XEP-0174: Serverless Messaging

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

2008-11-26
Version 2.0

Status	Type	Short Name
Final	Standards Track	linklocal

This specification defines how to communicate over local or wide-area networks using the principles of zero-configuration networking for endpoint discovery and the syntax of XML streams and XMPP messaging for real-time communication. This method uses DNS-based Service Discovery and Multicast DNS to discover entities that support the protocol, including their IP addresses and preferred ports. Any two entities can then negotiate a serverless connection using XML streams in order to exchange XMPP message and IQ stanzas.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

1.1 Motivation

The Extensible Messaging and Presence Protocol (XMPP) as defined in XMPP Core [1] does not support direct client-to-client interactions, since it requires authentication with a server: an XMPP client is allowed access to the network only after it has authenticated with a server, and the server will not grant access if authentication fails for any reason. If an unauthenticated client attempts to communicate directly with another client, such communication will fail because all XMPP communications are sent through one or more servers and a client cannot inject messages onto the network unless it first authenticates with a server.
However, it is possible to establish an XMPP-like communication system on a local (or even wide-area) network using the principles of zero-configuration networking. In this situation, the clients obviate the XMPP requirement for authentication with a server by relying on zero-configuration networking to establish serverless communication using the _presence._tcp DNS SRV service type. Once discovery has been completed, the clients are able to negotiate an XML stream between themselves and then exchange messages and other structured data using the XMPP <message/> and <iq/> stanzas.
Serverless messaging is typically restricted to a local network (or ad-hoc wide-area network) because of how zero-configuration networking works. It is impossible for clients that communicate via this serverless mode to insert messages into an XMPP network, which is why this kind of ”mesh” is most accurately referred to as an XMPP-like system that exists outside the context of existing XMPP networks (though see the Security Considerations regarding the ability to ”forward” messages from a serverless mesh to an XMPP network or vice-versa).
Such a ”mesh” can be quite valuable in certain circumstances. For instance, participants in a trade show or conference, users of the same wifi hotspot, or employees on the same local area network can communicate without the need for a pre-configured server. For this reason, support for serverless messaging has been a feature of Apple’s iChat client when operating in Bonjour (formerly Rendezvous) mode since 2002. Because it is desirable for other Jabber/XMPP clients to support such functionality, this document describes how to use zero-configuration networking as the basis for serverless communication, mainly for use on local links (although the protocol can also be used on ad-hoc wide-area networks).

1.2 How It Works

This section provides a friendly introduction to serverless messaging. The examples show usage on a local link using dynamically configured link-local addresses as described in RFC 3927 [2] (see the Wide-Area Networks section of this document regarding non-local usage).
Imagine that you are a Shakespearean character named Juliet. You are are using your laptop computer (a machine named ”pronto”) at a wifi hotspot in downtown Verona and you want to find other people to chat with on an ad-hoc basis (i.e., not people in your normal XMPP

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses <http://tools.ietf.org/html/rfc3927>.

roster). Therefore your chat client advertises a serverless address of "juliet@pronto" so that other people can dynamically find you at the hotspot. Your client does this by invoking a daemon on your machine that supports DNS-based Service Discovery ("DNS-SD") as defined in RFC 6763 [3] and Multicast DNS ("mDNS") as defined in RFC 6762 [4]. As a result, the daemon (1) publishes the following DNS records to the multicast DNS address 224.0.0.251 (or FF02::FB for IPv6) and (2) listens for multicast DNS queries requesting these records:

```
pronto.local. A 10.2.1.187

juliet@pronto._presence._tcp.local. SRV 5562 pronto.local.

_presence._tcp.local. PTR juliet@pronto._presence._tcp.local.
```

The meaning of these records is as follows:

- The A record specifies the IP address 10.2.1.187 at which the "pronto" machine will listen for connections.
- The SRV record (see RFC 2782 [5]) maps the presence service instance "juliet@pronto" to the machine "pronto.local." on port 5562.
- The PTR ("pointer") record (see RFC 2317 [6] and RFC 1886 [7]) says that there is a service of type "presence" on the local subnet (".local.") called "juliet@pronto" and that the service communicates over TCP.

