XMPP 3920 最靠谱的中文翻译文档（一）

# xmpp 协议之 可扩展消息出席协议：核心 RFC 3920

摘要：

此文档定义了可扩展消息出席协议（XMPP）的核心特性：协议使用 XML 元素在任意两个网络端点间近实时的交换结构化信息。当 XMPP 为交换 XML 数据提供一般化，可扩展的框架时，它主要用于建立满足 RFC2779 的即时消息与出席应用的需求。

1 介绍

1.1 概要

XMPP 是一个开放的可扩展标记语言[XML]协议，用于近实时的消息、出席与请求-响应服务。基本语法语义最初是由 Jabber 开源社区在 1999 年开 发的。2002 年，XMPP 工作组授权开发一个 Jabber 协议的改写本，将适用于 IETF 的即时消息（IM）与出席技术。

作为 XMPP 工作组的成果，此文档定义了 XMPP 1.0 的核心内容；提供即时消息与出席功能的扩展需求定义在 RFC2779[IM-REQS]中，由 XMPP：即时消息与出席[XMPP-IM]指定。

1.2 术语

文档中的大写关键字："MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", "OPTIONAL"在 BCP14, 在 RFC 2119 [TERMS]中描述。

2 一般架构

2.1 概述

虽然 XMPP 并未与任何特定网络架构结合，但到目前为止，它大致上已经由一个客户-服务器的架构实现了。其中，客户端利用 XMPP 访问基于[TCP]连接的一个服务器，并且，服务器间也通过 TCP 连接进行彼此间的通信。

XMPP

Client----------Server----------Server

TCP TCP

下图为此架构的高层视图（“-”表示使用 XMPP 通信，“=”表示使用任何其它协议通信）

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

符号表示如下：

1） C1，C2，C3 = XMPP 客户端

2） S1，S2 = XMPP 服务器

3） G1 = 网关：在 XMPP 与外部协议（非 XMPP）的消息网络间转换。

4） FN1 = 外部消息网络

5） C1 = 外部消息网络的客户端

2.2 服务器

服务器作为 XMPP 通信担当智能抽象层。它的主要责任是：

1） 管理连接其它实体的会话，以 XML 流格式（第 4 节）在已授权的客户端、服务器以及其它实体间来回传送。

2） 通过 XML 流在实体间路由具有合适地址的 XML 节（第 9 节）。

大多数与 XMPP 兼容的服务器设想有能力存储客户端的数据（例：基于 XMPP 即时消息与出席应用的用户的联系列表）；在这种情况下，XML 数据由服务器自身代表客户端直接处理，并不路由到其它实体。

2.3 客户端

大多数客户端通过[TCP]连接直接连到服务器，并且使用 XMPP，充分利用由服务器及任何相关服务所提供的功能。多种资源（例如：设备或位置）可能代表 每个被授权客户端同时连到服务器上。每个资源均由定义在地址方案（第 3 节）下的 XMPP 地址的资源标识符来区别（例如:<[url=mailto:node@domain/home]node@domain/home[/url]> vs. <[url=mailto:node@domain/work]node@domain/work[/url]>）。客户端与服务器的推荐连接端口为 5222，已由 IANA 注册（参考端口编号（15.9 节））。

2.4 网关

网关是服务器端的一种特殊服务，它的主要功能是将 XMPP 翻译成外部消息系统所使用的协议（非 XMPP），也可将数据翻译回 XMPP。例如 EMAIL 网关 （参考[SMTP]），

Internet Relay Chat（参考[IRC]），SIMPLE（参考[SIIMPLE]，Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions），短消息服务（SMS），遗留即时消息服务，诸如 AIM，ICQ，MSN Messenger，Yahoo! Instant Messenger。网关与服务器间的通信，网关与外部消息系统间的通信，均未在此文档中定义。

2.5 网络

由于每个服务器由网络地址指定，并且由于服务器与服务器间的通信是客户与服务器协议的直接扩展，实际上，系统由互相通信的服务器网络组成。举个例 子，<[url=mailto:juliet@example.com]juliet@example.com[/url]>能与<[url=mailto:romeo@example.net]romeo@example.net[/url]>交换消息、出席，以及其它 信息。这是使用网络寻址标准的消息协议（例如[SMTP]）所熟悉的模式。任意两服务器间的通信是可选的。如果可通信，此类通信就应当发生在绑定到 [TCP]连接的 XML 流上。服务器间连接的推荐端口为 5269，由 IANA 注册（参考端口编号（15.9 节））

3 寻址方案

3.1 概述

实体可被看作是使用 XMPP 进行通信的任意网络端点（例如：一个网络上的 ID）。任意此类实体均以与 RFC2396[URI]一致的格式来唯一设定地址。 由于历史原因，XMPP 实体的地址称作 Jabber 标识符或 JID。一个有效 JID 包含一套有序元素：域标识符，结点标识符，资源标识符。

JID 的语法定义如下，使用增广巴斯科范式[ABNF]（Augmented Backus-Naur Form）。（Ipv4 地址与 Ipv6 地址规则定义在[Ipv6]的附录 B；符合结点规则的允许字符序列由 Nodeprep profile of [STRINGPREP]定义，编入本文档的附录 A；符合资源规则的允许字符序列由 Resourceprep profile of [STRINGPREP]定义，编入本文档的附录 B；子域规则参考国际化域标识的概念，在[IDNA]中有述）。

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

所有 JID 均基于前述规则。此结构最普通的用法就是用户以<[url=mailto:user@host/resource]user@host /resource[/url]>形式标识一个即时消息

用户、用户连接的服务器、用户连接的资源（例如：特别的客户端）。

然而，结点类型可能不仅是客户端，举个例子，一个提供多用户聊天服务的特别聊天室，可以以<[url=mailto:room@service]room@service[/url]>（“room”是聊天室名，“service”是多用 户聊天服务的主机名）作为地址。并且，此聊天室的特别拥有者可能以<[url=mailto:room@service/nick]room@service/nick[/url]>（“nick”是此拥有者的房间 昵称）作地址，许多其它 JID 类型均有可能（例如：<domain/resource>可能是一个服务器端脚本或服务）。

JID（结点标识符，域标识符，资源标识符）的每个可允许部分长度不准超过 1023 字节，结果，最大总长度（包括[url=mailto:‘@’]‘@’[/url]，‘/’分隔符）为 3071 字节。

3.2 域标识符

域标识符是基本标识符，且是 JID 中仅有的一个必须的元素（仅有域标识符的 JID 是有效的）。它通常表示网络网关与“主要的”服务器，具有为其它实体间的 连接进行 XML 路由与数据管理的能力。然而，由域标识符作为参考的实体并不总是服务器，它可能是一项以服务器子域为地址的服务，提供多于服务器（例：多用 户聊天服务，用户目录，或外部消息系统的一个网关）的功能。

每个服务器或服务的域标识符将通过网络进行通信，它可能是 IP 地址，并应当是完全合法的域名（参考[DNS]）。域标识符必须是一个“国际化的域名”，定 义在[IDNA]，Nameprep [NAMEPREP] profile of stringprep [STRINGPREP]可以无错应用。比较两个域标识符之前，服务器必须（客户端是应该）首先对标签（定义在[IDNA]）应用 Nameprep profile，以补足每个标识符。

3.3 节点标识符

结点标识符是一个可选的辅助标识符，放在域标识符之前，后以[url=mailto:‘@’]‘@’[/url]字符分隔。它通常表示实体请求与使用由服 务器或网关（例如：一个客户端）提供的网络访问，虽然它也能表示其它种类的实体（例如：有多用户聊天服务功能的聊天室）。由结点标识符表示的实体，在特定 域上下文中，在 XMPP 即时消息与出席应用中被加以地址，此类地址称作“bare JID”，形式为<[url=mailto:node@domain]node@domain[/url]>

结点标识符必须像 the Nodeprep profile of [STRINGPREP]这样格式化，可以无错应用。比较两个结点标识符之前，服务器必须（客户端应该）首先对每个标识符应

用 Nameprep profile。

3.4 资源标识符

资源标识符是一个可选的第三位标识符，位于域标识符之后，后跟‘/’作为分隔符。资源标识符可以修改<[url=mailto:node@domain]node@domain[/url]>也可以只是<domain>地址。它通常表示 一个特别的会话、连接（例如：一个设备或位置），或属于带有节点标识符的对象（例如：在多用户聊天室的一个参与者）。当提供必要的信息来完成资源绑定（第 7 节）时，资源标识符对服务器与其它客户端均不透明，并且由客户端实现来定义，以后，它作为一个“已连接资源”参考。实体可能同时维护多连接，每个已连接 的资源均由资源标识符来进行区别。

资源标识符必须按 Resourceprep profile of [STRINGPREP]格式化，才能无错应用。比较两个资源标识符前，服务器必须（客户端应该）首先为每个标识符应用 Resourceprep profile。

3.5 决定地址

SASL 协商后（第 6 节），如果正确，资源绑定（第 7 节），流接收实体必须决定初始实体的 JID。

如果 SASL 协商（第 6 节）期间未指定授权身份，对服务器与服务器间的通信，初始实体的 JID 应当被授权身份，派生于认证身份，在 SASL（Simple Authentication and Security Layer 简单授权与安全层）说明[SASL]中定义。

如果 SASL 协商(第 6 节)期间未指定授权身份,对客户端到服务器的通信，“bare JID”（<[url=mailto:node@domain]node@domain[/url]>）应该被授权身份，被派生于授权认证， 定义在[SASL]。在资源绑定期间（第 7 节）“full JID”（<[url=mailto:node@domain/resource]node@domain/resource[/url]& gt;）的资源标识符部分应当是客户端与服务器间协商的资源标识符。

接收实体必须确保结果 JID（包括结点标识符，域标识符，资源标识符，分隔符）遵从此节中前面所定义的规则与格式；为满足此限制，接收实体可能需要替代由接收实体所决定的规范的 JID 初始实体所发送的 JID。

XMPP 3920 最靠谱的中文翻译文档（二）

XML 流

4.1 概述

使 presence-aware 实体间能够相互迅速的、异步交换相关的小负载的结构化信息有两种基本元素：XML 流与 XML 节。术语定义如下：

XML 流定义：XML 流是一个容器，用于网络上任意两实体间交换 XML 元素。XML 流的开始是以一个起始的 XML<stream>标记（有合 适的属性与命名空间声明）表示，XML 流的结尾以一个结束的 XML</stream>标记表示。在流的生命周期中，初始化它的实体能够通过流 发送极多的 XML 元素，元素与 XML 节（定义在此，<message/>, <presence/>, 或 <iq/>元素由缺省命名空间验证）都用于协商流（例：协商使用 TLS（第 5 节）或使用 SASL（第 6 节））。“初始流”是从初始实体（通常 是一个客户端或服务器）到接收实体（通常是一个服务器）的协商，并被看作与从初始实体到接收实体的会话一致。初始流能从初始实体到接收实体单向通信；为了 能够从接收实体到初始实体的信息交换，接收实体必须在反方向协商一个流（“响应流”）。

