# [XMPP 协议工作流程详解](http://blog.csdn.net/huyoo/article/details/24353105)

转载自：http://blog.csdn.net/huyoo/article/details/24353105

原文: http://ceit.uq.edu.au/content/how-xmpp-works-step-step

作者: Yilun Fan, 日期 2011-01-05 13:09

XMPP 核心协议 http://xmpp.org/rfcs/rfc3920.html

XMPP 要点.

* 1. 客户端(C) 和服务器端(S) 通过TCP连接5222端口进行全双工通信.
* 2. XMPP 信息均包含在 XML streams中.一个XMPP会话, 开始于<stream> 标签, 并结束于</stream>标签.所有其他的信息都位于这俩标签之间.
* 3. 出于安全目的考虑, 开始<stream>之后, 后续的内容会被适度的使用 Transpor Layer Security (TLS) 协商传输 和强制性的 Simple Authentication 和 Security Layer (SASL) 协商传输.
* 4. SASL协商完成后, 一个新的 stream 将会被迅速打开, 它将会更加安全和保密.

**第一步: 打开 stream**

Client: 客户端发送打开 stream 的片段到服务器, 请求一个新的 session.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

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

这里 “example.com” 是客户端试图连接的服务器的域名.

Server: Server 返回 XML stream, 以<stream:freatures> 开头, 包含要求 TLS 或者 SASL 协商谈判之一, 或者2个都要求.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<stream:features>**
2. **<starttls** xmlns='urn:ietf:params:xml:ns:xmpp-tls'**>**
3. **<required/>**
4. **</starttls>**
5. **<mechanisms** xmlns='urn:ietf:params:xml:ns:xmpp-sasl'**>**
6. **<mechanism>**DIGEST-MD5**</mechanism>**
7. **<mechanism>**PLAIN**</mechanism>**
8. **<mechanism>**EXTERNAL**</mechanism>**
9. **</mechanisms>**
10. **</stream:features>**

**第二步: 加密和认证.**

**2.1 如果服务器需要 TLS 交涉.**

Client: 客户端发送 STARTTLS 到服务器.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

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

Server: 服务器返回消息显示 TLS 已被允许:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

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

或者 TLS失败了:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

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

在失败的情况下, 服务器会关闭 TCP 连接.

Client: 如果 TLS 已被服务器正确处理, 客户端发送请求一个新的 session:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

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

Server: 服务器响应一个 XML stream, 指示是否需要 SASL 交涉.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<stream:stream** xmlns='jabber:client' xmlns:stream='http://etherx.jabber.org/streams' from='example.com' id='c2s\_234' version='1.0'**>**
2. **<stream:features>**
3. **<mechanisms** xmlns='urn:ietf:params:xml:ns:xmpp-sasl'**>**
4. **<mechanism>**DIGEST-MD5**</mechanism>**
5. **<mechanism>**PLAIN**</mechanism>**
6. **<mechanism>**EXTERNAL**</mechanism>**
7. **</mechanisms>**
8. **</stream:features>**

**2.2 SASL 交涉**

Client 客户端需要选择一个服务器上有效的认证方式来携带SASL交涉数据, 上面的情况, “DIGEST-MD5“, “PLAIN” 和 “EXTERNAL” 是一些可选项.  
  
“PLAIN” 认证模式是三者之中最简单的了. 它是这样工作的:  
  
Client: 客户端按照自己选择的认证模式发送一个将用户名和密码以base64编码的 stream. 用户名和密码按这种格式组织:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. “\0UserName\0Password”.

例如我想以用户名为“mbed@ceit.org”登录, 密码是“mirror”. 那么, 在进行base64编码之前, 用户名和密码按照上面的格式组织为一个新的字符串,“\0mbed\0mirror”, 再进行base64编码, 得到字符串“AG1iZWQAbWlycm9y”.

然后, 客户端发送下列 stream 到服务器.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<auth** xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='PLAIN'**>**AG1iZWQAbWlycm9y**</auth>**

Server: 如果服务器接受了认证信息, 服务器会发回 带 “success” 标签的 stream.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

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

或者:  
  
Server: 如果密码和用户名不匹配, 或者上面的base64编码有错误, 服务器发回错误信息的 stream.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<failure** xmlns='urn:ietf:params:xml:ns:xmpp-sasl'**/>**

“DIGEST-MD5” 认证模式的具体方法可以在这里找到: http://www.ietf.org/rfc/rfc2831.txt.

**第三步: 资源绑定(可选)**

Client: 客户端要求服务器绑定一个资源(可以理解为客户端的类型, 比如电脑, 手机, Web应用等):

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<iq** type='set' id='bind\_1'**>**
2. **<bind** xmlns='urn:ietf:params:xml:ns:xmpp-bind'**/>**
3. **</iq>**

或者  
Client: 客户端自己绑定一个资源:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<iq** type='set' id='bind\_2'**>**
2. **<bind** xmlns='urn:ietf:params:xml:ns:xmpp-bind'**>**
3. **<resource>**someresource**</resource>**
4. **</bind>**
5. **</iq>**

Server: 服务器发回另外一个<iq>片段, 如果“type” 标签的内容是“result”, 说明绑定是成功的, 否则说明绑定失败.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<iq** type='result' id='bind\_2'**>**
2. **<bind** xmlns='urn:ietf:params:xml:ns:xmpp-bind'**>**
3. **<jid>**somenode@example.com/someresource**</jid>**
4. **</bind>**
5. **</iq>**

**第四步: 请求一个新的session**

在 SASL 交涉完成之后或者可选资源绑定之后, 客户端必须建立一个 session 来开始即时消息发送和接收.  
  
Client: 客户端向服务器发送请求:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<iq** to='example.com' type='set' id='sess\_1'**>**
2. **<session** xmlns='urn:ietf:params:xml:ns:xmpp-session'**/>**
3. **</iq>**

Server: 服务器发回一个<iq> 片段表明 session 是否成功创建.  
  
创建成功的消息类似于:

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<iq** from='example.com' type='result' id='sess\_1'**/>**

如果服务器未能创建 session, 服务器将会回复一个如下消息或者其他类型的错误消息.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<iq** from='example.com' type='error' id='sess\_1'**>**
2. **<session** xmlns='urn:ietf:params:xml:ns:xmpp-session'**/>**
3. **<error** type='auth'**>**
4. **<forbidden** xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'**/>**
5. **</error>**
6. **</iq>**

**第五步: 客户端和服务器交换 XMPP 片段**

如果以上步骤均成功完成, 那么客户端就可以发送 XMPP 片段到服务器和接收 XML stream了.

客户端可以发送 <iq> 片段来向服务器请求 roster 或者其他信息. 并可以使用 <presence> 片段来改变客户端的 presence 状态(比如在线, 离开等)

即时消息和其他的负载可以通过发送 <message> 片段来完成.

**第六步: 关闭 stream**

最后, 如果客户端想要结束聊天和关闭 XMPP session, 客户端需要发送一个关闭 stream的片段到服务器.

**[html]** [view plain](http://blog.csdn.net/huyoo/article/details/24353105) [copy](http://blog.csdn.net/huyoo/article/details/24353105)

[在CODE上查看代码片](https://code.csdn.net/snippets/307744)

1. **<presence** type='unavailable'**/>**
2. /stream:stream**>**

然后, 服务器将会改变客户端的 presence 状态为 “Offline” , 并且关闭 和客户端的 TCP 连接.