XEP-0225: Component Connections

Peter Saint-Andre
mailto:peter@andyet.net
xmpp:stpeter@stpeter.im
https://stpeter.im/

2008-10-06
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	component

This document specifies a standards-track XMPP protocol extension that enables server components to connect to XMPP servers.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 - 2016 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF's Intellectual Property Rights Policy (a copy of which can be found at <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

Jabber Component Protocol (XEP-0114) [1] defines a protocol that enables a server component to connect to an XMPP server. However, there are a number of perceived limitations with that protocol:

- It does not support Transport Layer Security (TLS; see RFC 5246 [2]) for channel encryption.
- It does not support the Simple Authentication and Security Layer (SASL; see RFC 4422 [3]) for authentication.
- It does not enable a component to bind multiple hostnames to one stream (as, for example, a client can bind multiple resource identifiers).
- It multiplies namespaces beyond necessity, adding the "jabber:component:accept" and "jabber:component:connect" namespaces to "jabber:client" and "jabber:server".

This document specifies a standards-track protocol that addresses the basic requirements for component connections. In the future, additional documents may specify more advanced features on top of the protocol defined herein.

2 Requirements

This document addresses the following requirements:

1. Support Transport Layer Security for channel encryption.
2. Support the Simple Authentication and Security Layer for authentication.
3. Enable a component to bind multiple hostnames to one stream.
4. Use one of the existing default namespaces for XML streams between components and servers.

3 Stream Establishment

XML streams are established between a component and a server exactly as they are between a client and a server as specified in XMPP Core [4], with the following exceptions:

[1] XEP-0114: Jabber Component Protocol <http://xmpp.org/extensions/xep-0114.html>.
[2] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[3] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.
[4] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

1. The 'from' address of the initial stream header SHOULD be the "default" hostname of the component.
2. The JID asserted by the end entity (in this case a component) during STARTTLS negotiation and SASL negotiation MUST be of the form <domain> in conformance with the definition of a domain identifier from XMPP Core.
3. If a "simple user name" is included in accordance with the chosen SASL mechanism, it MUST be of the form <domain> in conformance with the definition of a domain identifier from XMPP Core.

4 Hostname Binding

The protocol defined in XEP-0114 depended on use of the 'to' address in the stream header to specify the hostname of the component. By contrast, client-to-server connections use stream establishment is followed by binding of a resource to the stream (in fact multiple resources can be bound to the stream). This protocol emulates client-to-server connections by using a hostname binding process that is similar to the resource binding process specified in XMPP Core.
If a server offers component binding over a stream, it MUST advertise a feature of "urn:xmpp:component:0".

Listing 1: Stream Feature

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

S: <stream:features>
     <bind xmlns='urn:xmpp:component:0'>
       <required/>
     </bind>
   </stream:features>
```

In order to bind a hostname, the component sends a bind request to the server.

Listing 2: Bind Request

```
C: <iq id='bind_1' type='set'>
     <bind xmlns='urn:xmpp:component:0'>
       <hostname>chat.example.com</hostname>
     </bind>
```

```
  </iq>
```

If the hostname can be bound, the server MUST return an IQ-result specifying the exact hostname that was bound.

Listing 3: Bind Result

```
S: <iq id='bind_1' type='result'>
     <bind xmlns='urn:xmpp:component:0'>
       <hostname>chat.example.com</hostname>
     </bind>
   </iq>
```

If the hostname cannot be bound, the server MUST return an IQ-error, which SHOULD be <bad-request/>, <conflict/>, <not-allowed/>, or <resource-constraint/>, just as with client resource binding as specified in RFC 3920.
Note: Although the JID asserted during STARTTLS and SASL negotiation MUST be of the form <domain> (i.e., an XMPP domain identifier), the <hostname/> element MAY be of the form <domain/resource>. This form can be used for application-specific functionality (e.g., load balancing), but such functionality is out of scope for this specification.
A component can send a subsequent bind request to bind another hostname (a server MUST support binding of multiple hostnames).

Listing 4: Another Bind Request

```
C: <iq id='bind_2' type='set'>
     <bind xmlns='urn:xmpp:component:0'>
       <hostname>foo.example.com</hostname>
     </bind>
   </iq>
```

If the server cannot process the bind request (e.g., because the component has already bound the desired hostname), the server MUST return an IQ-error (e.g., <conflict/>).
A component can also unbind a resource that has already been bound (a server MUST support unbinding).

Listing 5: Unbind Request

```
C: <iq id='unbind_1' type='set'>
     <unbind xmlns='urn:xmpp:component:0'>
       <hostname>foo.example.com</hostname>
     </unbind>
   </iq>
```

If the hostname can be unbound, the server MUST return an IQ-result.

Listing 6: Unbind Result

```
S: <iq id='unbind_1' type='result'/>
```

5 Security Considerations

This protocol improves upon the earlier component protocol defined in XEP-0114 by specifying the use of Transport Layer Security (TLS) for channel encryption and the Simple Authentication and Security Layer (SASL) for authentication. Because this protocol re-uses the XML stream establishment processes defined in XMPP Core, the security considerations from RFC 3920 and RFC 6120 apply to this protocol as well.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [5].

7 XMPP Registrar Considerations

7.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:component:0

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [6] shall add the foregoing namespace to the registry located at <http://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [7].

7.2 Protocol Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version

[5] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[6] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <http://xmpp.org/registrar/>.

[7] XEP-0053: XMPP Registrar Function <http://xmpp.org/extensions/xep-0053.html>.

number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

8 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:component:0'
    xmlns='urn:xmpp:component:0'
    elementFormDefault='qualified'>

  <xs:element name='bind'>
    <xs:complexType>
      <xs:sequence>
        <xs:choice minOccurs='0' maxOccurs='1'>
          <xs:element name='hostname' type='xs:string'/>
        </xs:choice>
        <xs:element name='required'
                    minOccurs='0'
                    maxOccurs='1'
                    type='empty'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='unbind'>
    <xs:complexType>
      <xs:sequence minOccurs='0'>
        <xs:element name='hostname' type='xs:string'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```