XML 节定义：XML 节是一个不连续的结构化信息语义单元，通过 XML 流从一个实体发送到另一个实体。XML 节以根</stream>的直接子层存在，如果它匹配产品[43]内容[XML]，则可以很好的平衡。

任何 XML 节的开始都由深度为 1 的 XML 流（例如：<presence>）的开始标记元素来清楚的表示，XML 节的结尾由相应的深度为 1 的关 闭标记来清楚的表示。为传送想要的信息，一个 XML 节可能包含必要的子元素（带有属性，元素，XML 字符数据）。在此定义的仅有的 XML 节 是<message/>，<presence/>，<iq/>元素，由流的缺省命名空间验证，在 XML 节（第 9 节） 中描述；为传输层安全（TLS：Transport Layer Security）协商，SASL 协商，或服务器回叫（第 8 节）而发送的 XML 元素，并不会当作 XML 节来考虑。

考虑一个客户端与服务器的会话例子。为了连接到服务器，客户端必须初始化一个 XML 流：发送一个起始的<stream>标记给服务，可选先于 一个指定 XML 版本的文本声明与字符编码支持（参考文本声明的内容（11.4）；也可参考字符编码（11.5））。服从本地策略与所提供的服务，服务器接 下来应该回复另一个 XML 流给客户端，再次可选先于一个文本声明。一但客户端完成了 SASL 协商（第 6 节），客户端可以通过流发送极多的 XML 节给网络上 的任意容器。当客户端想关闭流时，它简单发送一个关闭</stream>标记给服务器（也可以由

服务器来关闭流)，从这以后，客户端与服务器 都应终止潜在的连接(通常是一个 TCP 连接)。

习惯于将 XML 考虑成以文档为中心的人可能希望看到客户端与服务器的会话作为两个末端开口的（自由回答的）XML 文档的组成部分：一个从客户端到服务器， 另一个从服务器到客户端。从这个观点看，根<stream/>元素可被认为是每个“文档”的文档实体，两个“文档”都由通过两个 XML 流发送 的 XML 节的集聚来建立。然而，这种观点仅是一种方便；XMPP 并不以文档处理，而是以 XML 流或 XML 节来处理。

本质上，那么，一个 XML 流充当了所有通过会话发送的 XML 节的信封。可用图简单表示如下：

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

4.2 绑定到 TCP

虽然将一个 XML 流结合到一个[TCP]连接上不是必须的（例如：两个实体能通过其它诸如[HTTP]投票选举机制而彼此互连），此说明也只定义了 XMPP 到 TCP 的绑定。在客户端到服务器端通信的上下文中，服务器必须允许客户端为了从客户端到服务器与服务器到客户端的 XML 节发送共享的一个单 TCP 连接。在服务器到服务器的通信上下文中，服务器必须使用一条 TCP 连接用于从服务器到其对等服务器的 XML 节传送，另一条 TCP 连接（由对等初始 化）用于对其等服务器到服务器的 XML 节传送，总共有两条 TCP 连接。

4.3 流安全

当在 XMPP1.0 中协商 XML 流时，TLS 应当按 TLS 应用（第 5 节）所定义的来使用，SASL 必须按 SASL（第 6 节）所定义的来使用。“初始流”（例如：从初始实体到接收实体的流）与“响应流”（例如：从接收实体到初始实体的流）必须被分别保护，即使双向安全可能已通过相互的认证机制所建立。实体 不应当在流被认证之前，尝试通过流发送 XML 节（第 9 节），但如果这样做了，那么，其它实体不准接受此类节，并应当返回一个<non- authorized/>流错误，然后终止两端的 XML 流与潜在的 TCP 连接；注意，这只适用于 XML 节（例如：<message />, <presence/>, <iq/>元素，由缺省命名空间检查）并不适用于流协商（例如：用于协商使用 TLS（第 5 节）或使用 SASL（第 6 节））的 XML 元素。

4.4 流属性

流元素属性如下：

1） to—‘ to’属性应当仅用于从初始实体到接收实体的 XML 流头中，并且必须被设成一个接收实体服务的主机名。‘to’属性不应当设在接收实体回应初始实体的 XML 流头中；然而，如果‘to’属性包括在内，它应当被初始实体默默忽略。

2） from—‘ from’属性应当仅用于从接收实体到初始实体的 XML 流头中，并且必须被设成一个接收实体服务的主机名，此接收实体正授权访问初始实体。‘from’属 性不应在初始实体发送到接收实体的流头中；然而，如果‘from’属性包括在内，它应当被接收实体忽略。

3） id—‘ id’属性应当仅用于从接收实体到初始实体的 XML 流头中。此属性是唯一一个由接收实体创建的，作为初始实体流与接收实体间会话的密钥，并且，在接收应用 （通常是一个服务器）中是唯一的。注意：流 ID 可能是严格安全的，并且因此必须是即不能预测也不能重复的（参考[RANDOM]推荐关于随机安全观点）。 ‘id’属性不应在初始实体到接收实

体的 XML 流头中；然而，如果‘id’属性包含在内，应被接收实体忽略。

4） xml:lang—‘ xml:lang’属性（定义在[XML]的 12.2）应当包含在初始实体的初始流头中，用于指定缺省语言，此语言可以是任何通过流发送的人类可读的 XML 字符数据。如果属性包含在内，接收实体应当记住此值并做为初始流与响应流的缺省值；如果此属性不包含在内，接收实体应当为两个流使用一个可配置的缺 省值，它必须为响应流在头中通信。对所有通过初始化流发送的节，如果初始实体不包含‘xml:lang’属性，接收实体应当应用缺省值；如果初始实体包含 ‘xml:lang’属性，接收实体不准修改或删除它（参考 xml:lang（9.1.5））。‘xml:lang’属性值必须是一个 NMTOKEN（定 义在[XML]（2.3）），并且必须与定义在 RFC3006[LANGTAGS]中的格式一致。

5） version—版本属性出现设到至少是“1.0”信号值，支持定义在说明书中的相关流协议（包括流特征）。有关代与属性处理的具体规则定义如下：

可总结如下：

| | initiating to receiving | receiving to initiating |
|---|---|---|
| to | hostname of receiver | silently ignored |
| from | silently ignored | hostname of receiver |
| id | silently ignored | session key |
| xml:lang | default language | default language |
| version | signals XMPP 1.0 support | signals XMPP 1.0 support |

4.4.1 版本支持

XMPP 版本在此指定为“1.0”，特别的，这封装了流相关协议（TLS 应用（5），SASL 应用（6），流错误（4.7）），还有三个已定义的 XML 节 类型（<message/>, <presence/>, and <iq/>）的语义。XMPP 版本的编号方案是“<major>.<minor>”。Major 与 minor 数字必 须作为分离的整数对待，并且每个数字可能并不按单数字增加。因此"XMPP 2.4"是一个比"XMPP 2.13"低的版本，依次低于"XMPP 12.3"。前导零（例如："XMPP 6.01"）必须被接收者忽略并不准发送。

Major 版本号应当增加，只要流与节格式或是所需行为已很大程度上改变，以至于老版本如果对它不理解的并采取在旧版说明中指定的动作时，只简单忽略元素 与属性时无法与新版本实体互操作，就要增加主版本号。次版本号指新能力，并且必须被有一个更小次版本

号的实体所忽略，但被有更大次版本号的实体作信息目的 用。举例：次版本号可能指处理消息，出席，或 IQ 节新近定义的‘type’属性值；有更大次版本号的实体将简单注意它的通信者不理解此‘type’属性 值，并因此而不发送它。

以下规则由实现应用于产生与处理在流头中的‘版本’属性：

1） 初始实体必须在初始流头中将版本属性值设到它所支持的最高版本号（例如：如果它所支持的最高版本号定义在此说明中，必须设值为“1.0”）

2） 接收实体必须在响应流头中设置版本属性值或者是初始实体提供的值，或者是接收实体所支持的最高版本号，无论哪一个更低。接收实体必须对主、次版本号做数字比较，而不是"<major>.<minor>"字符串匹配。

3） 如果包含在响应流头中的版本号至少一个主版本号低于包含在初始流头中的版本号，并且新版本实体不能像上述那样与旧版本互操作，初始实体应当产生一个<unsupported-version/>流错误，并终止 XML 流与潜在的 TCP 连接。

4） 如果每个实体都收到一个带有“无版本号”属性的流头，实体必须考虑由其它实体支持版本将是“0.0”并不应当在发送响应流时包括‘version’属性。

### 4.5 命名空间声明

流元素必须拥有流命名空间声明和一个缺省的命名空间声明（命名空间声明定义在 XML 命名空间说明文档[XML-NAMES]中）。对有关流命名空间与缺省命名空间的更细节的信息，看命名空间名称与前缀（11.2）。

### 4.6 流特征

如果初始化实体包含版本属性，并在初始流头中，其值至少设为“1.0”，那么接收实体必须发送一个<features/>子元素（由流命名空 间前缀作前缀）给初始实体，以宣布任何可被协商的（或另外需要被广告的能力）流级别的特征。当前，这仅用于广告在此定义的 TLS 应用（5），SASL 应用 （6）和资源绑定（7），并且，会话按照[XMPP-IM]中所定义的来建立；然而，流特征的功能性可被用于广告其它将来可协商的特征。如果实体不理解或 不支持某些特征，那么它应当默默的忽略。如果一个或多个安全特征（例如：TLS 与 SASL）需要在非安全特征（例如：资源绑定）被提供之前成功被协商，非 安全相关特征不应当在相关安全特征被协商之前包含在流特征中被广告。

### 4.7 流错误

根流元素可能包含一个<error/>子元素，此元素由流命名空间前缀来加前缀。如果错误子元素感觉到一个流级别错误发生，它必须由一个兼容实体（通常是一个服务器而不是一个客户端）来发送。

#### 4.7.1 规则

以下规则应用于流级别错误：

1） 设想所有流级别错误均是不可恢复的；因此，如果一个错误在流级别层发生，那么检测错误的实体必须发送一个流错误给其它实体，发送一个关闭</stream>标记，并终止潜在的 TCP 连接。

2）如果在流被建立期间发生错误，接收实体必须一直发送起始<stream>标记，将<error/>元素作为流元素的子元素，发送 关闭</stream>标记，并终止潜在的 TCP 连接。此种情况下，如果初始实体在‘to’属性（或根本没提供‘to’属性）中提供了一个未 知主机，服务器应当在流头的‘from’属性中提供服务器的授权主机名，并在终止前发送。

#### 4.7.2 语法

流错误语法如下：

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

<error/>元素：

1） 必须包含一个子元素，此子元素与以下定义的已定义的节错误条件之一相一致；此元素必须被'urn:ietf:params:xml:ns:xmpp-streams'命名空间认为是合格的。

2） 可能包含一个<text/>子元素，此子元素包含了更详细描述错误的 XML 字符数据；

此元素必须被 'urn:ietf:params:xml:ns:xmpp-streams'命名空间认为是合格的，并且，应当拥有一个'xml:lang'属性来指明 XML 字符数据的自然语言。

