由那些到预期接收者的节, 对路由目的接收者地址为纯JID和全JID的节等同对待. 为了帮助确保顺序处理, 路由服务器必须通过单一的到远端域的出站流来路由这些节, 而不是在一个服务器－服务器流上发送一些节而在另一个服务器－服务器流上发送另一些节.

4. 从一个服务器路由到另一个服务器的递送到远端服务器的一个相关实体的节(例如, 从地址 <juliet@im.example.com> 发到 <romeo@example.net> 并且由 <im.example.com> 在一个服务器－服务器流上路由到 <example.net>). 递送服务器必须确保按照它从路由服务器的入站流上接收到的顺序来递送节, 对递送目的接收者地址为纯JID和全JID的节同等对待.
5. 由一个服务器发送到另一个服务器用于远端域直接处理的节(例如, 从 <im.example.com> 发到 <example.net>).

如果服务器对于特定请求的处理对它接下来可能从入站流收到的数据的处理有影响(例如, 强制性的通讯策略), 它必须挂起接下来的数据的处理，直到它已经处理完该请求.

顺序处理只适用于单一入站流. 所以, 服务器不负责保证它从相同本地帐号(例如, 通过两个不同入站流从<juliet@im.example.com/balcony> 和 <juliet@im.example.com/chamber>接收到的节) 或相同的远端域(例如, 由一个远端域通过两个不同的入站流进行的流协商; 然而, 服务器可以以<conflict/>流错误来关闭该流(4.9.3.3),如果远端服务器尝试协商多个流, 详见4.9.3.3)通过多个入站流接收到的数据的相干性.

一般注意事项

在顶层, 服务器处理XML节有三个主要的注意事项, 它们有时候相左但需要一个一致的办法来管理:

1. 如果可能最好把一个节递送到预定的接收者.
2. 如果一个节不能被递送, 通知发送者是有帮助的.
3. 促成目录获取攻击(13.11)和联机状态泄露(13.10.2)是不好的.

对于可能的和递送相关的攻击, 以下几点需要牢记:

1. 从攻击者的视角来看, 在服务器 (i) 递送节或存储成离线消息稍后递送(见 [{RFC6121|XMPP-IM]]) 和 (ii) 安静地忽略它(因为在任何那种用例之下都不会立刻返回一个错误)之间是否有任何一点有效的不同; 因此, 在服务器递送一个节或吧这个节离线存储稍后递送的场景中, 如果那个帐号不存在，它需要安静地忽略这个节.
2. 服务器如何处理发送到一个受到目录获取影响的纯JID<localpart@domainpart>的节, 因为如果服务器根据一个给定的纯JID是否有帐号来做出不同应答，那么攻击者可以确定一个帐号是否存在.
3. 服务器如何处理发送到一个受到出席信息泄露影响的全JID的节, 因为可以发送请求到多个全JID上并且根据该用户是否有一个全JID的设备在线来收到不同的应答. 使用随机的资源部分(是由客户端还是服务器生成参见第七章)明显减轻了这个攻击, 所以在某种程度上它被关心的程度地狱目录获取攻击.

很自然, 如果收到从一个用户的服务器返回的错误的实体已经知道用户的出席信息或被授权知道它，那么出席信息就没有被泄露(例如, 所谓出席信息订阅或直接的出席信息), 如果服务器返回一个错误给一个已经知道某用户存在的实体，那么该服务器没有受到目录获取攻击(例如, 因为实体在用户的联系人列表里); 这些事情的更完整的讨论见XMPP-IM.

没有'to'地址

如果节没有'to'属性, 服务器必须代表发送它的实体来直接处理它, 这里"直接处理它"的含义依赖于这个节是 message, presence, 还是 IQ. 因为从其它服务器收到的所有节必须拥有'to'属性, 这个规则只适用于从连接到该服务器的本地实体(通常是一个客户端)收到的节.

Message

如果该服务器接收到一个没有'to'属性的message节, 它必须把这个消息当作'to'地址是发送实体的纯JID <localpart@domainpart> 那样来处理.

Presence

如果该服务器接收到一个没有'to'属性的presence节, 它必须广播这个节到订阅了发送者实体的出席信息的实体, 如果适用的话(XMPP-IM定义了出席信息应用的这类广播的语义).

IQ

如果服务器接收到一个没有'to'属性的IQ节, 它必须代表发送这个节的帐户处理这个节, 如下:

1. 如果IQ节的类型为"get"或"set"并且服务器理解限定载荷的命名空间, 该服务器必须代表发送实体处理该节或返回一个适当的错误给发送实体. 尽管"处理"的含义是由限定的命名空间的语义来决定的, 通常服务器将返回一个适当的类型为"result"或"error"的IQ节来应答类型为"get"或"set"的节, 应答就好像服务器就是发送实体的纯JID那样. 作为一个例子, 如果发送实体发送一个类型为"get"而载荷由'jabber:iq:roster'命名空间限定(参见XMPP-IM)的IQ节, 那么服务器将返回和该发送实体的纯JID相关的roster给请求roster的发送实体的特定资源.
2. 如果IQ节的类型为"get"或"set"并且服务器不理解限定该载荷的命名空间, 服务器必须返回一个错误给发送实体, 这个错误必须是<service-unavailable/>.
3. 如果IQ节的类型是"error"或"result", 该服务器必须根据该IQ相关的载荷或类型是"get"或类型是"set"处理这个error或result(如果没有相关的节, 服务器必须忽略该error或result节).

远程域

如果JID的域部分包含一个'to'属性但是和该服务器配置的任何一个完整合格域名(FQDNs)都不匹配, 该服务器应该尝试路由该节到远端域(服从本地服务器关于域间通讯的规定和安全策略, 因为对于任何给定的部署来说这类通讯是可选的). 如下章所述, 有两个可能的用例.

> 安全警告: 这些规则仅适用于客户端-服务器流. 如8.1.1.2所述, 如果'to'属性里的JID的域部分和接收服务器的完全合格域名(FQDN)不匹配，那么服务器不能(MUST NOT)在一个服务器-服务器流上接收节.

现有流

如果一个服务器-服务器流已经存在于两个域之间, 发送者的服务器应该尝试在现有的流上为远端服务器路由这个节到授权服务器.

无现有流

如果在两个域之间无现有的服务器-服务器流, 发送者的服务器将如下处理:

1. 解析远端域的完整合格域名(FQDN)(如13.9.2所述).
2. 在两个域之间协商服务器-服务器流(如第五章和第六章所述).
3. 在新建的流上为远端域路由该节到授权服务器.

错误处理

如果一个到预期接收者服务器的节的路由不成功, 发送者的服务器必须返回一个错误给发送者. 如果远端域的解析不成功, 节错误必须是 <remote-server-not-found/> (8.3.3.16). 如果成功解析但是流无法协商, 节错误必须是<remote-server-timeout/> (8.3.3.17).

如果和期望的接收者的服务器的流协商成功了但是远端服务器不能递送该节到接收者, 远端服务器必须通过发送者的服务器返回一个适当的错误给发送者.

本地域

如果包含在'to'属性中的JID的域部分和该服务器的完整合格域名(FQDN)匹配, 该服务器必须首先确定该FQDN是由服务器本身还是由特定的本地服务来提供服务. 如果是后者, 该服务器必须路由该节到那个服务. 如果是前者, 该服务器必须执行接下来的动作. 无论如何, 该服务器不能(MUST NOT)路由或"转发"该节到另一个域, 因为处理所有'to'地址的域部分和该服务器的已配置FQDN匹配的节是服务器的责任(除此之外, 这还可以防止循环处理).

域部分

如果包含在'to'属性里的JID的格式为<domainpart>, 那个么该服务器必须要么 (a) 根据节类型做适当的处理，要么 (b) 返回一个错误节给发送者.

域部分/资源部分

如果包含在'to'属性中的JID的格式为<domainpart/resourcepart>, 那么该服务器必须要么 (a) 根据节类型做适当处理，要么 (b) 返回一个节错误给发送者.

本地部分@域部分

这种类型的地址通常和该服务器上的一个帐号相关. 以下规则提供一些通用方针; 更多即时消息和出席信息场景中的详细规则参见XMPP-IM.

没有此用户

如果没有本地帐号和<localpart@domainpart>相关, 节被如何处理取决于节的类型.

1. 对于message节 该服务器必须要么 (a) 安静地忽略该节，要么 (b) 返回一个<service-unavailable/>节错误(8.3.3.19)给发送者.
2. 对于presence节, 该服务器应该忽略该节(或如XMPP-IM所述那样表现).
3. 对于IQ节, 该服务器必须返回一个<service-unavailable/>节错误(8.3.3.19)给发送者.

用户存在

如果包含在'to'属性中的JID的格式为<localpart@domainpart>, 节如何被处理取决于节类型.

1. 对于message节, 如果该帐号存在至少一个已连接的资源那么该服务器应该递送这个节到至少一个已连接的资源. 如果不存在已连接的资源，那么该服务器必须要么 (a) 离线存储该message等该帐号下次有已连接资源时递送，要么 (b) 返回一个<service-unavailable/>节错误(8.3.3.19).
2. 对于presence节, 如果存在一个已连接的资源曾经发送过初始presence (即, 有一个如XMPP-IM所述的"presence会话")，那么该服务器应该递送该节到那个资源. 如果不存在已连接的资源，那么服务器应该忽略该节(或如XMPP-IM那样表现).
3. 对于IQ节, 服务器必须直接代表预期的接收者来处理它.

本地部分@域部分/资源部分

如果包含在'to'属性中的JID的格式为<localpart@domainpart/resourcepart>而且该用户存在但是没有已连接的资源能和该全JID精确匹配, 该节应该如10.5.3.2所述的JID格式为<localpart@domainpart>时那样地被处理.

如果包含在'to'属性中的JID的格式为<localpart@domainpart/resourcepart>, 该用户存在, 并且有一个已连接的资源和该全JID精确地匹配, 该服务器必须递送该节到那个已连接的资源.

XML用法

XML限制

XMPP定义了一类名为XML流的数据对象以及处理XML流的计算机程序的行为. XMPP是一个应用配置或XML的受限格式, 并且一个完整的XML流(包括开始和结束的流标签)就是一个合格的XML文档.

然而, XMPP不处理XML文档而是处理XML流. 因为XMPP不需要随意解析以及完成XML文档, XMPP不需要支持XML的完整功能集. 此外, XMPP使用XML来定义协议数据结构和扩展，用于网络实体之间的结构化交互，所以坚持那些XML-GUIDE中关于在IETF协议中对XML使用的限制中提供的建议. 结果是, XML的以下特性在XMPP中被禁止:

- 注释(定义于XML的2.5节)
- 处理指令(其中的2.6节)
- 内部或外部的DTD子集(其中的2.8节)
- 预定义实体的例外(其中的4.6节)的内部或外部的实体参考(其中的4.2节)

一个XMPP实现对这些特性必须做出如下的表现:

1. XMPP实现不能(MUST NOT)插入匹配这类特性的字符串到一个XML流中.
2. 如果一个XMPP实现从XML流收到匹配此类特性的字符串, 它必须以一个流错误来关闭流, 这个错误应该是<restricted-xml/>(4.9.3.18), 不过某些现有的实现发送的是<bad-format/>(4.9.3.1).

XML命名空间名字和前缀

XML命名空间(见XML-NAMES)被XMPP流用来创建数据所有权的严格界限. 命名空间的基本功能是把结构化地混合在一起的XML元素的不同词汇区分开来. 确保XMPP流感知命名空间使得任何允许的XML在XMPP中都能以任何数据元素被结构化地混合在一起. XMPP对于XML命名空间名字和前缀的特有规则定义于4.8(对于XML流)和8.4(对于XML节).

良好格式

在XML中, 有两种良好格式:

- 符合XML的2.1节中"良好格式"定义的"XML良好格式".
- 符合XML-NAMES的第七章中"命名空间的良好格式"的"命名空间良好格式".

以下规则适用:

1. XMPP实体不能(MUST NOT)生成不符合XML良好格式的数据.
2. XMPP实体不能(MUST NOT)接受不符合XML良好格式的数据; 反之它必须以<not-well-formed/>流错误(4.9.3.13)来关闭接收到那个数据的流.