Your chat client also wants to advertise some information about you (subject to your control so that you don't divulge private information). Therefore it invokes the mDNS daemon to also publish a single DNS TXT record (see RFC 1464 [8]) that encapsulates some strings of information, where the record name is the same as the SRV record and the record value follows the format described in the TXT Record section of this document. The strings are typically key-value pairs such as the following:

```
txtvers=1
1st=Juliet
email=juliet@capulet.lit
hash=sha-1
jid=juliet@capulet.lit
last=Capulet
msg=Hanging out downtown
```

[3] RFC 6763: DNS-Based Service Discovery <http://tools.ietf.org/html/rfc6763>.
[4] RFC 6762: Multicast DNS <http://tools.ietf.org/html/rfc6762>.
[5] RFC 2782: A DNS RR for specifying the location of services (DNS SRV) <http://tools.ietf.org/html/rfc2782>.
[6] RFC 2317: Classless IN-ADDR.ARPA delegation <http://tools.ietf.org/html/rfc2317>.
[7] RFC 1886: DNS Extensions to support IP version 6 <http://tools.ietf.org/html/rfc1886>.
[8] RFC 1464: Using the Domain Name System To Store Arbitrary String Attributes <http://tools.ietf.org/html/rfc1464>.

```
nick=JuliC
node=http://www.adiumx.com
phsh=a3839614e1a382bcfebbcf20464f519e81770813
port.p2pj=5562
status=avail
vc=CA!
ver=QgayPKawpkPSDYmwT/WM94uAlu0=
```

Other people at the hotspot can also advertise similar DNS records for use on the local link. Essentially, the mDNS daemons running on all of the machines at the hotspot collectively manage the ".local." domain, which has meaning only at the hotspot (not across the broader Internet). Queries and responses for services on the local link occur via multicast DNS over UDP port 5353 instead of via normal DNS unicast over UDP port 53. When a new machine joins the local link, it can send out queries for any number of service types, to which the other machines will reply. For the purpose of serverless messaging we are interested only in the "presence" service, but many other services could exist on the local link (see dns-sd.org for a complete list).

Now let us imagine that a fine young gentleman named Romeo joins the hotspot and that his chat client (actually his mDNS daemon) sends out multicast DNS queries for services of type "presence". To do this, his client essentially reverses the order of DNS record publication (explained above) by asking for pointers to presence services (i.e., PTR records that match "_presence._tcp.local."), querying each service for its service instance and port (i.e., SRV record), mapping each service instance to an IP address (i.e., A record), and finding out additional information about the entity using the service (i.e., TXT record parameters). [9] As a result, Romeo's client will discover any number of local presence services, among them a service named "juliet@pronto" (with some intriguing TXT record parameters) at IP address 10.2.1.187 and port 5562. Being a romantic fellow, he then initiates a chat with you by opening an XML stream to the advertised IP address and port.

```
<?xml version='1.0'?>
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        from='romeo@forza'
        to='juliet@pronto'
        version='1.0'>
```

Your client then responds with a response stream header.

```
<?xml version='1.0'?>
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        from='juliet@pronto'
```

[9] As explained in the DNS-SD specification, these queries might all be returned in the same answer.

```
        to='romeo@forza'
        version='1.0'>
```

Romeo then sends you an XMPP message.

```
<message from='romeo@forza' to='juliet@pronto'>
  <body>M'lady, I would be pleased to make your acquaintance.</body>
</message>
```