3） 可能包含一个用于说明特殊应用错误条件的子元素；此元素必须由一个已定义应用命名空间来认证，并且，它的结构由那个命名空间来定义。

<text/>元素是可选的。如果包含了此元素，它应当仅用于提供描述性或诊断性的信息，来补充一个已定义的条件或特殊应用条件的意思；它不 应当由一个应用以程序化的形式叙述。它不应当作为错误消息展示给一个用户，但可能另外显示与包含条件元素（或元素们）相关的错误消息。

4.7.3 已定义条件

以下定义了流级别错误条件：

1）<bad-format/>--已经发送 XML 的实体不能被处理；此错误可能用于代替更特殊的 XML 相关错误，例如：<bad-namespace-prefix/>, <invalid-xml/>, <restricted-xml/>, <unsupported-encoding/>, <xml-not-well-formed/>，虽然更特殊的错误是首选。

2）<bad-namespace-prefix/>--实体已经发送了一个不被支持的名空间前缀，或在一个需要那样一个前缀的元素中发送了没有命名空间的前缀（参考 XML 命名空间名与前缀（11.2））。

3）<conflict/>--服务器正为实体关闭活动流，因为一个已经被初始化的新流与现存流冲突。

4）<connection-timeout/>--一段时间内（可根据本地服务策略配置）实体并不通过流产生任何通信。

5）<host-gone/>--由初始实体在流头中提供的‘to’属性值对应于一个主机名，而此主机名已不再被一个服务器当作主机了。

6）<host-unknown/>--由初始实体在流头中提供的‘to’属性值于服务器所拥有的主机名不一致。

7）<improper-addressing/>--一个在两个服务器间发送的节，缺少‘to’或‘from’属性（或此属性无值）

8）<internal-server-error/>--服务器经历了错误配置或其它未定义内部错误使其无法提供服务。

9）<invalid-from/>--在‘from’地址中提供的 JID 或主机名与已授权的 JID 或有效域协商不匹配，此有效域协商为通过 SASL 或回叫服务器间的协商，或通过授权与资源绑定的客户端与服务器间的协商。

10）<invalid-id/>--流 ID 或回叫 ID 是无效的或与以前提供的 ID 不匹配。

11）<invalid-namespace/>--流命名空间名不只是 http://etherx.jabber.org/streams，或回叫命名空间名不只是"jabber:server:dialback"（参考 XML 命名空间名与前缀（11.2））

12）<invalid-xml/>--实体通过流向执行验证的服务器发送了无效的 XML（参考验证（11.3））。

13）<not-authorized/>--实体试图在流被认证前发送数据，或不授权执行一个相关流协商的活动；接收实体在发送流错误前不准处理违规节。

14）<policy-violation/>--实体违反了某些本地策略；服务器可能选择在<text/>元素或特殊-应用条件元素中指定策略。

15）<remote-connection-failed/>服务器不能适当的连接到远程实体，需要认证或授权。

16）<resource-constraint/>服务器缺少提供服务给流的必要的系统资源。

17）<restricted-xml/>实体试图发送受限的 XML 特征，例如评注、处理介绍，DTD，实体参考，或保留字符（参考（11.1））。

18）<see-other-host/>服务器将不提供服务给初始实体，但正重定向传输给另一个主机；服务器应当指定替换的主机名或 IP 地址（必须是有效域标识符），作为<see-other-host/>元素的 XML 字符数据。

19）<system-shutdown/>服务器被关闭，并且所有的活动流被关闭。

20）<undefined-condition/> 错误条件是由此列表中的其它已定义条件中的一个；此错误条件应当仅用在与特殊-应用条件相结合。

21）<unsupported-encoding/>初始实体已在不被服务器支持的编码中为流编码（11.5）

22）<unsupported-stanza-type/>初始实体已发送了一个不被服务器支持的第一级子流。

23）<unsupported-version/>由初始实体在流头提供的版本属性值指定了一个不被服务器支持的 XMPP 版本；服务器可能在<text/>元素中指定它支持的版本。

24）<xml-not-well-formed/>初始实体已经发送了不标准的 XML，标准的 XML 由[XML]定义。

#### 4.7.4 特殊应用条件

注意，一个应用可能通过在错误元素中包含一个合适的命名空间子元素来提供特殊应用

流错误信息。特殊应用元素应当补充或进一步验证一个已定义元素。因此，<error/>元素将包含两到三个子元素：

```
<stream:error>
 <xml-not-well-formed
    xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
 <text xml:lang='en' xmlns='urn:ietf:params:xml:ns:xmpp-streams'>
  Some special application diagnostic information!
 </text>
 <escape-your-data xmlns='application-ns'/>
</stream:error>
</stream:stream>
```

XMPP 3920 最靠谱的中文翻译文档（三）

4.8 简化的流例子

此部分包含两个简化的客户端与服务器（“C”行是从客户端发送到服务器，而“S”行是由服务器发送到客户端）间基于流会话的例子；这些例子解释进一步的概念。

```
A basic "session":

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

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

A "session" gone bad:

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

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

## 5 使用 TLS

### 5.1 概述

XMPP 包含一个方法，用于保护流不被篡改和偷听。此信道加密方法利用传输层安全（TLS）协议[TLS]，连同“STARTTLS”扩展，在为描述在 RFC 2595[USINGTLS]中的 IMAP[IMAP]，POP3[POP3]，ACAP[ACAP]等相似协议扩展模型。用于 STARTTLS 扩展的命 名空间名是'urn:ietf:params:xml:ns:xmpp-tls'。

一个给定域的管理者可能需要使用 TLS 来进行客户端到服务器的通信，服务器到服务器的通信，或二者兼有。客户端应使用 TLS 去保护流，在企图完成 SASL 协商之前，而且，服务器出于保护服务器到服务器的通信的考虑，应在两个域间使用 TLS。

应用以下规则：

1） 遵从此说明的初始实体必须包含版本属性，并在初始流头中将其值设为“1.0”。

2） 如果两服务器间的 TLS 协商发生，直到服务器宣称的域名系统（DNS）主机名被决定（参考服务器到服务器的通信（14.4））后，才能处理通信。

3）当与此说明一致的接收实体收到一个包含版本属性设为至少“1.0”的初

始化流时，发送一个流头作响应（包含版本标记）后，必须包含一 个<starttls/>元素（由'urn:ietf:params:xml:ns:xmpp-tls'命名空间认证)并带有它所支持的其它 流特征的列表。[服务器以<starttls/>响应]

4） 如果初始实体选择使用 TLS，TLS 协商必须在 SASL 协商处理之前完成；这种协商顺序是必要的，用于帮助保护 SASL 协商期间发送认证信息，并在 TLS 协商之前这段时间，使基于使用认证的 SASL EXTERNAL 机制成为可能。

5） 在 TLS 协商期间，实体不准在根流元素中发送任何空白字符（匹配[XML]内容，产品[3]）作为元素间（任何在 TLS 例子中的空白字符都只是为了便于阅读）的分隔符；这种限制有助于确保合适的安全层字节精度。

6）接收实体必须考虑 TLS 协商在发送<proceed/>元素的关闭“>”字符之后立即开始。初始实体必须考虑 TLS 协商在收到来自于接收实体的<proceed/>元素的关闭“>”字符之后立即开始。

7） 初始实体必须验证由接收实体表示的证书；参考证书验证（14.2）相关证书验证步骤。

8） 证书必须根据初始实体（例如：一个用户）提供的主机名来检查，而不是通过域名系统解析的主机名；例如：如果用户指定一个"example.com"的主机 名，而 DNS SRV[SRV]查找并返回"im.example.com"，证书必须作为"example.com"被检查。如果对任何此种 XMPP 实体（例如，客户 端或服务器）的一个 JID 在一个证书中被表示，它必须作为一个 UTF8String 来表示,UTF8String 在位于 subjiectAltName 中 的一个 otherName 实体中，使用[ASN.1]对象标识符"id-on-xmppAddr"，在本文档 5.1.1 中说明。

9） 如果 TLS 协商成功，接收实体必须抛弃 TLS 生效之前，来自初始实体的任何非安全格式的知识。

10） 如果 TLS 协商成功，初始实体必须抛弃 TLS 生效之前，来自接收实体的任何非安全格式知识。

11） 如果 TLS 协商成功，接收实体不准提供 STARTTLS 扩展给当流重新开如时被提供的带有其他流特征的初始实体。

12） 如果 TLS 协商成功，初始实体必须继续 SASL 协商。

13） 如果 TLS 协商结果失败，接收实体必须终止 XML 流与潜在的 TCP 连接。

14） 参考强制实施技术（14.7）相关的必须被支持的机制。

5.1.1 ASN.1 用于 XMPP 地址的对象标识符

上述[ASN.1]对象标识符"id-on-xmppAddr"定义如下：

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

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

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

XmppAddr ::= UTF8String
```

对象标识符可能也以点分制显示，格式为"1.3.6.1.5.5.7.8.5"

5. 2 叙述

当初始实体使用 TLS 保护一个带有接收实体的流时，步骤如下：

1） 初始实体打开一个 TCP 连接，靠发送开放 XML 流头给接收实体，此流头包含版本属性，并设其值至少为“1.0”，来初始化流。

2）接收实体以打开一个 TCP 连接并发送一个 XML 流头给初始实体作为响应，此流头包含值至少为“1.0”版本属性。

3） 接收实体靠包含带有其它支持流特征（如果 TLS 需要与接收实体交互，它应当靠包含一个<required/>元素作为<starttls/>的子元素来标记此事实)的列表来为初始实体提供 STARTTLS 扩展。

4） 初始实体发起 STARTTLS 命令（例：由 'urn:ietf:params:xml:ns:xmpp-tls' 命名空间确认的<starttls/>元素）去指导希望开始 TLS 协商去保护流的接收实体。

5）接收实体必须以由命名空间'urn:ietf:params:xml:ns:xmpp-tls'认证了的<proceed/>元素 或<failure/>元素响应。如果有失败情况发生，接收实体必须终止双方的 XML 流与潜在的 TCP 连接。如果接着向下进行，实体必须尝试 通过 TCP 连接完成 TLS 协商，并不准发送任何进一步的 XML 数据，直到 TLS 协商完成。

6） 初始实体与接收实体尝试依据[TLS]完成 TLS 协商。

7） 如果 TLS 协商不成功，接收实体必须终止 TCP 连接。如果 TLS 协商成功，

初始实体必须靠发送一个开始 XML 流头给接收实体(它并不需要先发送一个关 闭</stream>标记，因为接收实体与初始实体必须考虑到原始流根据成功的 TLS 协商而被关闭)，以初始一个新流。

8） 根据从初始实体接收的新流头，接收实体必须靠发送一个新 XML 流头给有可利用特征（不包括 STARTTLS 特征）的初始实体来响应。

5．3 客户端到服务器的例子

下面例子显示了一个客户端保护使用 STARTTLS(注：替换步骤显示在下一行，用来解释协议失败的情况；他们在本例中并不详尽也不是必须的由数据发送而触发)流的数据流。