3. XMPP实体不能(MUST NOT)生成不符合命名空间良好格式的数据.
4. XMPP实体不能(MUST NOT)接受不符合命名空间良好格式的数据(特别是, XMPP服务器不能(MUST NOT)路由或递送不符合命名空间良好格式的数据); 反之它必须要么返回一个<not-acceptable/>节错误(8.3.3.9)要么以一个<not-well-formed/>流错误(4.9.3.13)来关闭该流, 这里最好是以一个流错误关闭该流， 因为i接受这类数据可能会把一个实体暴露给特定的拒绝服务攻击.

互操作性备注: 因为这些约束是在RFC3920下就已经确认的, 有可能基于那个协议的实现会发送一些不遵守这些约束的数据.

验证

服务器不负责确保递送到一个已连接的客户端或路由到一个对端服务器的XML数据是有效的, 符合XML的2.8节中提供的"有效"的定义. 一个实现可以选择只接受或发送已经通过本文提供的schema被显式地验证过的数据, 但是这一行为是可选的. 建议客户端不要依靠这个能力来发送不符合schema的数据, 并且应该忽略入站XML流中任何不符的元素或属性.

参考文献: 术语"有效"("valid")和"良好格式"("well-formed")在XML中是不同的.

包含XML声明

在发送一个流头之前, 实现应该发送一个XML声明(和从XML产生的"XMLDecl"匹配). 应用必须遵从XML提供的关于XML声明的格式和哪些场景需要包含XML声明的规则.

因为外部标记声明在XMPP中是被禁止的(如11.1所述), 独立的文档声明(和来自XML产生的"SDDecl")将是无意义的并且因此不能(MUST NOT)被包含在一个在XML流上发送的XML声明中. 如果XMPP实体接收到一个包含了独立的文档声明的XML声明并且值为"no", 该实体必须要么忽略这个独立的文档声明, 要么以一个流错误来关闭这个流, 这个流错误应该是<restricted-xml/>(4.9.3.18).

字符编码

实现必须支持通用字符集UCS2字符串的UTF-8改造, 定义于CHARSETS和定义于UTF-8中的要保持一致性. 实现不能(MUST NOT)试图使用其他编码. 如果一方在一个XML流中探测到另一方试图发送编码不是UTF-8的XML数据, 它必须以一个流错误关闭这个流, 这个流错误应该是<unsupported-encoding/>(4.9.3.22), 虽然某些现有的实现发送的是<bad-format/>(4.9.3.1).

因为对于XMPP实现来说支持所有并且仅有UTF-8编码是强制性的并且因为UTF-8总是有相同的字节顺序, 实现不能(MUST NOT)在数据流的起始部分发送字节顺序标记("BOM"). 如果实体在一个XML流的任何位置(包括该流的第一个字符)接收到UNICODE字符 U+FEFF , 它必须把该字符当成一个零宽度的不中断的空白, 而不是任何字节顺序标记.

空格

除了明确不允许(例如, 在TLS协商和SASL协商的时候)的情况, 实体可以在XML节之间或在任何其他在流上发送的顶层元素之间发送空格作为分隔符. 发送这类空格的一个常见用法参见4.4.

XML版本

XMPP是一个XML 1.0的应用范本. XMPP的更多版本可以被定义于更高版本的XML, 但是本协议定义的XMPP只限于XML 1.0.

国际化事项

如11.6所定义的, XML流必须以UTF-8编码.

如4.7所定义的, XML流应该包含一个'xml:lang'属性来为期望展现给自然人用户的任何XML字符数据指定缺省语言. 如8.1.5所定义的, XML节应该包含一个'xml:lang'属性，如果该节包含期望展现给自然人用户的XML字符数据的话. 服务器应该代表连接的实体应用缺省的'xml:lang'属性于到路由或递送的节, 并且不能(MUST NOT)修改或删除从其他实体收到的节的'xml:lang'属性.

XMPP地址的国际化事项定义于XMPP-ADDR.

安全事项

基本情况

XMPP技术典型的被部署在非中心化的客户端-服务器体系架构中. 结果是, 当两个XMPP实体需要通讯时有很多可能的途径:

1. 双方实体都是服务器. 这种情况下, 实体可以在它们之间直接建立 服务器-服务器 流.
2. 一个实体是服务器而另一个实体是一个客户端并且该客户端的帐号由该服务器提供. 这种情况下, 实体可以在它们之间直接建立 客户端-服务器 流.
3. 双方实体都是客户端并且它们的帐号由同一个服务器提供. 这种情况下, 实体不能在它们之间建立直接的流, 但是在它们之间只有一个中间实体, 它们能理解该实体的策略并且它们对它有同等的信任级别(例如, 该服务器可能要求所有客户端连接使用传输层安全(TLS)).
4. 双方实体都是客户端但是它们的帐号由不同服务器提供. 这种情况下, 实体不能在它们之间建立直接的流并且它们之间有两个中间实体; 每个客户端可能对它的帐号所在的服务器有一定的信任度但是可能不知道另一个客户端连接的服务器的策略.

本协议仅覆盖了两个服务器或客户端和服务器之间的直接XML流的安全性(情景#1和情景#2), 这里每个流可以被认为是一个通讯路径中的一"跳". 对于一个多跳路径(情景#3和情景#4)的安全性的目标来说, 虽然非常需要, 但是超出了阿本协议的范围.

按照SEC-GUIDE, 本协议覆盖了通讯安全性(保密性, 数据完整性, 以及对端实体验证), 不可抵赖性, 和系统安全性(未经授权的使用, 不适当的使用, 和拒绝服务). 我们也讨论了常见的安全问题，类似信息泄露, 防火墙, 和目录获取, 以及和技术重用相关的最佳实践，类似 base 64, DNS, 加密哈希函数, SASL, TLS, UTF-8, 和XML.

威胁模型

对XMPP的威胁模型是基于SEC-GUIDE中描述的标准"互联网威胁模型". 假定攻击者感兴趣的是针对未受保护的XMPP系统做出以下攻击:

- 窃听
- 猜密码
- 通过字典攻击来破解密码
- 通过字典获取攻击来发现用户名
- 重发, 插入, 删除, 或修改节
- 欺骗用户
- 增加未授权的条目到服务器或帐号
- 不适当地使用一个服务器或帐号

- 拒绝服务其他实体
- 通过"中间人"攻击来破坏通讯流
- 在路过的服务器上增加控制

接下来一些相应的章节会描述如何防止这些威胁.

层顺序

协议被堆叠的层的顺序如下:

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

这个顺序有重要的安全影响, 如各项安全事项所述.

在XMPP内, XML节位于XML流之上, 如第四章所述.

保密性和完整性

对传输层安全(TLS)的适当密码组的使用提供了一个可靠的机制来确保保密性和在客户端和服务器之间以及两个服务器之间交换的数据的完整性. 所以, TLS能有助于阻止窃听, 密码嗅探, 中间人攻击, 和XML流上的节重发, 插入, 删除, 和修改. XMPP客户端和服务器必须如第五章定义的那样支持TLS.

> 参考文献: 流的保密性和完整性可以有不同于TLS的保护方法, ; 例如, 在安全层协商中引入一个SASL机制.

> 安全警告: 在XMPP中对TLS的使用适用于单个流. 因为XMPP典型地使用分布式的客户端–服务器体系架构来部署(详见2.5), 一个节可能经历多个流, 并且不是所有的那些流都是被TLS保护的. 例如, 某个以一个服务器上的会话从客户端发送的节(例如, <romeo@example.net/orchard>) 并且期望递送到一个另一台服务器上的会话的客户端(例如, <juliet@example.com/balcony>) 将经历三个流: (1) 从发送者的客户端到它的服务器的流, (2) 发送者服务器到接收者服务器的流, 和 (3) 从接收者的服务器到接收者的客户端的流. 此外, 该节将被作为明码被发送者的服务器和接收者的服务器处理. 所以, 即使从发送者的客户端到它的服务器的流是受保护的, 当一个节被发送者的服务器处理的时候，从发送者的服务器发送到接收者的服务器的时候，被接收者处理的时候，或从接收者的服务器发送到接收者的客户端的时候，在那个流上被发送的节的保密性和完整性也不能被保证. 只有一个用于端到端的加密鲁棒性技术能确保一个节的保密性和完整性，即使该节在一个通讯路径中经历了所有"跳"(例如, 一个满足E2E-REQS定义的需求的技术). 不幸的是, XMPP社区尚未产生一个适合广泛实现和部署的端到端加密技术, 并且这类技术的定义超出了本文的范围.

对端实体验证

用于验证的简单验证和安全层(SASL)的使用为对端实体验证提供了一个可靠的机制. 所以, SASL帮助防止用户欺骗, 未授权使用, 和中间人攻击. XMPP客户端和服务器必须如第六章所述支持SASL.

强安全性

STRONGSEC定义了"强安全性"并且它对于在互联网上的通讯非常重要. 为了让XMPP在客户端–服务器和服务器–服务器流上通讯, 术语"强安全性"使用安全技术来同时提供互相验证和完整性检查(例如, 一个TLS加密和使用适当的SASL机制进行SASL验证的组合).

实现必须支持强安全性. 服务供应者应该使用强安全性.

实现应该能在该部署中让一个终端用户或服务管理员对一个已连接的实体(客户端或服务器)展示证书做特定信任动作(译者注：即做是否信任该证书的确认动作); 当一个应用提供了那些, 它就不能(MUST NOT)使用通用PKI信任库来验证连接的实体了. 更多关于证书验证的详细规则和指南在下一节提供.

初始化流和应答流必须被安全地分开, 尽管可以通过机制提供双向验证来建立双向安全性.

证书

XML流的通道加密使用第五章所述的传输层安全性, 和某些情况下也使用第六章描述的验证, 通常是基于一个由接收实体(或者, 在相互证书验证的情况下, 既有接收实体又有初始化实体)展示的PKIX证书. 本节描述的重点是由XMPP实体展示的PKIX证书的生成和验证的最佳实践.

通常, 接下来的章节依靠并扩展X509在PKIX范本以及在TLS-CERTS中提供的规则和指南. 读者请参阅那些协议以便详细理解PKIX证书和它们在TLS中的的使用.

证书生成

通用事项

以下规则适用于被发行给XMPP服务器或客户端的终端实体的公共密钥证书:

1. 证书必须符合PKIX.
2. 证书不能(MUST NOT)包含一个cA逻辑值为真的basicConstraints扩展.
3. subject字段不能(MUST NOT)为空.
4. signatureAlgorithm应该是 PKCS #1 版本 1.5 使用 SHA-256 的签名算法，定义于PKIX-ALGO, 或如果可能则使用更强的算法.
5. 证书应该包含一个授权信息访问(AIA)扩展，它指定一个在线证书状态协议地址OCSP应答者(否则, 依赖的一方需要转去使用证书吊销列表(CRLs)，如OPKIX所述).

以下规则适用于发行证书给XMPP终端实体的权威认证(CA)证书:

1. 证书必须符合PKIX.
2. 证书必须包含一个设置了digitalSignature比特值的keyUsage扩展.
3. subject字段不能(MUST NOT)为空.
4. signatureAlgorithm应该是 PKCS #1 版本 1.5 使用 SHA-256 的签名算法，定义于PKIX-ALGO, 或如果可能则使用更强的算法.
5. 对于公共密钥证书的发行者, 该发行者的证书必须包含一个cA逻辑值为真的basicConstraints扩展.

服务器证书

规则

在一个由XMPP服务器展示的PKIX证书里(即, 一个"服务器证书"), 该证书应该包含一个或多个和位于该服务器上的XMPP服务相关的XMPP地址(即, 域部分). 这些定义于TLS-CERTS的规则和指南适用于XMPP服务器证书, 有以下XMPP特有的事项:

- 在XMPP客户端和服务器软件实现中要求支持 DNS-ID 标识类型 PKIX. 发行XMPP特有证书的认证机构必须支持 DNS-ID 标识类型. XMPP服务提供者应该在证书请求中包含 DNS-ID 标识类型.