1 步：客户端初始流给服务器：

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

步 2：服务器以发送给客户端一个流标记作为响应：

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        id='c2s_123'
        from='example.com'
        version='1.0'>
```

步 3：服务器发送 STARTTLS 扩展给客户端，并带有认证机制与任何其它流特征：

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

<mechanism>DIGEST-MD5</mechanism>

<mechanism>PLAIN</mechanism>

</mechanisms>

</stream:features>

步 4：客户端发送 STARTTLS 命令给服务器：

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

步 5：服务器通知客户端它被允许处理

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

步 5（替代）：服务器通知客户端 TLS 协商失败，并关闭流与 TCP 连接：

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

</stream:stream>

步 6：客户端与服务器试图协商通过现存的 TCP 连接 完成 TLS 协商。

步 7：如果 TLS 协商成功，客户端初始化一个新流给服务器：

<stream:stream

xmlns='jabber:client'

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

to='example.com'

version='1.0'>

步 7（代替 ）：如果 TLS 协商不成功，服务器关闭 TCP 连接。

步 8：服务器靠发送带有任何可利用流特征的流头给客户端作为响应。

<stream:stream

xmlns='jabber:client'

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

from='example.com'

id='c2s_234'

version='1.0'>

<stream:features>

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

<mechanism>DIGEST-MD5</mechanism>

<mechanism>PLAIN</mechanism>

<mechanism>EXTERNAL</mechanism>

</mechanisms>

</stream:features>

步 9：客户端继续 SASL 协商（6）

XMPP 3920 最靠谱的中文翻译文档（四）

6. 使用 SASL

6. 1 概述

XMPP 包含一个认证流的方法，此方法依靠一个简单认证与安全层（SASL）协议[SASL]的 XMPP-specific profile。SASL 提供一个一般化方法，用于给基于连接的协议加认证支持，并且，XMPP 使用一个一般化 XML 命名空间 profile，用于 SASL，遵从[SASL]的 profiling 需求。

以下规则应用：

1） 如果两个服务器间发生 SASL 协商，直到由服务器宣称的域名系统（DNS）主机名被解析了（参考服务器到服务器通信（14.4）），通信才可处理。

2） 如果实始实体能够进行 SASL 协商，，它必须在初始流头中包含值至少为“1.0”的版本属性。

3） 如果接收实体能够进行 SASL 协商，它必须在一个<mechanisms/>元素中广告一个或多个认证机制，此元素靠 ’urn:ietf:params:xml:ns:xmpp-sasl’命名空间响应从初始实体（如果开放流标记包含所设值至少为“1.0”的版本属性） 接收的开放流标记认证。

4） 在 SASL 协商期间，实体不准在根流元素中发送任何空白字符（匹配[XML]内容，产品[3]）作为元素间（任何在 SASL 例子中的空白字符都只是为了便于阅读）的分隔符；这种限制有助于确保合适的安全层字节精度。

5） 任何包含在 XML 元素中的 XML 字符数据，在 SASL 协商期间使用，必须使用 base64 编码，编码在 RFC3548 第三节有定义。

6） 如果所提供的一个“简单用户名”能够被选定 SASL 机制（例：由 DIGEST-MD5 与 CRAM-MD5 机制所支持，但不靠 EXTERNAL 与 GSSAPI 机制所支持）所支持，在认证期间，初始实体应当作为简单用户名提供它的发送域（IP 地址或包含在域标识符中的

全认证域名）在服务器对服务器的 通信情况下，或是它的已注册帐户名（包含在 XMPP 结点标识符中的用户或结点名）在客户到服务器的通信情况下。

7）如果初始实体希望代表其它实体与支持授权身份传输的被选 SASL 机制来行动，初始实体在 SASL 协商期间必须提供一个授权身份。如果初始实体不希望代表另 一个实体行动，它不准提供一个授权身份。正如[SASL]中指定的，初始实体不准提供一个授权身份，除非一个授权身份不同于缺省授权身份，此缺省授权身份 派生于描述在[SASL]中的认证身份。如果提供了，授权身份值对服务器来说必须是<domain>值形式（例：只有一个域标识符），对客户 端来说，必须是<node@domain>值形式（例：结点标识符与域标识符）。

8） 靠涉及到安全层协商的 SASL 协商的成功，接收实体必须抛弃来自本身没有获得 SASL 协商的初始实体的任何知识。

9） 靠涉及到安全层协商的 SASL 协商的成功，初始实体必须抛弃来自本身没有获得 SASL 协商的接收实体的任何知识。

10） 参考必须被支持的相关机制的强制实施技术（14.7）。

6．2 叙述

当初始实体使用 SASL 认证接收实体时，步骤如下：

1） 初始实体请求 SASL 认证，通过在开放 XML 流头中包含版本属性，并将其发送给接收实体，属性值设为“1.0”。

2） 发送一个 XML 流头作为回应后，接收实体广告一个可利用的 SASL 认证机制列表;列表中每一项都是一个<mechanism/>元素,作 为<mechanism/>容器元素的子元素，由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间认证，在流 命名空间中，依次是<features/>元素的子元素。如果使用 TLS(5)需要在一个特别认证机制可能使用之间建立，接收实体不准提供在 TLS 协商之前的可利用 SASL 认证机制列表中的机制。如果初始实体在 TLS 协商之前出示了有效证书，接收实体应当在 SASL 协商（参考[SASL]）之 间，提供 SASL EXTERNAL 机制给初始实体，虽然 EXTERNAL 机制可能在其它环境下被提供了。

3） 初始实体选择一个机制，靠发送一个已被'urn:ietf:params:xml:ns:xmpp-sasl'命名空间认定为合格 的<auth/>元素给接收实体,并为‘mechanism’属性包含一个合适的值。如果此机制需要 XML 字符数据，此元素可能包含 XML 字 符数据（在 SASL 术语中，“初始响应”）；如果初始实体需要发送一个 0 长度的初始响应，它必须按一个单等号符号（“=”）传输此响应，意味着响应出现， 但不包含数据。

4） 如果需要，接收实体靠发送一个<chanllenge/>元素来挑战实始实体，此元素由给初始实体的 'urn:ietf:params:xml:ns:xmpp-sasl'命名空间来限定；此元素可能包含 XML 字符数据（必须根据由初始实体选择的 SASL 机制的定义一致的来计算）。

5） 初始实体响应此挑战，靠发送由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<response/> 元素给接收实体；此元素可能包含 XML 字符数据（必须根据由初始实体选择的 SASL 机制的定义一致的来计算）。

6） 如果需要，接收实体发送更多的挑战，初始实体发送更多的响应。

Challenge/response 序列对继续，直到以下三种事情之一发生：

1）初始实体终止握手,靠发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定的<abort/>元素 给接收实体。根据接收一个<abort/>元素，接收实体应当允许一个可配置的但合理的重试号（至少 2），之后，必须终止 TCP 连接；这使初 始实体（例：一个终端用户客户端）能够忍受已提供的不正确的信任（例：一个错误类型的 password）而不用被迫重连。

2） 接收实体报告握手失败，靠发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定 的<failure/>元素给初始实体（失败的特殊原因应当以<failure/>元素的子元素进行通 信,<failure/>元素定义在 SASL 错误中（6.4））。如果错误情况发生,接收实体应当允许一个可配置的,但合理的重试号（至少 2），之后，它必须终止 TCP 连接；这使初始实体（例：一个终端用户客户端）能够忍受已提供的不正确的信任（例：一个错误类型的 password）而不用 被迫重连。

3） 接收实体报告握手成功，靠发送一个由'urn:ietf:params:xml:ns:xmpp-sasl'命名空间限定 的<success/>元素给初始实体；此元素可能包含 XML 字符数据（在 SASL 术语中，“additional data with success”），如果需要靠选定的 SASL 机制。根据接收的<success/>元素，初始实体必须靠发送一个开放的 XML 流头去初始化 一个新流给接收实体（它不必事先发送一个关闭</stream>标记，因为接收实体与初始实体必须考虑源流根据发送或接 收<success/>元素而将被关闭）。根据从初始实体接收的新流头,接收实体必须发送一个新 XML 流头给初始实体作为响应,并带有任何可 利用的特征（但并不包含 STARTTLS 与 SASL 特征）或一个空<features/>元素（重要表示没有其它特征可利用）；任何那种其它 在此未定义的特征必须由 XMPP 的相关扩展来定义。

6．3 SASL 定义

[SASL]的 profiling 需求要求协议定义提供以下信息：

服务名：“xmpp”

初始序列：初始实体提供一个开放 XML 流头后，并且接收实体按此响应后，接收实体提供一个可接收的认证方法列表。初始实体从列表中选择一个方法并作为‘machanism’属性值发送给接收实体，此属性被<auth/>元素拥有，随意的包括一个初始响应以避免环路。

交换序列：挑战与响应通过由接收实体到初始实体<challenge/>元素的交换与由初始实体到接收实体的<response />元素的交换而执行。接收实体靠发送一个<failure/>元素报告错误，发送一个<success/>元素报告成 功；初始实体靠发送<abort/>元素终止交换。根据成功协商，两端都认为源 XML 流将被关并且新流头由两端实体发送。

安全层协商：安全层在为接收实体发送<success/>元素的关闭“>”字符后立即有效，安全层在为初始实体发 送<success/>元素的关闭“>”字符后立即有效。层顺序为：首先是[TCP]，然后是[TLS]，然后是[SASL]，然后是 XMPP。

使用授权身份：授权身份可以被 XMPP 用于指示客户端非缺省<node@domain>或服务器发送<domain>。

6．4 SASL 错误

以下是 SASL 相关错误条件的定义：（略）

1）<aborted/>--

2）<incorrect-encoding/>--

3）<invalid-authzid/>--

4）<invalid-mechanism/>--

5）<mechanism-too-weak/>--

6）<not-authorized/>--

7）<temporary-auth-failure/>--

6．5 客户端到服务器的例子

以下例子显示了使用 SASL 授权的客户端与服务器端的数据流，正常情况下，是在 TLS 协商（注：显示在下面的替换步骤用于显示错误情况的协议；他们并不详尽也不是

必要的由本例中数据发送而触发。）成功之后。

步 1：客户端初始流给服务器：

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

步 2：服务器使用一个流标记作为响应发送给客户端：

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

步 3：服务器通知客户端可利用的认证机制：

```
<stream:features>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
        <mechanism>DIGEST-MD5</mechanism>
        <mechanism>PLAIN</mechanism>
    </mechanisms>
</stream:features>
```

步 4：客户端选择一个认证机制：

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

步 5：服务器发送一个[BASE64]编码挑战给客户端：

```
<challenge
xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>     cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9I
k9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNz
Cg==
```

</challenge>

解码挑战是：

realm="somerealm",nonce="OA6MG9tEQGm2hh",\

qop="auth",charset=utf-8,algorithm=md5-sess

步 5（替换）：服务器返回错误给客户端：

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

<incorrect-encoding/>

</failure>

</stream:stream>

步 6：客户端发送一个[BASE64]编码响应挑战：

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

dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0i

T0E2TUc5dEVRR2OyaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5jPTAw

MDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20i LHJlc3BvbnNlPW

QzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNoYXJzZXQ9dXRmLTgK

</response>

步 7：服务器发送另一个[BASE64]编码挑战给客户端：

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

cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3YjgOZGJhYmNkZmZmZAo=

</challenge>

解码挑战是：

rspauth=ea40f60335c427b5527b84dbabcdfffd

步 7（替换）：服务器返回错误给客户端：

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

<temporary-auth-failure/>

</failure>

</stream:stream>

步 8：客户端响应挑战：

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

步 9：服务器通知客户端认证成功：

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

步 9（替换）：服务器通知客户端认证失败：

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

步 10：客户端初始化一个新流给服务器：

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

步 11：服务器通过发送流头来响应客户端，伴随有任意另外的特征（或空特征元素）：

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        id='c2s_345'
        from='example.com'
        version='1.0'>
<stream:features>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
    <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
</stream:features>
```

6．6 服务器到服务器的例子

以下例子显示服务器与服务器使用 SASL 认证的数据流，正常情况下，是在 TLS 协商之后（注：以下可替换步骤是由失败情况提供的；他们不是详尽的也不是必要的由数据发送而触发）。

步 1：Server1 初始化流给 Server2：

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

步 2：Server2 发送一个流标记响应 Server1：

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

步 3：Server2 通知 Server1 可利用的认证机制：

```
<stream:features>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
        <mechanism>DIGEST-MD5</mechanism>
        <mechanism>KERBEROS_V4</mechanism>
    </mechanisms>
</stream:features>
```

步 4：Server1 选择一个认证机制：

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

步 5：Server2 发送一个[BASE64]编码挑战给 Server1：

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9
ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNz
</challenge>
```

编码挑战是：

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

步 5（替换）：Server2 返回错误给 Server1

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

步 6：Server1 发送[BASE64]编码响应挑战：

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
dXNlcm5hbWU9ImV4YW1wbGUub3JnIixyZWFsbT0ic29tZXJlYWxtIixub25j
ZT0iT0E2TUc5dEVRR2OyaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5j
PTAwMDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5vcmciLHJlc3BvbnNlPWQ
zODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNoYXJzZXQ9dXRmLTgK
</response>
```

解码响应是：

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

步 7：Server2 发送另一个[BASE64]编码挑战给 Server1：

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

解码挑战是：

```
rspauth=ea40f60335c427b5527b84dbabcdfffd
```

步 7（替换）：Server2 返回错误给 Server1：

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

步 8：Server1 响应挑战：

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

步 8（替换）：Server1 终止协商：

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

步 9：Server2 通知 Server1 成功认证：

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

步 9（替换）：Server2 通知 Server1 认证失败：

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

<aborted/>

</failure>

</stream:stream>

步 10：Server1 初始化一个新流给 Server2：

<stream:stream

xmlns='jabber:server'

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

to='example.com'

version='1.0'>

步 11：Server2 通过发送一个流头响应 Server1，并伴随着其它特征（或空特征元素）：

<stream:stream

xmlns='jabber:client'

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

from='example.com'

id='s2s_345'

version='1.0'>

<stream:features/>

7. 资源绑定

接收实体 SASL 协商（6）之后，初始实体可能想要或是需要绑定一个特殊资源至那个流。普通的，这仅用于客户端：为了遵从在此指定的寻址格式（3）与节传 送规则（10），必须有一个资源标识符联合客户端的<node@domain>（即可以由服务器产生也可以由客户应用提供）；这确保基于流使 用的地址是“全 JID”形式<node@domain/resource>。

根据在 SASL 协商中接收的一个成功指示，客户端必须发送一个新流头给服务器，服务器必须用可利用流特征列表中的内容来响应。特别的，如果服务器需要客户 端在 SASL 成功协商后，将资源绑定到流上，它必须包括一个由在流特征列表中的 'urn:ietf:params:xml:ns:xmpp-bind'命 名空间限定的空<bind/>元素。成功 SASL 协商后（不是前），它通过发送响应流的头表示给客户端：

服务器广告资源绑定特征给客户端：

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        id='c2s_345'
        from='example.com'
        version='1.0'>
<stream:features>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
</stream:features>
```

根据这样的通知，资源绑定是需要的，客户端必须靠送给服务器一个包含由'urn:ietf:params:xml:ns:xmpp-bind' 命名空间限定的，类型“set”（参考 IQ 语义（9.2.3））的 IQ 节，将资源绑定到流上。

如果客户端希望允许服务器代表自己产生资源标识符，它发送一个类型“set”的 IQ 节，包含一个空<bind/>元素：

客户端请求服务器绑定资源：

```
<iq type='set' id='bind_1'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
</iq>
```

支持资源绑定的服务器必须能代表一个客户端产生一个资源标识符。由服务器产生的资源标识符必须对<node@domain>是唯一的。如果客 户端希望指定资源标识符，它发送一个类型为“set”的 IQ 节，包含所需资源的标识符，作为<bind/>元素子元 素<resource/>的 XML 字符数据。

客户端绑定一个资源：

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

一旦服务器为客户端产生了一个资源标识符或是接受了由客户端提供的资源标识符，它必须返回一个类型为“result”的 IQ 节给客户端，必须包含一个<jid>子元素，来为服务器决定的已连接资源指定全 JID：

服务器通知客户端成功资源绑定：

```
<iq type='result' id='bind_2'>
    <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
        <jid>somenode@example.com/someresource</jid>
    </bind>
</iq>
```

服务器应当接受由客户端提供的资源标识符，但可能用一个服务器产生的资源标识符覆盖它；在这种情况，服务器不应当返回一个节错误 （例：<forbidden/>）给客户端，取而代之，应当以以上显示的 IQ 结果，传达产生的资源标识符给客户端。

当客户端提供一个资源标识符，以下节错误条件是可能的（参考节错误（9.3））：

1） 提供的资源标识符不能被与 Resourceprep（附录 B）一致的服务器处理。

2） 客户端不允许绑定资源到流上（例：因为结点或用户已经达到了在被允许的连接的资源的数目）。

3） 已提供资源标识符已经使用，但服务器并不允许用同样的标识符绑定多连接资源。

用于这些错误条件的协议显示如下。

资源标识符不能被处理：

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

客户端不允许绑定资源：

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

资源标识符在使用：

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

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

如果，完成资源绑定步骤之前，客户端尝试发送一个 XML 节，而不只是一个带有由'urn:ietf:params:xml:ns:xmpp-bind'命 名空间限定的<bind/>子元素的 IQ 节，服务器不准处理此节，并且，应当返回一个<not-authorized/>节错误 给客户端。

XMPP 3920 最靠谱的中文翻译文档（五）

8. 服务器回叫

8. 1 概述

Jabber 协议来自于 XMPP 适用的，包含一个“服务器回叫”方法，用以保护免受域哄骗，因此，使哄骗 XML 节更困难。服务器回叫并不是一个安全机制， 并且仅导致服务器身份弱验证（参考服务器到服务器的通信（14.4）相关方法的安全特性）。域需要健壮的安全性，应当使用 TLS 与 SASL；参考服务器到 服务器通信（4.4）细节。如果 SASL 用于服务器到服务器的认证，回叫不应当使用，因为它是不必要的。包含回叫文档主要是出于与现存实现与部署向后兼容 的原因。

服务器回叫方法因域名系统（DNS）存在而成为可能，由于一个服务器能够（正常的）对一个给定域发现授权服务器。因为回叫依靠 DNS，域内通信不准处理，直到由服务器宣称的域名系统（DNS）的主机名被解析（参考服务器到服务器的通信（14.4））。

服务器回叫是单向的，导致一个方向上一个流身份的（弱）验证。因为服务器回叫不是一个认证机制，通过回叫是不可能进行双向认证的。因此，服务器回叫必须在每个方向上完成，为了使在两个域间进行双向通信成为可能。

产生与验证密钥的方法用于服务器回叫，必须考虑被用的主机名，由接收服务器产生的流 ID，和由授权服务器的网络秘密知道。流 ID 在服务器回叫中是严格安全的，并且因此必须是即不可预测也不可重复的（参考[RANDOM]推荐资料相关用于安全观点的随机性。）

任何在回叫协商期间发生的错误必须考虑一个流错误，导致终止流与潜在的 TCP 连接。协议描述中说明的可能的错误条件如下。

以下术语应用：

1） 源服务器——试图在两个域间建立连接的服务器。
2） 接收服务器——尝试认证源服务器是否按它声明的那样去表达。
3） 授权服务器——回答由源服务器宣称的 DNS 主机名；对基本环境来说是源服务器，但在

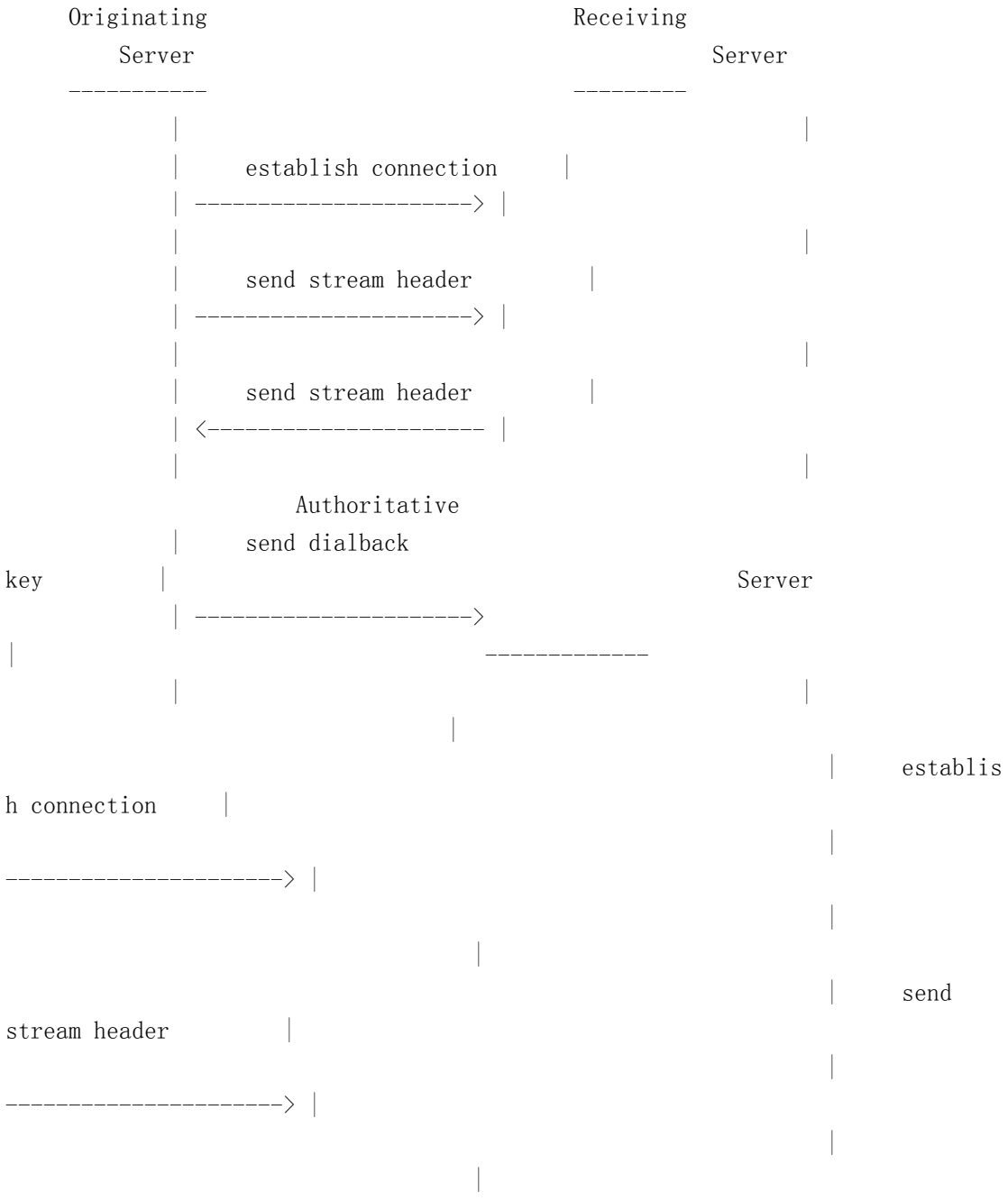
源服务器网络中可以是一个分离的机器。

8．2 事件顺序

以下是回叫事件顺序的简单总结：

1） 源服务器建立到接收服务器的连接。
2） 源服务器通过连接，给接收服务器发送‘key’值。
3） 接收服务器建立到认证服务器的连接。
4） 接收服务器向授权服务器发送相同的‘key’值。
5） 授权服务器回答密钥值是否有效。
6） 接收服务器通知源服务器授权是否通过。

我们可以将事件顺序以下图表示：




8. 3 协议

服务器间具体协议交互如下：

1） 源服务器建立 TCP 连接到接收服务器。

2） 源服务器发送流头给接收服务器：

<stream:stream

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

xmlns='jabber:server'

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

注：‘to’与‘from’属性在根流元素中是可选的。包含 xmlns:db 命名空间声明，名字显示向接收实体指示源服务器支持回叫。如果命名空间名不正 确，那么，接收服务器必须产生一个<invalid-namespace/>流错误条件并终止 XML 流与 TCP 连接。

3） 接收服务器应当发送一个流头返回给源服务器，包含一个用于交互的唯一的 ID：

<stream:stream

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

xmlns='jabber:server'

xmlns:db='jabber:server:dialback'

id='457F9224A0...'>

注：‘to’与‘from’属性在根流元素中是可选的。如果命名空间名不正确，那么源服务器必须产生一个<invalid- namespace/>流错误条件,并终止 XML 流与 TCP 连接。而且，接收服务器应当回应，但可能根据适当的安全策略默默终止 XML 流与 TCP 连 接。然而，如果接收服务器想要处理，它必须发送一个流头返回给源服务器。

4） 源服务器发送一个回叫密钥给接收服务器：

<db:result