- 在XMPP客户端和服务器软件实现中要求支持 SRV-ID 标识类型 PKIX-SRV(为了验证的目的，XMPP客户端实现只需要支持"_xmpp-client"服务类型，而XMPP服务器实现需要同时支持"_xmpp-client" 和 "_xmpp-server" 服务类型). 发行XMPP特有证书的认证机构必须支持 SRV-ID 标识类型. XMPP服务提供者应该在证书请求中包含 SRV-ID 标识类型.
- 为了向后兼容，在XMPP客户端和服务器软件实现中鼓励支持 XmppAddr 标识类型(定义域 13.7.1.4), 但是不再鼓励它出现在认证机构发布的证书中或XMPP服务提供者请求的证书中.
- 服务器证书中的DNS域名可以包含通配符 '*' 来完成标识中的最左边的标签.

示例

对我们的第一个例子(相对简单), 考虑一个名为"Example Products, Inc."的公司，它在"im.example.com"跑了一个XMPP服务(即, 该服务器上的用户地址格式为"user@im.example.com"), 而查询到的用于"im.example.com"的xmpp-client和xmpp-server服务的SRV记录位于在一台机器上, 名为"x.example.com", 如下:

```
_xmpp-client._tcp.im.example.com. 400 IN SRV 20 0 5222 x.example.com
_xmpp-server._tcp.im.example.com. 400 IN SRV 20 0 5269 x.example.com
```

由x.example.com出示的证书包含以下陈述:

- SRVName (id-on-dnsSRV)的一个otherName类型包含一个IA5String (ASCII) 字符串"_xmpp-client.im.example.com"
- SRVName (id-on-dnsSRV)的一个otherName类型包含一个IA5String (ASCII) 字符串"_xmpp-server.im.example.com"
- dNSName包含一个ASCII字符串"im.example.com"
- XmppAddr (id-on-xmppAddr)的一个otherName类型包含一个UTF-8字符串"im.example.com"
- CN包含一个ASCII字符串"Example Products, Inc."

对我们的第二个例子(复杂一些), 考虑一个名为"Example Internet Services"的ISP. 它在"example.net"一个XMPP服务(即, 该服务器上的用户地址格式为"user@example.net"), 但是查询到的用于"example.net"的xmpp-client和xmpp-server服务的SRV记录位于在两台机器上("x1.example.net"和"x2.example.net"), 如下:

```
_xmpp-client._tcp.example.net. 68400 IN SRV 20 0 5222 x1.example.net.
_xmpp-client._tcp.example.net. 68400 IN SRV 20 0 5222 x2.example.net.
_xmpp-server._tcp.example.net. 68400 IN SRV 20 0 5269 x1.example.net.
_xmpp-server._tcp.example.net. 68400 IN SRV 20 0 5269 x2.example.net.
```

Example Internet Services 也在chat.example.net上跑了一个聊天室服务, 并且为该服务提供了一个 xmpp-server SRV 记录(从而允许来自远端域的实体可以访问该服务). 它也可能在将来提供其他类似服务, 所以它希望在它的证书里展示一个通配符来处理这类发展.

由x1.example.net或x2.example.net出示的证书包含以下陈述:

- SRVName (id-on-dnsSRV)的一个otherName类型包含一个IA5String (ASCII) 字符串"_xmpp-client.example.net"
- SRVName (id-on-dnsSRV)的一个otherName类型包含一个IA5String (ASCII) 字符串"_xmpp-server.example.net"

- SRVName (id-on-dnsSRV)的一个otherName类型包含一个IA5String (ASCII) 字符串"_xmpp-server.chat.example.net"
- dNSName包含一个ASCII字符串"example.net"
- dNSName包含一个ASCII字符串"*.example.net"
- XmppAddr (id-on-xmppAddr)的一个otherName类型包含一个UTF-8字符串"example.net"
- XmppAddr (id-on-xmppAddr)的一个otherName类型包含一个UTF-8字符串"chat.example.net"
- CN包含一个ASCII字符串"Example Internet Services"

客户端证书

在一个自然人用户控制的XMPP客户端展示的PKIX证书里(即, 一个 "客户端证书"), 建议该证书包含一个或多个和XMPP用户相关的JIDs. 如果包含了一个JID, 该JID必须被展示为一个如13.7.1.4所述的的XmppAddr.

XmppAddr标识类型

XmppAddr标识类型是一个UTF8字符串，在subjectAltName里带了一个otherName实体, 使用ASN.1对象标识"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
```

作为一个替代"id-on-xmppAddr"的标记, 该对象标识符可以被展示为带点分隔的显示格式(即, "1.3.6.1.5.5.7.8.5")或定义于URN-OID的统一资源名称标记(即, "urn:oid:1.3.6.1.5.5.7.8.5").

所以在证书中包含JID <juliet@im.example.com> 的例子可以被用以下三种方法中的任何一种进行格式化:

id-on-xmppAddr:

subjectAltName=otherName:id-on-xmppAddr;UTF8:juliet@im.example.com

dotted display format:

subjectAltName=otherName:1.3.6.1.5.5.7.8.5;UTF8:juliet@im.example.com

URN notation:

subjectAltName=otherName:urn:oid:1.3.6.1.5.5.7.8.5;UTF8:juliet@im.example.com

推荐在证书生成中使用"id-on-xmppAddr"格式, 但是在证书验证的时候必须支持三种格式.

为了显式地定义和限制XMPP网络中一个证书的期望的使用,"id-on-xmppAddr"对象标识符可被用于插入PKIX的4.2.1.12]]定义的扩展密钥.

证书验证

当对端如第五章和第六章所述，为了加密或验证XML流而向一个XMPP实体出示一个服务器证书或客户端证书, 该实体必须尝试验证这个证书来决定该证书是否被视为一个"受信任的证书", 即, 一个可接受用于加密和/或验证的证书要符合该XMPP实体的本地服务策略或已配置的设置.

对于服务器证书和客户端, 验证的实体都必须做以下的事情:

1. 尝试验证该证书的完整性.
2. 尝试验证该证书已经被发行的认证机构正确地签名.
3. 尝试验证完整的证书路径.
4. 检查13.7.1.1定义的常规情况下终端实体的公开密钥和认证机构证书的规则或13.7.1.2和13.7.1.3分别定义的服务器证书和客户端证书.
5. 通过证书收回列表(CRLs)或在线证书状态协议OCSP检查证书收回消息, 或两者都检查.

如果那些验证尝试中的任何一种失败了, 验证实体必须单方面中止该会话.

接下来的章节描述适用于服务器–服务器流和客户端–客户端流的额外的身份验证规则.

一旦流的一端的身份被验证了, 该验证实体应该也把已验证的身份关联到它从对端接收到的流头的'from'地址(如果有的话). 如果两个身份不匹配, 验证实体应该中止连接尝试(无论如何, 可能有好的原因导致身份不匹配, 如4.7.1所述).

服务器证书

对服务器证书, 适用定义于TLS-CERTS的规则和指南, 条件是13.7.1.4定义的XmppAddr标识符被允许作为一个参考标识符.

标识符的检查设置如下:

- 初始化实体把它的参考标识符的源域设为它在初始化流头中通讯的'to'地址; 即, 这是它希望接收实体在一个PKIX证书里提供的身份.
- 接收实体把它的参考标识符的源域设为初始化实体在初始化流头中与它通讯的'from'地址; 即, 这是初始化实体试图确认的.

在服务器－服务器通讯的情形中, 当(初始化实体)验证来自一个对端服务器的入站服务器－服务器连接时，TLS-CERTS中描述的匹配过程由一个应用服务器(接收实体)来执行. 在这种情况下, 接收方实体验证初始化实体的身份并把初始化实体在初始化流头中的'from'属性中声明的DNS域名用作它的参考身份的源域. 然而, TLS-CERTS描述的匹配过程保持不变并适用于同一方法.

客户端证书

当一个XMPP服务器验证一个客户端出示的证书时, 有三种可能的情况, 下面的章节将会描述.

被检查的身份设置如下:

- 该客户端把它的参考身份的源域设为它在初始化流头中通讯的'to'地址; 即, 这个身份是它期望服务器在一个PKIX证书中提供的.
- 该服务器把它的参考身份的纯JID设为初始化流头中与之通讯的'from'地址; 即, 这个身份是该客户端尝试断言的.

场景#1

如果似乎被一个证书路径来验证的客户端证书在一个信任锚中止了(如PKIX的6.1所述), 该服务器必须为13.7.1.4描述的XmppAddr的任何实例检查证书. 有三种可能的子场景:

子场景#1:

> 该服务器找到一个XmppAddr，它的展示的JID的域部分和该服务器配置的FQDNs(完整合格域名)之一相匹配; 该服务器应该把这个展示的JID当作该客户端的已验证身份.

子场景#2:

> 该服务器找到多个XmppAddr，它们展示的JID的域部分和该服务器配置的FQDNs(完整合格域名)之一相匹配; 该服务器应该使用这些展示的JID中的一个作为该客户端的已验证身份, 对它们的选择基于初始化流头中'from'地址的的纯JID(如果有的话), 基于包含在初始化流头的'to'地址的域部分, 或遵循本地服务策略(例如在一个基于该客户端证书中其他信息的用户数据库中查找).

子场景#3:

> 该服务器没找到XmppAddrs, 或至少找到一个XmppAddr但是其展示的JID的域部分和该服务器的FQDNs(完整合格域名)中的任何一个都不匹配; 该服务器不能(MUST NOT)使用该展示的JID(如果有)作为该客户端的已验证身份，而必须使用符合本地服务策略(例如在一个基于该客户端证书中其他信息的用户数据库中查找)的其他办法来验证该客户端. 如果该身份不能这样被验证, 该服务器可以中止验证过程并终止TLS协商.

场景#2

如果该客户端证书是被一个服务器不知道的证书颁发机构认证的, 该服务器必须按场景#1的子场景#3处理.

场景#3

如果该客户端证书是自签名的, 该服务器必须按场景#1的子场景#3处理.

在长连接流中检查证书

因为XMPP使用长连接XML流, 在流协商期间出示的证书可能会超时或在流仍处于活跃状态的时候被撤销(这特别对应于服务器–服务器流的上下文中). 所以, 每一方对长连接流应该:

1. 缓存其他方出示的证书以及该证书依赖的任何证书(类似一个根证书或一个认证机构的中间证书)的过期日期, 并且当任何这类证书过期的时候以一个流错误 <reset/> (4.9.3.16)关闭流.
2. 根据在其他方出示的证书以及该证书依赖的任何证书(类似一个根证书或一个认证机构的中间证书)的授权信息访问(AIA)扩展定期查询在线证书状态协议OCSP应答者列表, 并且当任何这类证书被撤销的时候以一个流错误<reset/> (4.9.3.16)挂比流. 建议在通讯时通过OCSP应答接收到的nextUpdate字段的时间快到的时候查询OSCP应答者, 或者如果未设置nextUpdate字段, 则每隔24小时查询一次.

在流被关闭之后, 来自关闭的流的初始化实体需要重连，并且接收实体需要基于新流协商期间出示的证书来验证初始化实体.

在XMPP扩展中使用证书

证书可以被用于XMPP扩展，为了应用层加密或在XML流这一级进行验证(例如, 用于点对点加密). 这类扩展将定义他它们自己的证书处理规则. 最低限度, 这类扩展是被鼓励和本协议定义的规则保持一致性, 特别是只在必要的时候定义额外的规则.

强制实现的TLS和SASL技术

接下来的TLSm密码套件和SASL机制是强制实现的(自然的, 实现可以同时支持其他密码套件和机制). 对于和TLS密码套件相关的安全性事项, 参见13.9.4和TLS. 对于和SASL机制相关的安全性事项, 参见13.9.4, SASL, 和用于特定SASL机制的协议，类似 SCRAM, DIGEST-MD5, 和 PLAIN.

仅用于验证

对于仅用于验证, 服务器和客户端必须支持SASL的SCRAM机制 -- 特别是, SCRAM-SHA-1 和 SCRAM-SHA-1-PLUS 变种.

> 安全警告: 即使可能只是在不保密的情况下完成验证, 建议服务器和客户端在尝试以SASL验证之前用TLS来保护流, 同时帮助保护SASL协商期间交换的信息比并帮助防止特定的如13.9.4和13.9.5所述的降级攻击. 即使使用了TLS, 实现也应该如13.9.4所述强迫通道绑定.
>
> 互操作性备注: SCRAM机制的 SCRAM-SHA-1 或 SASL-SCRAM-SHA-1-PLUS 变种替代 SASL DIGEST-MD5 机制成为在只验证的场景中XMPP的强制实现的基于密码的方法. 为了保持对已部署的基础设施的向后兼容性, 鼓励实现继续支持DIGEST-MD5定义的DIGEST-MD5; 无论如何, 和DIGEST-MD5的互操作性问题广为人知并将长期存在.

仅用于保密

对于仅保密的情况, 服务器必须支持带TLS_RSA_WITH_AES_128_CBC_SHA密码套件的TLS.

> 安全警告: 因为一个仅保密的连接的安全性能弱于保密加验证, 建议服务器和客户端尽量使用双保险来连接(例如, 在尝试以SASL验证之前先用TLS保护流). 在实践中, 当对端服务器没有出示一个被信任的证书而是使用服务器回拨XEP-0220用于弱身份验证的时候,仅保密几乎不用于服务器-服务器连接, 但是仅保密的TLS仍能防止对连接的随意窃听.

保密加密码验证

对于保密加密码验证, 服务器和客户端必须实现带TLS_RSA_WITH_AES_128_CBC_SHA密码套件的TLS加上SASL SCRAM, 特别是 SCRAM-SHA-1 和 SCRAM-SHA-1-PLUS 变种(倾向于 SCRAM-SHA1-PLUS , 如13.9.4所述).

如下面的安全性警告里面的详细解释所说, 在特定情况下，当它不可能提供更多替代安全的时候,服务器可提供带TLS_RSA_WITH_AES_128_CBC_SHA套件的TLS加SASL PLAIN; 另外, 客户端应该实现TLS上的PLAIN，为了最大化和那些无法部署更多替代安全的服务器的互操作性.

> 安全警告: 实践中, 很多服务器提供, 很多客户端使用, TLS 加 SASL PLAIN. 强烈建议使用 SCRAM机制中的 SCRAM-SHA-1，特别是 SCRAM-SHA-1-PLUS 变种来取代 PLAIN 机制, 因为它们优越的安全性能(包括强迫通道绑定 SCRAM-SHA-1-PLUS 的能力，如13.9.4所述). 客户端应该把TLS加SASL PLAIN当成最低优先级的技术,只在和无法提供SCRAM(或其他提供超过TLS加SASL PLAIN的安全替代)的服务器交互时使用, 必须优先使用更安全的机制(例如, EXTERNAL, SCRAM-SHA-1-PLUS, SCRAM-SHA-1, 或老的 DIGEST-MD5 机制)来取代PLAIN机制, 并且如果该流没有如13.7.2.1所述通过使用完整证书验证的TLS的来获得最低限度的保密性和完整性的话，该客户端不能(MUST NOT)使用PLAIN机制.如果流的保密性和完整性没有

被TLS或同等安全层的保护, 服务器不能(MUST NOT)提供SASL PLAIN. 服务器应该不提供TLS加SASL PLAIN，除非它不能提供一些SASL SCRAM的变种(或其他比TLS加SASL PLAIN更安全的替代), 例如, 因为XMPP服务依赖于不受XMPP管理员控制的数据库或目录的验证, 类似插件式验证模块(PAM), 轻量级目录访问协议(LDAP)目录LDAP, 或验证, 授权, 和 核查 (AAA) 密钥管理协议 (关于指南, 参考 [RFC6120#参考文献|AAA]]). 无论如何, 如果服务器需要和已安装的不支持SCRAM或其他比TLS加SASL PLAIN更安全的替代的客户端交互，即使服务器支持更安全的替代的时候，它提供TLS加SASL PLAIN可能也是适当的.

保密加无密码验证

对于保密加无密码验证, 服务器必须而且客户端应该实现带TLS_RSA_WITH_AES_128_CBC_SHA密码套件的TLS加上带PKIX证书的SASL EXTERNAL机制(见SASL的的附录A).

技术重用

在SASL中使用Base 64

客户端和服务器都必须验证SASL协商期间收到的任何 base 64 数据. 实现必须拒绝(不是忽略)任何没有显式地被base 64字母所允许的字符串; 这帮助防止创建可能用于"泄露"信息的隐蔽通道.

如果('=') 字符被包含在其他字符串数据里但又不是最后一个字符的话(例如, "=AAA" 或 "BBBB=CCC")，实现不能(MUST NOT)在非法输入上中断而必须拒绝任何包含填充 ('=') 字符的base 64字符串序列; 这帮助防止缓冲区溢出攻击和该实现上的其他攻击.

因为base 64编码是在视觉上隐藏那些很容易识别的信息（如密码），所以它不提供任何计算的保密性.

对base 64编码的所有使用必须按照BASE64第四章的定义并且填充比特值必须设为零.

使用DNS

XMPP典型地依赖域名系统(特别是 DNS-SRV 记录) 在客户端连接到服务器或对端服务器连接到另一个服务器之前把完整合格域名解析成一个IP地址. 在尝试协商XML流之前, 除非初始化实体已经如第三章定义的那样解析了接收实体的DNS域名(尽管在每一次连接尝试之前去解析DNS域名不是必要的, 因为DNS解析结果可能被当成长时间存在的值暂时缓存了)，否则它不能(MUST NOT)进行下一步动作. 无论如何, 在缺乏安全DNS选项(例如, 由DNSSEC提供的)的情况下, 一个能访问DNS服务器数据的恶意攻击者, 或能导致一个欺骗性的应答被缓存在一个递归的解析者器中, 可能会导致初始化实体连接到任何由攻击者选择的XMPP服务器上. 部署和验证服务器证书有助于防止这类攻击.

使用哈希函数

XMPP本身不直接强制任何特定的加密哈希函数的使用. 然而, 在XMPP以来的技术上(例如, TLS和特定的SASL机制), 以及各个XMPP扩展, 可能强制加密哈希函数的使用. 建议那些实现XMPP技术的人或开发XMPP扩展的人密切监视对和XMPP相关的互联网协议的最先进的加密散列函数的攻击. 一些有用的指导, 参考HASHES.

使用SASL

因为初始化实体选择从接收实体出示的列表中选择一个可接受的SASL机制, 初始化实体依赖接收实体的列表用于验证. 如果攻击者能获得通道的控制权并因而出示一个弱的机制列表，那么这个依赖会导致某种降级攻击的可能. 为了减轻这种攻击, 在尝试SASL协商之前双方应该使用TLS保护通道并且要么如

13.7.2.1所述执行完整证书验证，要么使用一个提供通道绑定的SASL机制, 类似 SCRAM-SHA-1-PLUS. (使用完整证书验证的TLS保护通道有助于确保SASL协商期间的信息交换的保密性和完整性.)