```
        to='Receiving Server'
        from='Originating Server'>
    98AF014EDC0...
</db:result>
```

注：此密钥并不被接收服务器所检查，因为接收服务器并不保存相关源服务器间会话信息。由源服务器产生的密钥必须部分基于接收服务器在先前步骤提供的 ID 值，并部分基于源服务器与授权服务器的保密共享。如果‘to’地址值并不与接收服务器所识别的主机名匹配，那么，接收服务器必须产生一 个<host-unknown/>流错误条件并终止 XML 流与潜在的 TCP 连接。如果‘from’地址值与带有接收服务器已经建立的连接的域 匹配，那么，接收服务器可能选择为新连接产生一个<not-authorized/>流错误条件，然后终止 XML 流与潜在的与新请求相关的 TCP 连接。

5） 接收服务器建立一个 TCP 连接支持由源服务器宣称的域，作为它连接到授权服务器的结果。（注意：作为优化，一个实现可能重用一个现存的连接。）

6） 接收服务器发送给授权服务器一个流头：

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

注：在根流元素中，‘to’与‘from’属性是可选的。如果命名空间名不正确，则授权服务器必须产生一个<invalid-namespace/>流错误条件并终止两个 XML 流与潜在的 TCP 连接。

7） 授权服务器发送给接收服务器一个流头：

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

注：如果命名空间名不正确，则接收服务器必须产生一个<invalid-namespace/>流错误条件并终止它与授权服务器间的两个 XML 流与潜在的 TCP 连接。如果流错误发生在接收服务器与授权服务器间，则接收服务器必须产生一个<remote-connection- failed/>流错误条件并终止它与发起服务器间的两个 XML 流与潜在的 TCP 连接。

8） 接收服务器发给授权服务器要求认证密钥的请求：