SASL框架本身不提供办法把SASL验证绑定到一个在更底层协商中提供保密性和完整性保护的安全层(例如, TLS). 这类的绑定众所周知是一个 "通道绑定" (见 CHANNEL). 一些SASL机制提供了通道绑定, 在XMPP的场景中它典型地表现为绑定到 TLS (见 CHANNEL-TLS). 如果一个SASL机制提供了一个通道绑定(例如, SCRAM 就是这样), 那么使用那个机制的XMPP实体应该优先选择该通道绑定变种(例如, "SCRAM-SHA-1-PLUS" 优先于 "SCRAM-SHA-1"). 如果一个SASL机制不提供通道绑定, 那么该机制不能提供方法来验证正在使用SASL验证的源和目标端的低层安全性; 进一步, 如果端点不是同一个, 那么低层安全性不能被信任用来保护SASL验证的实体的数据传输. 在这种情况下, 一个SASL安全层应该有效地忽略低层安全性的联机状态来被协商.

很多部署的XMPP服务使用密码验证客户端连接. 众所周知大部分自然人用户选择相对弱的密码. 尽管服务的开通超出了本文的范围, 允许基于密码验证的XMPP服务器应该对密码强度强制最低标准来帮助阻止字典攻击. 因为所有基于密码验证的机制都易受到猜密码攻击, XMPP服务器必须在SASL验证时限制允许的重试次数, 如6.4.5所述.

一些SASL机制(例如, ANONYMOUS) 不对客户端到服务器提供强的对端实体验证. 建议在服务管理员允许这类机制的时候警告. 在XMPP中使用SASL ANONYMOUS机制的最佳实践参见XEP-0175.

使用TLS

TLS的实现典型地支持传输层安全性协议的多个版本以及旧的安全套接字层(SSL)协议. 因为已知的安全性漏洞, XMPP服务器和客户端不能(MUST NOT)请求, 提供, 或使用SSL 2.0. 更多细节, 参见跟随TLS-SSL2的TLS的附录E.2.

为阻止中间人攻击, TLS客户端(它可能是一个XMPP客户端或一个XMPP服务器) 必须验证TLS服务器的证书并且必须和对服务器的完整合格域名和TLS证书消息中出示的服务器身份， 如13.7.2.1所述(详见TLS-CERTS.

对TLS协商的支持严格来说是可选的. 然而, 支持TLS再协商的实现必须实现和使用TLS重协商扩展TLS-NEG. 更多细节在5.3.5中.

使用UTF-8

使用UTF-8使得传输非ASCII字符串成为可能, 并且从而允许字符"哄骗"情节, 这里显示的值和它本身的值不一样. 此外, 已知有一些和解码UTF-8相关的攻击情节. 对于这两中情况, 详见UTF-8.

使用XML

因为XMPP是一个可扩展标记语言XML的应用范例, 很多XML-MEDIA和XML-GUIDE里描述的安全性事项也适用于XMPP. XMPP从几个方面减轻上述的风险, 类似11.1下的禁令以及减少外部参考样式表或转换, 但是这些缓解因素是不全面的.

信息泄露

IP地址

客户端的IP地址和访问方法不能(MUST NOT)被服务器公开(例如, 典型地发生在IRC).

联机状态信息

XMPP的核心方面之一是联机状态信息: 关于一个XMPP实体的网络可用性信息(即, 该实体当前是在线还是离线). nce leak" occurs 当一个实体的网络可用性被疏忽和偶然地泄露给第二个未被授权了解第一个实体的网络可用性的实体的时候，就发生了"出席信息泄漏".

尽管联机状态信息更完整的讨论是在XMPP-IM, 注意XMPP服务器不能(MUST NOT)泄露联机状态信息是非常重要的. 特别是在核心XMPP级, 即时寻址和网络可用性是和特定的已连接资源相关的; 所以, 任何对已连接的资源的全JID的泄露都会构成一个联机状态泄露. 为了帮助阻止这类的联机状态泄露, 服务器不能(MUST NOT)对可能发生的潜在攻击者向某实体的纯JID(<localpart@domainpart>)或全JID(<localpart@domainpart/resourcepart>)发送XML节的时候返回不同的节错误.

目录搜集

如果服务器在应答一个不存在的用户帐号的节的时候生成了一个节错误, 使用 <service-unavailable/>节错误条件(8.3.3.19)能帮助防止目录搜集攻击, 因为它和其他集中错误返回同样的错误条件, 例如, 未知的IQ子元素的命名空间, 对于某个域不允许"离线消息存储" (XEP-0160) 或消息转发. 无论如何, 当没有使用更高级的屏蔽技术的时候，错误节的确切的XML的细微差别, 以及这类错误返回的时机, 都能让攻击者确定一个用户的出席信息(见XEP-0016 和 XEP-0191). 所以, 一个实行高警告级别的服务器可能对于收到的某些特定类型的节的所有应答中都不返回任何错误, 这样一个不存在的用户似乎表现得像一个没兴趣和发送者交谈的用户.

拒绝服务

DOS 定义的拒绝服务如下:

> 拒绝服务(DoS)攻击是一种这样的攻击， 使用一个或多个机器攻击一个受害者并试图阻止受害者做有用的工作. 受害者可能是一个网络服务器, 客户端 或路由器, 一个网络连接或整个网络, 一个单独的互联网用户或一个使用互联网的商业公司, 一个互联网服务提供商(ISP), 国家, 或这些变种的任意组合.

本文讨论的一些事项有助于防止拒绝服务攻击(例如, 要求服务器不能(MUST NOT)处理未提供适当的验证身份的客户端发来的XML节，也不能(MUST NOT)处理其身份未通过SASL验证或只被服务器回拨弱验证的对端服务器发来的XML节).

另外, XEP-0205 提供了一个防止潜在的对于XMPP系统的拒绝服务攻击的最佳实践的详细讨论. 建议包括:

1. 服务器实现应该允许服务器管理员限制它在任何时候从给定IP地址接受的TCP连接数量. 如果一个实体尝试连接但是已经达到了TCP连接数量的上限, 接收服务器不能(MUST NOT)允许继续新的连接.
2. 服务器实现应该允许服务器管理员限制它在给定时间间隔里从给定IP地址接收的TCP连接尝试的数量. 如果一个实体尝试连接但是已经达到了连接尝试次数的上限, 接收服务器不能(MUST NOT)允许继续新的连接.
3. 服务器实现应该允许服务器管理员限制它在任何时候允许一个帐号绑定的已连接资源的数量. 如果客户端尝试绑定一个资源但是已经达到了配置的允许资源数量, 接收服务器必须返回一个<resource-constraint/>节错误(8.3.3.18).
4. 服务器实现应该允许服务器管理员限制它从一个已连接的客户端或对端服务器接收到的节的大小(这里 "大小" 是包含定义于XML 2.4节的所有XML, 从该节打开的 "<" 字符到关闭的 ">" 字符). 一个已部署的服务器的最大节大小不能(MUST NOT)小于 10000 字节, 它体现了有利于发起方实体的表达和服务器处理节的开销之间的妥协. 服务器实现不应该盲目地为所有的部署设置 10000 字节的值，而应该允许服务器管理员设置他们自己的限制.如果一个已连接的资源或对端服务器发送一个违反了字节数上限的节, 接收服务器必须要么返回一个<policy-violation/>节错误(8.3.3.12), 这将允许发送者纠错, 或以一个<policy-violation/>流错误(4.9.3.14)关闭该流.

5. 服务器实现应该允许服务器管理员限制一个已连接的客户端在给定时间段发送给单个接收者的XML节的数量. 如果一个已连接客户端在给定时间段内发送太多节给单个的接收者, 接收服务器应该不(SHOULD NOT)该节而是并应该返回<policy-violation/>节错误(8.3.3.12).
6. 服务器实现应该允许服务器管理员限制它允许一个已连接客户端或对端服务器在给定的时间段内使用的带宽.
7. 服务器实现可以允许服务器管理员限制它允许一个已连接资源或对端服务器通过活动的连接发送的节的类型(基于扩展的内容 "载荷"). 这个限制和约束是部署策略的事情.
8. 服务器实现可以拒绝路由或递送任何它认为是滥用的节, 可以返回或不返回一个错误给发送者.

更多关于对XMPP系统拒绝服务攻击的详细建议, 参考XEP-0205.

防火墙

尽管 DNS SRV记录能指示连接的实体使用5222(客户端-服务器)和5269 (服务器-服务器)以外的TCP端口, XMPP通讯典型地发生在那些在IANA(见 第14章)注册了的端口. 使用那些广为人知的端口让管理员们很容易地通过现有的常见部署的防火墙来允许或禁止XMPP活动.

域间联盟

术语 "联盟" 常用于描述两个服务器域之间的通讯.

因为服务配置是一个策略的问题, 对于任何给定的服务器来说，支持联盟是可选的. 如果一个特定的服务器允许联盟, 它应该允许前面说过的强安全性来确保身份验证和保密性; 为支持达到这个目的，兼容的实现应该支持TLS和SASL.

在RFC 3920定义 TLS 加上带证书的 SASL EXTERNAL 来加密和验证服务器-服务器流之前, 只有一个对于对端服务器进行弱身份验证的方法，就是定义于XEP-0220的服务器回拨. 即使同时使用了DNSSEC, 服务器回拨也只提供了弱身份验证而未提供保密性或完整性. 在撰写本文的时候, 服务器回拨仍是被广泛使用的技术,用来对服务器-服务器流提供一定程度的保证. 这一现实导致了从TLS + SASL EXTERNAL到服务器回拨的降级攻击的可能性，如果攻击者能获得通道的控制权从而让初始化服务器以为接收服务器不支持TLS或没有适当的证书的话. 为阻止这一攻击, 即使当前证书是自签名或不可信的证书，双方也应该在进行下一步之前使用TLS保护通道.

不可抵赖

提供双方实体的身份验证和数据完整性的系统有潜在的能力让一个实体证明另一个实体打算发送特定的数据给第三方实体. 尽管XMPP系统能提供双方实体身份验证和数据完整性, XMPP从未被定义为提供不可抵赖性.

IANA事项

以下小节更新了RFC3920提供的注册项. 本章是根据IANA-GUIDE来解释的.

TLS数据的XML命名空间名

用于在可扩展的消息和出席信息协议(XMPP)中的STARTTLS协商数据的URN子命名空间定义如下. (该命名空间坚持XML-REG定义的格式.)

URI:

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

协议:

RFC 6120

描述:

这是定义于RFC 6120中的可扩展消息和出席信息协议(XMPP)中的STARTTLS协商数据的XML命名空间名.

注册联系人:

IESG <iesg@ietf.org>

SASL数据的XML命名空间名

用于在可扩展的消息和出席信息协议(XMPP)中的SASL协商数据的URN子命名空间定义如下. (该命名空间坚持XML-REG定义的格式.)

URI:

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

协议:

RFC 6120

描述:

这是定义于RFC 6120中的可扩展消息和出席信息协议(XMPP)中的SASL协商数据的XML命名空间名.

注册联系人:

IESG <iesg@ietf.org>

流错误的XML命名空间名

用于在可扩展的消息和出席信息协议(XMPP)中的流错误数据的URN子命名空间定义如下. (该命名空间坚持XML-REG定义的格式.)

URI:

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

协议:

RFC 6120

描述:

这是定义于RFC 6120中的可扩展消息和出席信息协议(XMPP)中的流错误数据的XML命名空间名.

注册联系人:

IESG <iesg@ietf.org>

资源绑定的XML命名空间名

用于在可扩展的消息和出席信息协议(XMPP)中的资源绑定的URN子命名空间定义如下. (该命名空间坚持XML-REG定义的格式.)

URI:

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

协议:

RFC 6120

描述:

这是定义于RFC 6120中的可扩展消息和出席信息协议(XMPP)中的资源绑定的XML命名空间名.

注册联系人:

IESG <iesg@ietf.org>

节错误的XML命名空间名

用于在可扩展的消息和出席信息协议(XMPP)中的节错误数据的URN子命名空间定义如下. (该命名空间坚持XML-REG定义的格式.)

URI:

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

协议:

RFC 6120

描述:

这是定义于RFC 6120中的可扩展消息和出席信息协议(XMPP)中的节错误数据的XML命名空间名.

注册联系人:

IESG <iesg@ietf.org>

GSSAPI服务名

IANA已经把"xmpp"注册为一个GSS-API服务名, 如6.6章所述.

端口号和服务名

IANA已经把"xmpp-client"和"xmpp-server"分别注册为[[RFC6120#规范引用|TCP]端口号5222和5269的关键字. 依据IANA-PORTS, 本文更新了现有的注册项如下.

服务名:

xmpp-client

传输协议:

TCP

描述:

一个为XMPP客户端应用的连接提供支持的服务

注册机构:

IETF XMPP Working Group

联系方式:

IESG <iesg@ietf.org>

参考:

RFC 6120

端口号:

5222

服务名:

xmpp-server

传输协议:

TCP

描述:

一个为XMPP服务器应用的连接提供支持的服务

注册机构:

IETF XMPP Working Group

联系方式:

IESG <iesg@ietf.org>

参考:

RFC 6120

端口号:

5269

一致性需求

本章描述了一个协议特性集合来概述本协议的一致性需求. 本特性集合适用于软件认证, 互操作性测试, 和实现报告. 对于每个特性, 本章提供了以下信息:

- 自然人可读的名称
- 描述信息
- 本文特定章节的参考，该章节规范化地定义了该特性
- 该特性是否适用于客户端角色，服务器角色，或同时适用于两者(这里 "N/A" 表示该特性不适用于该特定角色)
- 该特性是必须(MUST)还是应该(SHOULD)被实现, 这里大写的术语被视为KEYWORDS所描述的含义

这里定义的特性集合尝试坚持Larry Masinter于2005年在IETF的NEWTRK工作组中建议的概念和格式, 摘自INTEROP. 尽管该特性集合比REPORTS中说的更详细, 它为实现报告的生成提供了一个合适的基础，这些报告将被提交以支持本协议按照PROCESS从建议标准发展成草案标准.

```
特性:
    bind-gen
描述:
    根据需求生成一个随机资源.
章节:
    7.6
角色:
    客户端 N/A, 服务器 必须.
```

```
特性:
    bind-mtn
描述:
    把资源绑定作为强制协商的.
章节:
    7.3.1
角色:
    客户端 必须, 服务器 必须.
```

```
特性:
    bind-restart
描述:
    在资源绑定的协商之后不重启该流.
章节:
    7.3.2
角色:
    客户端 必须, 服务器 必须.
```

```
特性:
    bind-support
描述:
    支持客户端把资源绑定到一个已验证的流上.
章节:
    7
```

角色:
客户端 必须, 服务器 必须.

特性:
sasl-correlate
描述:
当使用SASL验证一个流对端的时候, 把认证标识符结果关联到它从对端接收到的SASL协商的流头中的'from'地址(如果有的
章节:
6.4.6
角色:
客户端 应该, 服务器 应该.

特性:
sasl-errors
描述:
支持协商过程中的SASL错误.
章节:
6.5
角色:
客户端 必须, 服务器 必须.

特性:
sasl-mtn
描述:
把SASL作为强制协商.
章节:
6.3.1
角色:
客户端 必须, 服务器 必须.

特性:
sasl-restart
描述:
在SASL协商之后发起或处理流重启.
章节:
6.3.2
角色:
客户端 必须, 服务器 必须.

特性:
sasl-support
描述:
支持简单验证和安全层用于验证.
章节:
6
角色:
客户端 必须, 服务器 必须.

特性:
security-mti-auth-scram
描述:
仅支持 SASL SCRAM 机制用于验证(这表示同时支持 SCRAM-SHA-1 和 SCRAM-SHA-1-PLUS 变种).
章节:
13.8

角色:
客户端 必须, 服务器 必须.

特性:
security-mti-both-external
描述:
支持 TLS 和 SASL EXTERNAL 用于保密和验证.
章节:
13.8
角色:
客户端 应该, 服务器 必须.

特性:
security-mti-both-plain
描述:
支持TLS使用TLS_RSA_WITH_AES_128_CBC_SHA密码组加上SASL PLAIN机制用于保密和验证.
章节:
13.8
角色:
客户端 应该, 服务器 可以.

特性:
security-mti-both-scram
描述:
支持TLS使用TLS_RSA_WITH_AES_128_CBC_SHA密码组加上SASL SCRAM机制的SCRAM-SHA-1和SCRAM-SHA-1-PLUS变种
章节:
13.8
角色:
客户端 必须, 服务器 必须.

特性:
security-mti-confidentiality
描述:
支持TLS使用TLS_RSA_WITH_AES_128_CBC_SHA密码组只用于保密.
章节:
13.8
角色:
客户端 N/A, 服务器 应该.

特性:
stanza-attribute-from
描述:
支持所有节种类共有的'from'属性.
章节:
8.1.2
角色:
客户端 必须, 服务器 必须.

特性:
stanza-attribute-from-stamp
描述:
标记或重写所有从已连接的客户端收到的节的'from'地址.
章节:
8.1.2.1

角色:
客户端 N/A, 服务器 必须.

特性:
stanza-attribute-from-validate
描述:
验证所有从对端服务器收到的节的'from'地址.
章节:
8.1.2.2
角色:
客户端 N/A, 服务器 必须.

特性:
stanza-attribute-id
描述:
支持所有节类型共有的'id'属性.
章节:
8.1.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-attribute-to
描述:
支持所有节类型共有的'to'属性.
章节:
8.1.1
角色:
客户端 必须, 服务器 必须.

特性:
stanza-attribute-to-validate
描述:
确保所有从对端服务器收到的节包含一个'to'地址.
章节:
8.1.1
角色:
客户端 N/A, 服务器 必须.

特性:
stanza-attribute-type
描述:
支持所有节类型共有的'type'属性.
章节:
8.1.4
角色:
客户端 必须, 服务器 必须.

特性:
stanza-attribute-xmllang
描述:
支持所有节类型共有的'xml:lang'属性.
章节:
8.1.5
角色:
客户端 必须, 服务器 必须.

特性:
stanza-error
描述:
为所有节种类生成和处理"error"类型的节.
章节:
8.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-error-child
描述:
确保"error"类型的节包含一个<error/>子元素.
章节:
8.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-error-id
描述:
确保"error"类型的节保留出发该错误的节提供的'id'.
章节:
8.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-error-reply
描述:
不对一个"error"类型的节应答另一个"error"类型的节.
章节:
8.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-extension
描述:
正确地处理由一个不被支持的XML命名空间限定的XML数据, 这里"正确地处理"意味着在一个消息或出席信息节的情况下忽略该
章节:
8.4
章节:
客户端 必须, 服务器 必须.

特性:
stanza-iq-child
描述:
确切地在一个类型为"get"或"set"的<iq/>节中包含一个子元素, 在一个类型为"result"的<iq/>节中包含零或一个子元素
章节:
8.2.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-iq-id
描述:
确保所有<iq/>节包含一个'id'属性.
章节:
8.2.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-iq-reply
描述:
以一个类型为"result"或 "error"的<iq/>节应答一个类型为"get"或"set"的<iq/>节.
章节:
8.2.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-iq-type
描述:
确保所有<iq/>节包含一个'type'属性, 它的值为"get", "set", "result", 或 "error".
章节:
8.2.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-kind-iq
描述:
支持<iq/>节.
章节:
8.2.3
角色:
客户端 必须, 服务器 必须.

特性:
stanza-kind-message
描述:
支持<message/>节.
章节:
8.2.1
角色:
客户端 必须, 服务器 必须.

特性:
stanza-kind-presence
描述:
支持<presence/>节.
章节:
8.2.2
角色:
客户端 必须, 服务器 必须.

特性:
stream-attribute-initial-from
描述:
在发起流头中包含一个'from'属性.
章节:
4.7.1
角色:
客户端 应该, 服务器 必须.

特性:
stream-attribute-initial-lang
描述:
在发起流头中包含一个'xml:lang'属性.
章节:
4.7.4
角色:
客户端 应该, 服务器 应该.

特性:
stream-attribute-initial-to
描述:
在发起流头中包含一个'to'属性.
章节:
4.7.2
角色:
客户端 必须, 服务器 必须.

特性:
stream-attribute-response-from
描述:
在应答流头中包含一个'from'属性.
章节:
4.7.1
惧色:
客户端 N/A, 服务器 必须.

特性:
stream-attribute-response-id
描述:
在应答流头中包含一个'id'属性.
章节:
4.7.3
角色:
客户端 N/A, 服务器 必须.

特性:
stream-attribute-response-id-unique
描述:
确保应答流头中的'id'属性在接收实体的上下文中是唯一的.
章节:
4.7.3
角色:
客户端 N/A, 服务器 必须.

特性:
stream-attribute-response-to
描述:
在应答流头中包含一个'to'属性.
章节:
4.7.2
角色:
客户端 N/A, 服务器 应该.

特性:
stream-error-generate
描述:
在探测到一个流相关的错误条件时生成一个流错误(跟随一个关闭流的标签并且中止该TCP连接).
章节:
4.9
角色:
客户端 必须, 服务器 必须.

特性:
stream-fqdn-resolution
描述:
在打开一个到接收尸体的TCP连接之前解析FQDNs.
章节:
3.2
角色:
客户端 必须, 服务器 必须.

特性:
stream-negotiation-complete
描述:
在接收实体发送空的或仅包含自愿协商特性的流特性声明之前，不认为流协商已经完成.
章节:
4.3.5
角色:
客户端 必须, 服务器 必须.

流特性:
stream-negotiation-features
描述:
在发送一个应答流头之后发送流特性.
章节:
4.3.2
角色:
客户端 N/A, 服务器 必须.

特性:
stream-negotiation-restart
描述:
当流特性协商需要流重启时，认为前一个流被替换了，并在这样一个流特性协商之后发送或接收一个新的发起流头.
章节:
4.3.3
角色:
客户端 必须, 服务器 必须.

特性：
stream-reconnect
描述：
如果TCP连接被意外中止，以指数退避算法重连.
章节：
3.3
角色：
客户端 必须，服务器 必须.

特性：
stream-tcp-binding
描述：
绑定一个XML流到一个TCP连接.
章节：
3
角色：
客户端 必须，服务器 必须.

特性：
tls-certs
描述：
在TLS协商期间出示的证书所指定的身份.
章节：
13.7.2
角色：
客户端 必须，服务器 必须.

特性：
tls-mtn
描述：
如果STARTTLS是被声明的唯一特性或如果STARTTLS特性声明包含了一个空的<required/>元素，认为TLS是强制协商的.
章节：
5.3.1
角色：
客户端 必须，服务器 必须.

特性：
tls-restart
描述：
在TLS协商之后发起或处理一个流重启.
章节：
5.3.2
角色：
客户端 必须，服务器 必须.

特性：
tls-support
描述：
支持传输层安全用于流加密.
章节：
5
角色：
客户端 必须，服务器 必须.

特性:
tls-correlate
描述:
当验证一个在TLS协商期间由流对端出示的证书时, 关联该已验证身份到它从该对端接收的流头的'from'地址(如果存在).
章节:
13.7.2
角色:
客户端 应该, 服务器 应该.

特性:
xml-namespace-content-client
描述:
支持'jabber:client'作为内容命名空间.
章节:
4.8.2
角色:
客户端 必须, 服务器 必须.

特性:
xml-namespace-content-server
描述:
支持'jabber:server'作为内容命名空间.
章节:
4.8.2
角色:
客户端 N/A, 服务器 必须.

特性:
xml-namespace-streams-declaration
描述:
确保有一个命名空间声明用于'http://etherx.jabber.org/streams'命名空间.
章节:
4.8.1
角色:
客户端 必须, 服务器 必须.

特性:
xml-namespace-streams-prefix
描述:
确保所有由'http://etherx.jabber.org/streams'命名空间限定的元素的前缀(如果有)被定义于该命名空间声明中.
章节:
4.8.1
角色:
客户端 必须, 服务器 必须.

特性:
xml-restriction-comment
描述:
不生成或接受XML注释.
章节:
11.1
角色:
客户端 必须, 服务器 必须.

```
特性：
    xml-restriction-dtd
描述：
    不生成或接受内部或外部的DTD子集.
章节：
    11.1
角色：
    客户端 必须, 服务器 必须.
```

```
特性：
    xml-restriction-pi
描述：
    不生成或接受XML处理指示.
章节：
    11.1
角色：
    客户端 必须, 服务器 必须.
```

```
特性：
    xml-restriction-ref
描述：
    不生成或接受预定义的实体之外的内部或外部的实体参考.
章节：
    11.1
角色：
    客户端 必须, 服务器 必须.
```

```
特性：
    xml-wellformed-xml
描述：
    不生成或接不良格式XML的数据.
章节：
    11.3
角色：
    客户端 必须, 服务器 必须.
```