```
<db:verify
        from='Receiving Server'
        to='Originating Server'
        id='457F9224A0...'>
    98AF014EDC0...
</db:verify>
```

注：经过这儿的是来自接收服务器的流头的主机名、源标识符，到步骤 3 中的发起服务器，源服务器发送给接收服务器的密钥在步骤 4。根据这些信息，还有授权服 务器网络中的共享密钥信息，密钥被验证。任何验证方法可能用于产生密钥。如果‘to’地址值与授权服务器识别的主机名不匹配，那么，授权服务器必须产生一 个<host-unknown/>流错误条件并终止两个 XML 与潜在的 TCP 连接。如果‘from’地址值与源服务器打开 TCP 连接时（或任 意相关有效域，例如接收服务器的主机名或其它有效域一个有效子域）所表

示的主机名不匹配，则授权服务器必须产生一个<invalid- from/>流错误条件并终止两个 XML 流与潜在的 TCP 连接。

9） 授权服务器验证密钥是否有效

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

或

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

注：如果 ID 与步骤 3 中的接收服务器不匹配，那么接收服务器必须产生一个<invalid-id/>流错误条件并终止两个 XML 流与潜在的 TCP 连接。如果‘to’地址值与接收服务器所识别的主机名不匹配，则接收服务器必须产生一个<host-unknown/>流错误条件并终 止两个 XML 流与潜在的 TXP 连接。如果‘from’地址值与源服务器打开 TCP 连接时（或任意相关有效域，例如接收服务器的主机名或其它有效域一个有效 子域）所表示的主机名不匹配，则接收服务器必须产生一个<invalid-from/>流错误条件并终止两个 XML 流与潜在的 TCP 连接。返 回认证信息给接收服务器之后，授权服务器应当终止他们之间的流。

10） 接收服务器通知源服务器结果：

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

注：在这里，连接可通过一个 type='valid'或报告为无效来被认证。如果连接无效，则接收服务器必须终止两个 XML 流与潜在的 TCP 连接。如果连 接被认证，数据可被源服务器发送并被接收服务器读取；在此这前，所有发送给接收服务器的 XML 节应该默默被扔掉。

前述结果是接收服务器已经认证了源服务器的身份，为了节通过“初始流”（如，从源服务器到接收服务器的流）的 XML 能被源服务器发送与接收服务器能接收，为了验证使用“响应流”（如，从接收服务器到源服务器）实体的身份，回叫必须以相反方向完成。

成功回叫协调后，接收服务器应当接收来自通过现存已认证连接的源服务器的子序列<db:result/>包（例如，认证需求发送到子域或其它 由接收服务器服务主机名）；这使在一个方向上的原来的已认证连接的"piggybacking"成为可能。

即使回叫协调成功，服务器必须认证从其它服务器接收的 XML 节，包括‘from’属性与‘to’属性；如果一个节并不满足此限制，接收节的服务器必须产生 一个

<improper-addressing/>流错误条件并终止两个 XML 流与潜在的 TCP 连接。进一步讲，服务器必须认证从其它服务 器，包括流的一个已验证域的‘from’属性；如果节并不满足此限制，收到节的服务器必须产生一个<invalid-from/>流错误条件 并终止两个 XML 流与潜在的 TCP 连接。这两个检查有助于阻止关联到特别节的哄骗。

9. XML 节

TLS 协商（5 节）后，如果需要 SASL 协商（6 节）与资源绑定（7 节），XML 节可通过流来发送。定义了三种 XML 节用于 ’jabber:client’与’jabber:server’命名空间：<message/>, <presence/>, and <iq/>。另外，这种节有五个通用属性。这些通用属性，像三种节的基本语义一样，都定义在此；与即时消息与表示应用相关的 XML 节的更详细 信息在[XMPP-IM]中提供。

9. 1 通用属性

以下五个属性对 message, presence 与 IQ 均通用：

9. 1. 1 to

‘to’属性指定接收节的 JID。

在‘jabber:client’命名空间中，节应当处理‘to’属性，虽然，由服务器处理的从客户端到服务器端的节不应该拥有‘to’属性。

在’jabber:server’命名空间中，节必须拥有‘to’属性；如果服务器收到一个不满足此限制的节，它必须产生一个<improper-addressing/>流错误条件并终止两个 XML 流与错误服务器的潜在连接。

如果‘to’属性无效或不能连接，发现此事实的（通常是发送的或接收的服务器）实体必须返回一个合适的错误给发送者，设置错误节的‘from’属性为错误服务器提供的‘to’属性值。

9. 1. 2 from

‘from’属性指明发送者的 IID。

当服务器收到一个在由’jabber:client’命名空间认证的已授权流的上下文中的 XML 节，它必须做以下事件之一：

1） 验证客户端提供的‘from’属性值就是用于联合实体的已连接资源的值。

2） 加一个‘from’地址值给节，此节的值是裸 JID（<node@domain>）或全 JID（<node@domain/resource>），这些 JID 由服务器决定用于产生节的已联接资源（看地址决定（3.5 节））。

如果一个客户端试图发送‘from’属性并不匹配实体的已联接资源的 XML 节，服务器应该返回一个<invalid-from/>流错误给客 户端。如果一个客户端试图通过一个流来发送一个还未授权的 XML 节，服务器应当返回一个<not-authorized/>流错误给客户端。 如果产生了，这些条件都必须关闭流并终止潜在的 TCP 连接；这有助于阻止来自于欺诈客户端的否认服务攻击。

当一个服务器产生一个来自于服务器本身的节，用于传送到一个已连接的客户端（例如：在由服务器代表客户端提供的数据存储服务的上下文中），节必须既（1）不包括‘from’属性或（2）包括‘from’属性，其值是帐户的裸 JID（<node@domain>）或客户的全 JID（<node@domain/resource>）。服务器不准发送给客户端一个不包括‘from’

属性的节，它必须设想节是从服务器 到已连接客户端。

在'jabber:server'命名空间中，一个节必须处理一个‘from’属性；如果服务器收到不满足此限制的节，它必须产生一 个<improper-addressing/>流错误条件。更进一步，包含在‘from’属性中的 JID 的域标识符部分必须匹配发送服务器 （或任何已认证相关域，如发送服务器的主机名或其它由发送服务器已认证域）的主机名，当在 SASL 协商或回叫协商通信中；如果一个服务器收到一个不满足此 约束的节，它必须产生一个<invalid-from/>流错误条件。这些条件都必须关闭流并终止潜在的 TCP 连接；这有助于阻止欺诈服务器 的否认服务攻击。

9．1．3 id

可选‘id’属性可能由发送实体因内部跟踪收发(特别是跟踪固有在 IQ 节语义中的请求-响应交互）节而使用。对值‘id’属性来说，它是可选的唯一全局的，在域内的或流中的。IQ 节语义强加了其它约束；看 IQ 语义（9.2.3）。

9．1．4 type

类型域属性指定目的或消息上下文，出席或 IQ 节的详细信息。‘type’属性的特别允许值依赖节是否是一个消息，出席，或 IQ；消息与出席节的值是特别用 于即时消息与出席应用的，并因此定义义在[XMPP-IM]，然而 IQ 节的值特指 IQ 节在一个结构化的请求-响应“会话”中的角色，并因此定义在以下 IQ 语义（9.2.3 节）。对三种节仅有的一个通用‘type’值是“error”；看节错误（9.3 节）。

9．1．5 xml:lang

此节应当处理一个‘xml:lang’属性（定义在[XML]2.2 节），如果节包含倾向于表示到一个人类用户（RFC2277[CHARSET]中有解 释，“对人的国际化”）的 XML 字符数据。‘xml:lang’属性值指定任意人类可读 XML 字符数据的缺省语言，可能被特定的子元素的 ‘xml:lang’属性覆盖。如果节没有‘xml:lang’属性，实现必须设想为流指定的缺省语言已在以下流属性（4。4 节）中定义。 ‘xml:lang’属性的值必须是一个 NMTOKEN 并必须遵从定义在 3066[LANGTAGS]中的格式。

9．2 基本语义

9．2．1 消息语义

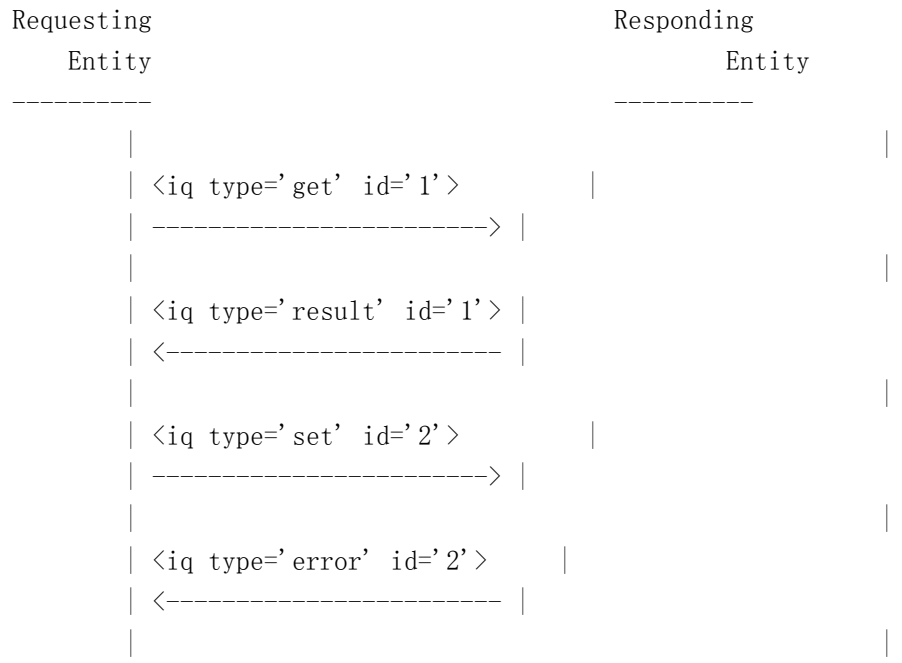
<message/>节种类可被看作“推”机制，一个实体推信息给其它实体，与 EMAIL 系统中发生的通信类似。所有消息节应该拥有‘to’ 属性，指定有意的消息接收者；根据接收到那样的一个节，服务器应该路由或传送它到有意的接收者（参考服务器处理用于相关 XML 节的通用路由与传送规则 XML 节的规则（10 节））。

9．2．2 出席语义

<presence/>元素可被看作基本广播或“出版-订阅”机制，多实体收到他们已订阅（在这种情况下，网络可利用信息）实体的信息。总的 来说，出版实体应该发送一个不带‘to’属性的出席节，在这种情况下，与此实体相连的服务器应该广播或复用节给所有订阅实体。然而，一个出版实体也可能发 送一个带有‘to’属性的出席节，此种情况下，服务器应该路由或传送节到有意的接收者。参考处理 XML 节（10 节）的服务器规则，用于通用路由与相关 XML 节的传送规则，并且用于即时消息与出席应用的出席-特定规则[XMPP-IM]。

9．2．3 IQ语义

信息/请求，或IQ，是一个请求-响应机制，与[HTTP]在某些方面相似。IQ语义让一个实体向其它实体请求或接收其它实体的响应成为可能。请求与响应 的数据内容由IQ无素的直接子元素的命名空间声明定义，并且，交互由请求实体通过使用‘id’属性来跟踪。因此，IQ交互遵从结构化数据交换的一个通用模 式，此交换例如得到/结果或设置/结果（虽然如果合适的话，对一个请求的响应可能会以错误返回）：




为了加强这些语义，以下规则应用：

1） 对IQ节来说，‘id’属性是REQUIRED。

2） 对IQ节来说。‘type’属性是需要的。值必须是以下之一：

*get——节是一个用于信息或需求的请求。

*set——节提供所需数据，设置新值，或替换现存值。

*result——节是成功得到或设置请求的响应。

*error——先前发送得到或设置的相关过程或传送的错误（参考节错误（9.3节））。

3）收到类型为“get”或“set”的IQ请求的实体必须以类型为“result”或“error”的IQ响应来响应（响应必须保留请求的‘id’属性）。

4） 收到类型为“result”或“error”的节不准靠发送一个进一步的类型为“result”或“error”的IQ响应节来响应；然而，如以上显示，请求实体可能发送另一个请求（如：一个类型为“set”的IQ，为了提供通过得到/结果对发现的所需的信息）。

5） 类型为“get”或“set”的IQ节必须包含一个并仅有一个子元素，指定特别的请求或响应语义。

6） 一个类型为“result”的IQ节必须包含0或一个子元素。

7） 类型为“error”类型的IQ节应当包含在相关“get”或“set”子元素中，并且，必须包含一个<error/>子元素；详细信息，参考节错误（9.3节）。

9．3 节错误

节相关错误以类似流错误（4.7 节）的方式处理。然而，不像流错误，节错误不可是不可恢复的；因此，暗含相关源发送者行为的错误节能按顺序纠正错误。

9．3．1 规则

以下规则应用于节相关错误：

1） 检测相关节错误条件的接收或处理实体必须返回给发送实体一个同种节（消息，出席或 IQ），它的‘type’属性被设置成值“error”（那样的节在此被称为“错误节”）。

2） 产生错误节的实体应当包含被送的源 XML，为了发送者能够检测，并且，如果必要的话，在试图重送前纠正 XML。

3） 一个错误节必须包含一个<error/>子元素。

4） 一个<error/>子元素不准被包括，如果‘type’属性有不止一个“错误”值（或无‘类型’属性）。

5） 接收一个错误节的实体不准响应带有进一步错误节的节；这有助于阻止循环。

9．3．2 语法

节相关错误语法如下：

```
<stanza-kind to='sender' type='error'>
    [RECOMMENDED to include sender XML here]
    <error type='error-type'>
        <defined-condition xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
        <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'
                    xml:lang='langcode'>
            OPTIONAL descriptive text
        </text>
        [OPTIONAL application-specific condition element]
    </error>
</stanza-kind>
```

节种类是消息、出席或 iq 之一。

<error/>元素的‘type’属性值必须是以下之一：

*cancel——不重试（错误不可恢复）

*continue——进行（仅是一个警告条件）

*modify——改变数据发送后重试

*auth——提供信任后重试

*wait——等待之后重试（错误是临时的）

<error/>元素：

必须包含一个子元素，此子元素与以下指定的已定义的节错误条件一致；此元素必须被'urn:ietf:params:xml:ns:xmpp-stanzas'命名空间所认证。

可能包含<text/>子元素，此子元素包含 XML 字符数据，用于描绘更细节的错误；此元素必须被'urn:ietf:params:xml:ns:xmpp-stanzas'命名空间所认证，并且应该拥有一个'xml:lang'属性。

可能包含一个子元素，用于特殊-应用错误条件；此元素必须由一个已定义-应用命名空间认证，并且，它的结构由此命名空间定义。

<text/>元素是可选的。如果包括在内，它应当仅用于提供描述性或诊断性信息，这些信息用于补充已定义条件或特殊-应用条件的意思。它不 应当由应用程序性的描述。它不应当用作向用户表达的错误信息，但可能显示除与包含条件元素（或元素们）相关的错误消息。

最后，为维护向后兼容性，此方案（在[XMPP-IM]中指定的）允许可选的在<error/>元素中包含‘code’属性。

9．3．3 已定义条件

以下条件被定义用于节错误。

<bad-request/>——发送者已发送畸形的或不能被处理的（例如，一个包含未识别‘type’属性值的 IQ 节）XML；相关错误类型应当是“modify”。

<conflict/>——访问不被授权，因为一个现存资源或会话以同样名字或地址存在；相关错误类型应当是“cancel”。

<feature-not-implemented/>——被请求特征未被接收者或服务器实现，并且因此不能被处理；相关错误类型应当是“cancel”。

<forbidden/>——请求实体不拥有执行行为的所需许可；相关错误类型应当是“auth”。

<gone/>——接收者或服务器不在以此地址联系（错误节可能包含一个新地址在<gone/>元素的 XML 字符数据中）；相关错误类型应当是“modify”。

<internal-server-error/>——服务器不能处理节，因为错误配置或一个另外-未定义内部服务器错误；相关错误类型应当是“wait”。

<item-not-found/>——JID 地址或被请求项不能被发现；相关错误类型应当是“cancel”。

<jid-malformed/>——已提供的发送实体或与一个 XMLL 地址（例：‘to’属性值）通信或其它方面（例：资源标识符）与地址方案（3 节）中定义的语法不符；相关错误类型应当是“modify”。

<not-acceptable/>——接收者或服务器理解请求，但拒绝处理它，因为它不满足由接收者或服务器（例：消息中相关可接受字的本地策略）所定义的标准；相关错误类型应当是“modify”。

<not-allowed/>——接收者或服务器不允许任何实体执行动作；相关错误类型应当是“cancel”。

<not-authorized/>——发送者必须在被允许执行动作前提供合适的信任，或已经提供不合适的信任；相关错误类型应当是“auth”。

<payment-required/>——请求实体未授权去访问被需求服务，因为需要付费；相关错误类型应当是“auth”。

<recipient-unavailable/>——有意的接收者临时不可用；相关错误类型应当是“wait”（注：一个应用不准返回此错误，如果这样做将提供关于意向接收 者对未授权知道此类信息的实体的网络可利用性信息）。

<redirect/>——接收者或服务器为此信息重定向请求到其他实体，通常是临时的（错误节应当包含可替换地址，必须是一个有效的 JID，在<redirect/>元素的 XML 字符数据中）；相关错误类型应当是“modify”。

<registration-required/>——请求实体未被授权访问所请求服务，因为需要注册；相关错误类型应当是“auth”。

<remote-server-not-found/>——一个指定作为意向接收者的部分或全部的 JID 的远程服务器或服务不存在；相关错误类型应当是“cancel”。

<remote-server-timeout/>——一个指定作为意向接收者（或被请求去执行一个请求）的部分或全部的 JID 的远程服务器或服务不能在一个合理的时间内被联系到;相关错误类型应当是“wait”。

<resource-constraint/>——服务器或接收者缺少必要的系统资源去服务请求;相关错误类型应当是“wait”。

<service-unavailable/>——服务器或接收者当前并不提供所请求的服务;相关错误类型应当是“cancel”。

<subscription-required/>——请求实体不被授权访问被请求服务，因为需要订阅；相关错误类型应当是“auth”。

<undefined-condition/>——错误条件并不是此列表中由其它条件定义的那些之一;任何错误类型可能与此条件相关，并且，它应当仅用于与一个特殊-应用条件相连。

<unexpected-request/>——接收者或服务器理解请求，但此时（例：请求无序/请求不在状态）并不期望它；相关错误类型应当是“wait”。

9．3．4 特殊-应用条件

像所知道的,一个应用可能靠包含一个错误元素中的合适的-命名空间的子元素来提供特殊-应用节错误信息。特殊-应用元素应当补充或进一步认证一个已定义元素。因此，<error/>元素将包含两个或三个子元素：

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