```
特性：
    xml-wellformed-ns
描述：
    不生成或接不良格式命名空间的数据.
章节：
    11.3
角色：
    客户端 必须, 服务器 必须.
```

参考

规范引用

[BASE64] Josefsson, S., “The Base16, Base32, and Base64 Data Encodings,” RFC 4648, October 2006 (TXT).

[CHANNEL] Williams, N., “On the Use of Channel Bindings to Secure Channels,” RFC 5056, November 2007 (TXT).

[CHANNEL-TLS] Altman, J., Williams, N., and L. Zhu, “Channel Bindings for TLS,” RFC 5929, July 2010 (TXT).

[CHARSETS] Alvestrand, H., “IETF Policy on Character Sets and Languages,” BCP 18, RFC 2277, January 1998 (TXT, HTML, XML).

[DNS-CONCEPTS] Mockapetris, P., “Domain names - concepts and facilities,” STD 13, RFC 1034, November 1987 (TXT).

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

[IPv6-ADDR] Kawamura, S. and M. Kawashima, “A Recommendation for IPv6 Address Text Representation,” RFC 5952, August 2010 (TXT).

[KEYWORDS] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

[LANGMATCH] Phillips, A. and M. Davis, “Matching of Language Tags,” BCP 47, RFC 4647, September 2006 (TXT).

[LANGTAGS] Phillips, A. and M. Davis, “Tags for Identifying Languages,” BCP 47, RFC 5646, September 2009 (TXT).

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP,” RFC 2560, June 1999 (TXT).

[PKIX] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, May 2008 (TXT).

[PKIX-ALGO] Jonsson, J. and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,” RFC 3447, February 2003 (TXT).

[PKIX-SRV] Santesson, S., “Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name,” RFC 4985, August 2007 (TXT).

[PLAIN] Zeilenga, K., “The PLAIN Simple Authentication and Security Layer (SASL) Mechanism,” RFC 4616, August 2006 (TXT).

[RANDOM] Eastlake, D., Schiller, J., and S. Crocker, “Randomness Requirements for Security,” BCP 106, RFC 4086, June 2005 (TXT).

[SASL] Melnikov, A. and K. Zeilenga, “Simple Authentication and Security Layer (SASL),” RFC 4422, June 2006 (TXT).

[SCRAM] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, “Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms,” RFC 5802, July 2010 (TXT).

[STRONGSEC] Schiller, J., “Strong Security Requirements for Internet Engineering Task Force Standard Protocols,” BCP 61, RFC 3365, August 2002 (TXT).

[TCP] Postel, J., “Transmission Control Protocol,” STD 7, RFC 793, September 1981 (TXT).

[TLS] Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, August 2008 (TXT).

[TLS-CERTS] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)," RFC 6125, March 2011.

[TLS-NEG] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension," RFC 5746, February 2010 (TXT).

[TLS-SSL2] Turner, S. and T. Polk, "Prohibiting Secure Sockets Layer (SSL) Version 2.0," RFC 6176, March 2011.

[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.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.0," 2010.

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

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," STD 66, RFC 3986, January 2005 (TXT, HTML, XML).

[X509] International Telecommunications Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks," ITU-T Recommendation X.509, ISO Standard 9594-8, March 2000.

[XML] Maler, E., Yergeau, F., Sperberg-McQueen, C., Paoli, J., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," World Wide Web Consortium Recommendation REC-xml-20081126, November 2008 (HTML).

[XML-GUIDE] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols," BCP 70, RFC 3470, January 2003 (TXT, HTML, XML).

[XML-MEDIA] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types," RFC 3023, January 2001 (TXT).

[XML-NAMES] Thompson, H., Hollander, D., Layman, A., Bray, T., and R. Tobin, "Namespaces in XML 1.0 (Third Edition)," World Wide Web Consortium Recommendation REC-xml-names-20091208, December 2009 (HTML).

[XMPP-ADDR] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format," RFC 6122, March 2011.

[XMPP-IM] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence," RFC 6121, March 2011.

参考文献

[AAA] Housley, R. and B. Aboba, "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management," BCP 132, RFC 4962, July 2007 (TXT).

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," STD 68, RFC 5234, January 2008 (TXT).

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

[ANONYMOUS] Zeilenga, K., “Anonymous Simple Authentication and Security Layer (SASL) Mechanism,” RFC 4505, June 2006 (TXT).

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

[DIGEST-MD5] Leach, P. and C. Newman, “Using Digest Authentication as a SASL Mechanism,” RFC 2831, May 2000 (TXT).

[DNSSEC] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “DNS Security Introduction and Requirements,” RFC 4033, March 2005 (TXT).

[DNS-TXT] Rosenbaum, R., “Using the Domain Name System To Store Arbitrary String Attributes,” RFC 1464, May 1993 (TXT).

[DOS] Handley, M., Rescorla, E., and IAB, “Internet Denial-of-Service Considerations,” RFC 4732, December 2006 (TXT).

[E2E-REQS] Saint-Andre, P., “Requirements for End-to-End Encryption in the Extensible Messaging and Presence Protocol (XMPP),” Work in Progress, March 2010.

[EMAIL-ARCH] Crocker, D., “Internet Mail Architecture,” RFC 5598, July 2009 (TXT).

[ETHERNET] “Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications,” IEEE Standard 802.3, September 1998.

[GSS-API] Linn, J., “Generic Security Service Application Program Interface Version 2, Update 1,” RFC 2743, January 2000 (TXT).

[HASHES] Hoffman, P. and B. Schneier, “Attacks on Cryptographic Hashes in Internet Protocols,” RFC 4270, November 2005 (TXT).

[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 (TXT, PS, PDF, HTML, XML).

[IANA-GUIDE] Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” BCP 26, RFC 5226, May 2008 (TXT).

[IANA-PORTS] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, “Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Transport Protocol Port Number and Service Name Registry,” Work in Progress, February 2011.

[IMAP] Crispin, M., “INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1,” RFC 3501, March 2003 (TXT).

[IMP-REQS] Day, M., Aggarwal, S., and J. Vincent, “Instant Messaging / Presence Protocol Requirements,” RFC 2779, February 2000 (TXT).

[INTEROP] Masinter, L., “Formalizing IETF Interoperability Reporting,” Work in Progress, October 2005.

[IRC] Kalt, C., “Internet Relay Chat: Architecture,” RFC 2810, April 2000 (TXT).

[IRI] Duerst, M. and M. Suignard, “Internationalized Resource Identifiers (IRIs),” RFC 3987, January 2005 (TXT).

[LDAP] Zeilenga, K., “Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map,” RFC 4510, June 2006 (TXT).

[LINKLOCAL] Cheshire, S., Aboba, B., and E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses,” RFC 3927, May 2005 (TXT).

[MAILBOXES] Crocker, D., “MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS,” RFC 2142, May 1997 (TXT, HTML, XML).

[POP3] Myers, J. and M. Rose, “Post Office Protocol – Version 3,” STD 53, RFC 1939, May 1996 (TXT).

[PROCESS] Bradner, S., “The Internet Standards Process -- Revision 3,” BCP 9, RFC 2026, October 1996 (TXT).

[REPORTS] Dusseault, L. and R. Sparks, “Guidance on Interoperation and Implementation Reports for Advancement to Draft Standard,” BCP 9, RFC 5657, September 2009 (TXT).

[REST] Fielding, R., “Architectural Styles and the Design of Network-based Software Architectures,” 2000.

[RFC3920] Saint-Andre, P., Ed., “Extensible Messaging and Presence Protocol (XMPP): Core,” RFC 3920, October 2004 (TXT, HTML, XML).

[RFC3921] Saint-Andre, P., Ed., “Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence,” RFC 3921, October 2004 (TXT, HTML, XML).

[SASLPREP] Zeilenga, K., “SASLprep: Stringprep Profile for User Names and Passwords,” RFC 4013, February 2005 (TXT).

[SEC-TERMS] Shirey, R., “Internet Security Glossary, Version 2,” RFC 4949, August 2007 (TXT).

[SMTP] Klensin, J., “Simple Mail Transfer Protocol,” RFC 5321, October 2008 (TXT).

[SEC-GUIDE] Rescorla, E. and B. Korver, “Guidelines for Writing RFC Text on Security Considerations,” BCP 72, RFC 3552, July 2003 (TXT).

[TLS-EXT] Eastlake 3rd, D., “Transport Layer Security (TLS) Extensions: Extension Definitions,” RFC 6066, January 2011 (TXT).

[TLS-RESUME] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, “Transport Layer Security (TLS) Session Resumption without Server-Side State,” RFC 5077, January 2008 (TXT).

[URN-OID] Mealling, M., “A URN Namespace of Object Identifiers,” RFC 3061, February 2001 (TXT).

[USINGTLS] Newman, C., “Using TLS with IMAP, POP3 and ACAP,” RFC 2595, June 1999 (TXT).

[UUID] Leach, P., Mealling, M., and R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace,” RFC 4122, July 2005 (TXT, HTML, XML).

[XEP-0001] Saint-Andre, P., “XMPP Extension Protocols,” XSF XEP 0001, March 2010.

[XEP-0016] Millard, P. and P. Saint-Andre, “Privacy Lists,” XSF XEP 0016, February 2007.

[XEP-0045] Saint-Andre, P., “Multi-User Chat,” XSF XEP 0045, July 2007.

[XEP-0060] Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe," XSF XEP 0060, July 2010.

[XEP-0071] Saint-Andre, P., "XHTML-IM," XSF XEP 0071, September 2008.

[XEP-0077] Saint-Andre, P., "In-Band Registration," XSF XEP 0077, September 2009.

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

[XEP-0100] Saint-Andre, P. and D. Smith, "Gateway Interaction," XSF XEP 0100, October 2005.

[XEP-0114] Saint-Andre, P., "Jabber Component Protocol," XSF XEP 0114, March 2005.

[XEP-0124] Paterson, I., Smith, D., and P. Saint-Andre, "Bidirectional-streams Over Synchronous HTTP (BOSH)," XSF XEP 0124, July 2010.

[XEP-0138] Hildebrand, J. and P. Saint-Andre, "Stream Compression," XSF XEP 0138, May 2009.

[XEP-0156] Hildebrand, J. and P. Saint-Andre, "Discovering Alternative XMPP Connection Methods," XSF XEP 0156, June 2007.

[XEP-0160] Saint-Andre, P., "Best Practices for Handling Offline Messages," XSF XEP 0160, January 2006.

[XEP-0174] Saint-Andre, P., "Link-Local Messaging," XSF XEP 0174, November 2008.

[XEP-0175] Saint-Andre, P., "Best Practices for Use of SASL ANONYMOUS," XSF XEP 0175, September 2009.

[XEP-0178] Saint-Andre, P. and P. Millard, "Best Practices for Use of SASL EXTERNAL with Certificates," XSF XEP 0178, February 2007.

[XEP-0191] Saint-Andre, P., "Simple Communications Blocking," XSF XEP 0191, February 2007.

[XEP-0198] Karneges, J., Hildebrand, J., Saint-Andre, P., Forno, F., Cridland, D., and M. Wild, "Stream Management," XSF XEP 0198, February 2011.

[XEP-0199] Saint-Andre, P., "XMPP Ping," XSF XEP 0199, June 2009.

[XEP-0205] Saint-Andre, P., "Best Practices to Discourage Denial of Service Attacks," XSF XEP 0205, January 2009.

[XEP-0206] Paterson, I. and P. Saint-Andre, "XMPP Over BOSH," XSF XEP 0206, July 2010.

[XEP-0220] Miller, J., Saint-Andre, P., and P. Hancke, "Server Dialback," XSF XEP 0220, March 2010.

[XEP-0225] Saint-Andre, P., "Component Connections," XSF XEP 0225, October 2008.

[XEP-0233] Miller, M., Saint-Andre, P., and J. Hildebrand, "Domain-Based Service Names in XMPP SASL Negotiation," XSF XEP 0233, June 2010.

[XEP-0288] Hancke, P. and D. Cridland, "Bidirectional Server-to-Server Connections," XSF XEP 0288, October 2010.

[XML-FRAG] Grosso, P. and D. Veillard, "XML Fragment Interchange," World Wide Web Consortium CR CR-xml-fragment-20010212, February 2001 (HTML).

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

[XML-SCHEMA] Thompson, H., Maloney, M., Mendelsohn, N., and D. Beech, "XML Schema Part 1: Structures Second Edition," World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004 (HTML).

[XMPP-URI] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)," RFC 5122, February 2008 (TXT).

附录A. XML Schemas

以下schemas正式定义了本文用到的各命名空间, 符合XML-SCHEMA. 因为XML流和节的验证是可选的, 这些schemas不是规范性的并且只被提供为描述的目的.

A.1. Stream命名空间

```
<?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:import namespace='jabber:client'/>
  <xs:import namespace='jabber:server'/>
  <xs:import namespace='urn:ietf:params:xml:ns:xmpp-sasl'/>
  <xs:import namespace='urn:ietf:params:xml:ns:xmpp-streams'/>
  <xs:import namespace='urn:ietf:params:xml:ns:xmpp-tls'/>

  <xs:element name='stream'>
    <xs:complexType>
      <xs:sequence xmlns:client='jabber:client'
                   xmlns:server='jabber:server'>
        <xs:element ref='features'
                    minOccurs='0'
                    maxOccurs='1'/>
        <xs:any namespace='urn:ietf:params:xml:ns:xmpp-tls'
                minOccurs='0'
                maxOccurs='1'/>
        <xs:any namespace='urn:ietf:params:xml:ns:xmpp-sasl'
                minOccurs='0'
                maxOccurs='1'/>
        <xs:any namespace='##other'
                minOccurs='0'
                maxOccurs='unbounded'
                processContents='lax'/>
        <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: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:string' 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:anyAttribute namespace='##other' processContents='lax'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='features'>
    <xs:complexType>
      <xs:sequence>
        <xs:any namespace='##other'
                minOccurs='0'
                maxOccurs='unbounded'
                processContents='lax'/>
      </xs:sequence>
    </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:any     namespace='##other'
                    minOccurs='0'
                    maxOccurs='1'
                    processContents='lax'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

A.2. Stream Error命名空间

```
<?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='not-well-formed' type='empty'/>
  <xs:element name='policy-violation' type='empty'/>
  <xs:element name='remote-connection-failed' type='empty'/>
  <xs:element name='reset' 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: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='not-well-formed'/>
      <xs:element ref='policy-violation'/>
      <xs:element ref='remote-connection-failed'/>
      <xs:element ref='reset'/>
      <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: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>
```

A.3. STARTTLS命名空间

```
<?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:choice minOccurs='0' maxOccurs='1'>
        <xs:element name='required' type='empty'/>
      </xs:choice>
    </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>
```

A.4. SASL命名空间

```
<?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'
                    minOccurs='1'
                    maxOccurs='unbounded'
                    type='xs:NMTOKEN'/>
        <xs:any namespace='##other'
                minOccurs='0'
                maxOccurs='unbounded'
                processContents='lax'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

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

  <xs:element name='auth'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute name='mechanism'
                        type='xs:NMTOKEN'
                        use='required'/>
        </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='success' type='xs:string'/>

  <xs:element name='failure'>
    <xs:complexType>
      <xs:sequence>
        <xs:choice minOccurs='0'>
          <xs:element name='aborted' type='empty'/>
          <xs:element name='account-disabled' type='empty'/>
          <xs:element name='credentials-expired' type='empty'/>
          <xs:element name='encryption-required' 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='malformed-request' 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:element ref='text' minOccurs='0' maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <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>
```

A.5. Client命名空间

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:client'
    xmlns='jabber:client'
    elementFormDefault='qualified'>

  <xs:import
      namespace='urn:ietf:params:xml:ns:xmpp-stanzas'/>

  <xs:element name='message'>
     <xs:complexType>
        <xs:sequence>
          <xs:choice minOccurs='0' maxOccurs='unbounded'>
            <xs:element ref='subject'/>
            <xs:element ref='body'/>
            <xs:element ref='thread'/>
          </xs:choice>
          <xs:any      namespace='##other'
                       minOccurs='0'
                       maxOccurs='unbounded'
                       processContents='lax'/>
          <xs:element ref='error'
                       minOccurs='0'/>
        </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='type'
                       use='optional'
                       default='normal'>
```

```
        <xs:simpleType>
          <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='chat'/>
            <xs:enumeration value='error'/>
            <xs:enumeration value='groupchat'/>
            <xs:enumeration value='headline'/>
            <xs:enumeration value='normal'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute ref='xml:lang' use='optional'/>
   </xs:complexType>
</xs:element>

<xs:element name='body'>
  <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:element name='subject'>
  <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:element name='thread'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <xs:attribute name='parent'
                      type='xs:NMTOKEN'
                      use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='presence'>
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs='0' maxOccurs='unbounded'>
        <xs:element ref='show'/>
        <xs:element ref='status'/>
        <xs:element ref='priority'/>
      </xs:choice>
      <xs:any     namespace='##other'
                  minOccurs='0'
                  maxOccurs='unbounded'
                  processContents='lax'/>
      <xs:element ref='error'
                  minOccurs='0'/>
    </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='type' use='optional'>
      <xs:simpleType>
        <xs:restriction base='xs:NMTOKEN'>
          <xs:enumeration value='error'/>
```