```
        <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
        <too-many-parameters xmlns='application-ns'/>
    </error>
</iq>

<message type='error' id='another-id'>
    <error type='modify'>
        <undefined-condition
                    xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
        <text xml:lang='en'
                    xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
            Some special application diagnostic information...
        </text>
        <special-application-condition xmlns='application-ns'/>
    </error>
</message>
```

XMPP 3920 最靠谱的中文翻译文档（六）

10． 处理 XML 节的服务器规则

兼容服务器实现必须确保有序处理任两实体间的 XML 节。

超出有序处理的需求，每个服务器实现将包含它自己的“传送树”用于处理它所接收的节。那样的一个树决定是否一个节需要被路由到其它域，内部处理，或传送到与被连节点相关的资源。以下规则应用：

10．1 无‘to’地址

如果节拥有无‘to’属性，服务器应当代表发送它的实体处理它。因为所有从其它服务器收到的节必须拥有一个‘to’属性，此规则仅应用于从一个连到服务器 的已注册实体（如客户端）收到的节。如果服务器收到一个无‘to’属性的出席节，服务器应当广播它到被订阅到发送实体的出席实体，如果可利用的话（用于定 义在[XMPP-IP]即时消息与表示应用的出席广播的语义。）如果服务器接收一个类型为“get”或“set”的没有‘to’属性的 IQ 节，并且它理解 认证节内容的命名空间，它必须也能代表发送实体处理节或返回给发送实体（在“process”意思处被认证命名空间的语义决定）一个错误。

10．2 外部域

如果 JID 的域标识符部分的主机包含在‘to’属性中并不匹配服务器本身的已配置主机名

或子域中的已配置主机之一，服务器应当路由节到外部域（服从本地服务提供与相关内部域通信的安全策略）。有两种可能情况：

一个服务器到服务器流已在两域间存在：发送者的服务器为现存流的外部域路由节到已授权服务器。

两域间存在无主机到主机流：发送者的服务器（1）解析外部域（定义在以下服务器到服务器通信（节 14。4））的主机名，（2）在两域间（定义在如下使用 TLS（节 5）并且使用 SASL（节 6））协商服务器到服务器的流，并（3）为通过新近-建立的流的外部域路由节到授权服务器。

如果路由到接收者的服务器不成功，发送者的服务器必须返回一个错误给发送者；如果接收者的服务器能被联系但被接收者的服务器传送到接收者是不成功的，接收者的服务器必须经由发送者的服务器返回一个错误给发送者。

10．3 子域

如果包含在‘to’属性中的 JID 域标识符部分的主机名匹配服务器本身已配置主机名之一的子域，服务器必须也处理节本身或路由节到一个特别的对那个子域（如果子域被配置）有责任的服务，或返回一个错误给发送者（如果子域不被配置）。

10．4 仅有域或特别资源

如果包含在‘to’属性中的 JID 域标识符部分的主机名匹配服务器本身的一个已配置主机名，并且包含在‘to’属性中的 JID 是<domain>或<domain/resource>形式，服务器（或在内的一个已定义资源）必须合乎节种类处理节或返回错 误节给发送者。

10．5 同域中的节点

如果包含在‘to’属性中的 JID 域标识符部分的主机名匹配服务器本身的一个已配置主机名，并且包含在‘to’属性中的 JID 是<node@domain>或<node@domain/resource>形式，服务器应当传送节到由包含在‘to’属性中 的 JID 表达的节的意向接收者。以下规则应用：

1） 如果 JID 包含一个资源标识符（例：是<node@domain/resource>形式）并且，这儿存在一个已连接资源匹配全 JID，接收者的服务器应当传送的节到确切匹配此资源标识符流或会话。

2） 如果 JID 包含一个资源标识符并且这儿存在匹配全 JID 的无连接资源，接收者的服务器应当返回一个<service-unavailable/>节错误给发送者。

3） 如果 JID 是<node@domain>形式，并且这儿存在为此结点的至少一个已连接资源，接收者的服务器应当传送节到连接资源的至少一个，根据应用-特殊规则（一套传送规则，用于定义在[XMPP-IM]即时消息与出席应用）。

## 11． XMPP 内的 XML 使用

### 11．1 约束

XMPP 是流 XML 元素的一个简单与特殊的协议，用来近实时的交换结构化信息。由于 XMPP 不需要任意分析与完整 XML 文档，这儿没有 XMPP 需要支持[XML]全特征的需求。特别的，以下约束应用。

关于 XML 产生，一个 XMPP 实现不准注入以下任意一个 XML 流：

*评论（定义在[XML]节 2。5）

*处理说明（2。6 节）

*内部或外部 DTD 子集（2。8 节）

*除了预定义实体（4。6 节）的内部或外部实体参考。

*包含映射到预定义实体（4。6 节）保留字符的字符数据或属性值；那样的字符必须被避免

关于 XML 处理，如果一个 XMPP 实现接收到那样的约束 XML 数据，它必须忽略此数据。

### 11．2 XML 命名空间名与前缀

XML 命名空间[XML-NAMES]被用在所有与 XMPP-兼容的 XML 中，去创建数据拥有权的严格界限。命名空间的基本功能是分离结构的混合在一起的 XML 元素的不同词汇。确保 XMPP-兼容 XML 是命名空间-了解使任意允许的 XML 能够与 XMPP 中的任意数据元素结构化的混合。XML 命名空间名与前 缀的规则定义在以下子部分。

11．2．1 流命名空间

流命名空间声明在所有 XML 流头中都是需要的。流命名空间名必须是'http://etherx.jabber.org /streams'。<stream/>元素与它的<features/>与<error/>子元素的元素名必须被 所有实例中的流命名空间认定合格。一个实现应当为那些元素产生仅有的'stream:'前缀，并且因为历史原因可能接受仅有的'stream:'前缀。

11．2．2 缺省命名空间

缺省命名空间声明是需要的，并且用在所有 XML 流中，为了定义允许的根流元素的第一级子元素。此命名空间声明必须与初始流与响应流相同，为了两个流一致的 被认证合格。缺省命名空间声明应用于流与所有在由其它命名空间认证合格的流（除非由另一命名空间显示认定合格，或由流命名空间或回叫命名空间前缀认证）中 发送的节。

服务器实现必须支持以下两个缺省命名空间（由于历史原因，一些实现可能支持仅有的那些两个缺省命名空间）：

*jabber:client——缺省命名空间，当流用于客户端与服务器通信时所声明的。

*jabber:server——缺省命名空间，当流用于两服务器间通信时声明的。

客户端实现必须支持'jabber:client'缺省命名空间，并且由于历史原因可能只支持缺省命名空间。

实现不准为缺省命名空间中的元素产生命名空间前缀，如果缺省命名空间是'jabber:client'或'jabber:server'。一个实现不应当 为元素产生命名空间前缀，元素由'jabber:client'与'jabber:server'之外的内容（与流相反）命名空间认证的。

注：'jabber:client'与'jabber:server'命名空间是接近同一的，但用在不同的上下文中（客户端到服务顺通信用 'jabber:client'与服务器到服务器通信用'jabber:server'）。这两个仅有的不同是‘to’与‘from’属性在 'jabber:client'中发送的节中是可选的，然而在'jabber:server'中发送的节是必须的。如果一个兼容实现接受一个由 'jabber:client'或'jabber:server'命名空间认证合格的流，它必须支持所有三个核心节种类的（消息，出席，与 IQ）通用属性 （9。1 节）与

基本语义（9。2 节）。

11．2．3 回叫命名空间

回叫命名空间声明对于所有用在服务器回叫（8 节）中的元素都是需要的。回叫命名空间的名字必须是'jabber:server:dialback'。所有 由这个命名空间认证合格的元素必须被加前缀。一个实现应当为那种元素仅产生'db:'前缀并可能接受仅有的'db:'前缀。

11．3 确认（验证）

除了'jabber:server'命名空间中节的相关‘to’与‘from’地址，服务器不为转发到客户端或另一个服务器的 XML 元素负责；一个实现可 能选择提供仅有的认证数据元素，但这是可选的（虽然一个实现不准接受 XML，那也不是好格式）。客户端不应当依赖此能力去发送数据，这些数据与方案并不 符，并且应当忽略一个来的 XML 流中的非构造元素或属性。XML 流与节的验证是可选的，包含在此的方案仅用于描述目的。

11．4 包含文本声明

实现应当在发送流头之前发送文本声明。应用必须遵循文本声明包含在内的相关环境的[XML]中的规则。

11．5 字符编码

实现必须支持 UTF-8 (RFC 3629 [UTF-8])统一字符集(ISO/IEC 10646-1 [UCS2])字符传输，RFC 2277 [CHARSET]中查。实现不准试图使用其它编码。