```
          <xs:enumeration value='probe'/>
          <xs:enumeration value='subscribe'/>
          <xs:enumeration value='subscribed'/>
          <xs:enumeration value='unavailable'/>
          <xs:enumeration value='unsubscribe'/>
          <xs:enumeration value='unsubscribed'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
</xs:element>

<xs:element name='show'>
  <xs:simpleType>
    <xs:restriction base='xs:NMTOKEN'>
      <xs:enumeration value='away'/>
      <xs:enumeration value='chat'/>
      <xs:enumeration value='dnd'/>
      <xs:enumeration value='xa'/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name='status'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='string1024'>
        <xs:attribute ref='xml:lang' use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:simpleType name='string1024'>
  <xs:restriction base='xs:string'>
    <xs:minLength value='1'/>
    <xs:maxLength value='1024'/>
  </xs:restriction>
</xs:simpleType>

<xs:element name='priority' type='xs:byte'/>

<xs:element name='iq'>
  <xs:complexType>
    <xs:sequence>
      <xs:any     namespace='##other'
                  minOccurs='0'
                  maxOccurs='1'
                  processContents='lax'/>
      <xs:element ref='error'
                  minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='from'
                  type='xs:string'
                  use='optional'/>
    <xs:attribute name='id'
                  type='xs:NMTOKEN'
                  use='required'/>
    <xs:attribute name='to'
                  type='xs:string'
                  use='optional'/>
    <xs:attribute name='type' use='required'>
      <xs:simpleType>
        <xs:restriction base='xs:NMTOKEN'>
          <xs:enumeration value='error'/>
          <xs:enumeration value='get'/>
          <xs:enumeration value='result'/>
          <xs:enumeration value='set'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
```

```
  </xs:element>

  <xs:element name='error'>
    <xs:complexType>
      <xs:sequence xmlns:err='urn:ietf:params:xml:ns:xmpp-stanzas'>
        <xs:group ref='err:stanzaErrorGroup'/>
        <xs:element ref='err:text'
                    minOccurs='0'/>
      </xs:sequence>
      <xs:attribute name='by'
                    type='xs:string'
                    use='optional'/>
      <xs:attribute name='type' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='auth'/>
            <xs:enumeration value='cancel'/>
            <xs:enumeration value='continue'/>
            <xs:enumeration value='modify'/>
            <xs:enumeration value='wait'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

A.6. Server命名空间

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:server'
    xmlns='jabber:server'
    elementFormDefault='qualified'>

  <xs:import
      namespace='urn:ietf:params:xml:ns:xmpp-stanzas'/>

  <xs:element name='message'>
     <xs:complexType>
        <xs:sequence>
          <xs:choice minOccurs='0' maxOccurs='unbounded'>
            <xs:element ref='subject'/>
            <xs:element ref='body'/>
            <xs:element ref='thread'/>
          </xs:choice>
          <xs:any namespace='##other'
                  minOccurs='0'
                  maxOccurs='unbounded'
                  processContents='lax'/>
          <xs:element ref='error'
                      minOccurs='0'/>
        </xs:sequence>
        <xs:attribute name='from'
                      type='xs:string'
                      use='required'/>
        <xs:attribute name='id'
                      type='xs:NMTOKEN'
                      use='optional'/>
        <xs:attribute name='to'
                      type='xs:string'
                      use='required'/>
        <xs:attribute name='type'
                      use='optional'
                      default='normal'>
          <xs:simpleType>
```

```
          <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='chat'/>
            <xs:enumeration value='error'/>
            <xs:enumeration value='groupchat'/>
            <xs:enumeration value='headline'/>
            <xs:enumeration value='normal'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute ref='xml:lang' use='optional'/>
    </xs:complexType>
</xs:element>

<xs:element name='body'>
  <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:element name='subject'>
  <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:element name='thread'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <xs:attribute name='parent'
                      type='xs:NMTOKEN'
                      use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='subject'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <xs:attribute name='parent'
                      type='xs:NMTOKEN'
                      use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='presence'>
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs='0' maxOccurs='unbounded'>
        <xs:element ref='show'/>
        <xs:element ref='status'/>
        <xs:element ref='priority'/>
      </xs:choice>
      <xs:any     namespace='##other'
                  minOccurs='0'
                  maxOccurs='unbounded'
                  processContents='lax'/>
      <xs:element ref='error'
                  minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='from'
                  type='xs:string'
```

```
                  use='required'/>
    <xs:attribute name='id'
                  type='xs:NMTOKEN'
                  use='optional'/>
    <xs:attribute name='to'
                  type='xs:string'
                  use='required'/>
    <xs:attribute name='type' use='optional'>
      <xs:simpleType>
        <xs:restriction base='xs:NMTOKEN'>
          <xs:enumeration value='error'/>
          <xs:enumeration value='probe'/>
          <xs:enumeration value='subscribe'/>
          <xs:enumeration value='subscribed'/>
          <xs:enumeration value='unavailable'/>
          <xs:enumeration value='unsubscribe'/>
          <xs:enumeration value='unsubscribed'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
</xs:element>

<xs:element name='show'>
  <xs:simpleType>
    <xs:restriction base='xs:NMTOKEN'>
      <xs:enumeration value='away'/>
      <xs:enumeration value='chat'/>
      <xs:enumeration value='dnd'/>
      <xs:enumeration value='xa'/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name='status'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='string1024'>
        <xs:attribute ref='xml:lang' use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:simpleType name='string1024'>
  <xs:restriction base='xs:string'>
    <xs:minLength value='1'/>
    <xs:maxLength value='1024'/>
  </xs:restriction>
</xs:simpleType>

<xs:element name='priority' type='xs:byte' default='0'/>

<xs:element name='iq'>
  <xs:complexType>
    <xs:sequence>
      <xs:any namespace='##other'
              minOccurs='0'
              maxOccurs='1'
              processContents='lax'/>
      <xs:element ref='error'
                  minOccurs='0'/>
    </xs:sequence>
    <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='required'>
```

```
        <xs:simpleType>
          <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='error'/>
            <xs:enumeration value='get'/>
            <xs:enumeration value='result'/>
            <xs:enumeration value='set'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute ref='xml:lang' use='optional'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='error'>
    <xs:complexType>
      <xs:sequence xmlns:err='urn:ietf:params:xml:ns:xmpp-stanzas'>
        <xs:group ref='err:stanzaErrorGroup'/>
        <xs:element ref='err:text'
                    minOccurs='0'/>
      </xs:sequence>
      <xs:attribute name='by'
                    type='xs:string'
                    use='optional'/>
      <xs:attribute name='type' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NMTOKEN'>
            <xs:enumeration value='auth'/>
            <xs:enumeration value='cancel'/>
            <xs:enumeration value='continue'/>
            <xs:enumeration value='modify'/>
            <xs:enumeration value='wait'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

A.7. Resource Binding命名空间

```
<?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>
        <xs:element name='resource' type='resourceType'/>
        <xs:element name='jid' type='fullJIDType'/>
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name='fullJIDType'>
    <xs:restriction base='xs:string'>
      <xs:minLength value='8'/>
      <xs:maxLength value='3071'/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name='resourceType'>
    <xs:restriction base='xs:string'>
      <xs:minLength value='1'/>
      <xs:maxLength value='1023'/>
```

```
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

A.8. Stanza Error命名空间

```
<?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='not-authorized' type='empty'/>
  <xs:element name='policy-violation' 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-authorized'/>
      <xs:element ref='not-allowed'/>
      <xs:element ref='policy-violation'/>
      <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B. 联系地址

和MAILBOXES一致, 提供XMPP服务的组织被鼓励提供一个"XMPP"的互联网邮箱用于查询那个服务相关的事情, 这里最终的mailto URI的host部分是组织的域, 而不是XMPP服务本身的域(例如, XMPP服务可以在im.example.com提供但是互联网邮箱是<xmpp@example.com>).

附录C. 帐户设置

帐户设置超出了本文的范围. 对于帐户设置可能的方法包括由一个服务器管理员创建帐户和使用XEP-0077所述的'jabber:iq:register'命名空间进行带内帐户注册. 一个XMPP服务器实现或管理功能必须确保在帐户设置期间被分配的任何JID(包括 本地部分, 域部分, 资源部分, 和分隔符)符合XMPP-ADDR定义的XMPP地址的规范格式.

附录D. 和RFC 3920的不同

基于从实现和部署经验衍生的共识以及正式的互操作性测试, 以下是对 RFC 3920的实质性修改(除了许多编辑性质的修改).

- 把XMPP地址格式的规范移到单独的文档中去了.
- 在流头上建议或要求使用'from'和'to'属性.
- 更完整地定义了流关闭握手.
- 定义了建议的流重连算法.
- 把<xml-not-well-formed/>流错误条件的名字改成<not-well-formed/>以符合XML规范.
- 移除不必要和不用的<invalid-id/>流错误(见RFC 3920的历史文档).
- 定义了收到被禁止的XML特性之后返回的<restricted-xml/>流错误应答.
- 更全面的定义了<see-other-host/>流错误的格式和处理, 包括和RFC 3986以及RFC 5952关于IPv6地址的一致性(例如, 以方括号'[' 和 ']'关闭IPv6地址).
- 定义了对于客户端-服务器流SASL SCRAM机制是一个强制协商的技术.
- 定义了对于客户端-服务器流TLS加上SASL PLAIN机制是一个强制实现的技术.
- 定义了服务器必须支持SASL EXTERNAL机制, 但是客户端只是推荐支持它(因为最终用户X.509证书很难获得并且没有被广泛地部署).
- 移除对于服务器-服务器流困难的 两个连接 规则.
- 更清晰地定义了公共密钥证书和发行者证书的证书范例.
- 增加了<reset/>流错误(4.9.3.16)条件来处理过期/撤销的证书或一个现有的流上的附加安全关键特性.
- 增加了<account-disabled/>, <credentials-expired/>, <encryption-required/>, 和 <malformed-request/> SASL错误条件来处理被RFC 3920误漏或在RFC 4422中讨论过

但是不在RFC 2222中的的错误流.
- 移除不用的<payment-required/>节错误.
- 移除对于映射到特定的预定义实体的转义字符的不必要需求, 因为它们在XML中不需要被转义.
- 澄清DNS SRV查询和回退的过程.
- 澄清SASL安全层的处理.
- 澄清一个SASL简单用户名是本地部分, 而不是纯JID.
- 澄清流协商过程和相关的流程图.
- 澄清流特性的处理.
- 增加了'by'属性到用于节错误的<error/>元素，这样检测到错误的实体可以把它的JID包含在里面用于诊断或跟踪的目的.
- 澄清违反XML1.0和XML namespaces定义的良好格式的数据的处理.
- 定义了更多安全事项的细节, 特别是关于联机状态泄露和拒绝服务攻击.
- 从本文中移除服务器回拨协议到由XMPP标准基金会维护的一个独立的协议.

附录E. 致谢

本文衍生自自RFC 3920，是对它的更新. 没有贡献者们和评论者们的工作，本文是不可能完成的.

从RFC 3920发布以来，数百人提供了实现反馈, 错误报告, 澄清的请求, 和改善建议. 尽管本文的编辑已经努力找出所有这类反馈, 但他独自对任何其余的错误和含糊的地方负责.

特别感谢 Kevin Smith, Matthew Wild, Dave Cridland, Philipp Hancke, Waqas Hussain, Florian Zeitz, Ben Campbell, Jehan Pages, Paul Aurich, Justin Karneges, Kurt Zeilenga, Simon Josefsson, Ralph Meijer, Curtis King, 和其他在工作组最后召集期间做评论的人.

也分别感谢代表安全局进行复审的 Yaron Sheffer 和 Elwyn Davies和通用领域复审小组.

工作组主席是 Ben Campbell 和 Joe Hildebrand. 责任区域编辑是 Gonzalo Camarillo.

作者的地址

```
       Peter Saint-Andre
       Cisco
       1899 Wyknoop Street, Suite 600
       Denver, CO 80202
       USA
Phone:+1-303-308-3282
EMail:psaintan@cisco.com
```