And you reply.

```
<message from='juliet@pronto' to='romeo@forza'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

You chat with Romeo for a while, then your client closes the stream.

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

And Romeo's client does the same.

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

Finally you decide to head home, so your mDNS daemon sends a Multicast DNS "Goodbye" packet for your PTR record. As a result, everyone else at the hotspot receives a Multicast DNS "Remove" event, at which point they cancel any outstanding A, SRV, TXT, or NULL record queries related to your presence service.

2 Glossary

Term	Description
Bonjour	Apple Computer's implementation of zero-configuration networking, formerly known as Rendezvous. See <http://www.apple.com/macosx/features/bonjour/>.
DNS-SD	A convention for naming and structuring DNS SRV records such that a client can dynamically discover a domain for a service using only standard DNS queries. See draft-cheshire-dnsext-dns-sd. For a full list of registered DNS-SD records, see <http://www.dns-sd.org/ServiceTypes.html>.
Multicast DNS (mDNS)	A technology that provides the ability to perform DNS-like operations on a local link in the absence of any conventional unicast DNS server. See draft-cheshire-dnsext-multicastdns.

Term	Description
Zero-configuration networking	A set of technologies that enable the use of the Internet Protocol for local or wide-area communications. See <http://www.zeroconf.org/>.

3 DNS Records

In order to advertise its availability for serverless messaging, a client MUST publish four different kinds of DNS records:

1. A PTR record of the following form:

```
_presence._tcp.local. PTR user@machine._presence._tcp.local.
```

2. An address (”A” or ”AAAA”) record of the following form (where the IP address can be either an IPv4 address or an IPv6 address):

```
machine.local. A ip-address
```

3. An SRV record of the following form:

```
user@machine._presence._tcp.local <ttl> SRV <priority> <weight>
    port-number machine.local.
```

4. A TXT record whose name is the same as the SRV record and whose value follows the format described in the TXT Record section of this document, consisting of a set of strings that typically represent a series of key-value pairs such as the following:

```
txtvers=1
1st=user-first-name
email=user-email-address
hash=entity-capabilities-algorithm
jid=user-jabber-id
last=user-last-name
msg=freeform-availability-status
n=entity-capabilities-application-name
nick=user-nickname
node=application-identifier
n=entity-capabilities-operating-system
phsh=sha1-hash-of-avatar
```

```
port.p2pj=5562
status=avail-away-or-dnd
vc=capabilities-string
ver=entity-capabilities-identity
```

Note: The DNS-SD specification stipulates that the TXT record MUST be published, but that it MAY contain no more than a single zero byte (e.g., if the user does not wish to publish any personal information).

The ”machine” is the name of the computer, the ”user” is the system username of the principal currently logged into the computer, the ”port” can be any unassigned port number, and the ”ip-address” is the physical address of the computer on the local network.
So, for example, if the machine name is ”pronto”, the username is ”juliet”, the chosen port is ”5562”, the IP address is ”10.2.1.187”, and the personal information is that plausibly associated with a certain Shakespearean character, the DNS records would be as follows:

```
_presence._tcp.local. PTR juliet@pronto._presence._tcp.local.

juliet@pronto._presence._tcp.local. SRV 5562 pronto.local.

pronto.local. A 10.2.1.187

juliet@pronto._presence._tcp.local. IN TXT
  ”txtvers=1”
  ”1st=Juliet”
  ”email=juliet@capulet.lit␣”
  ”hash=sha-1”
  ”jid=juliet@capulet.lit”
  ”last=Capulet”
  ”msg=Hanging␣out␣downtown”
  ”nick=JuliC”
  ”node=http://www.adiumx.com”
  ”phsh=a3839614e1a382bcfebbcf20464f519e81770813”
  ”port.p2pj=5562”
  ”status=avail”
  ”vc=CA!”
  ”ver=QgayPKawpkPSDYmwT/WM94uAlu0=””
```

The IPv4 and IPv6 addresses associated with a machine might vary depending on the local network to which the machine is connected. For example, on an Ethernet connection the physical address might be ”192.168.0.100” but when the machine is connected to a wireless network the physical address might change to ”10.10.1.187”. See RFC 3927 for details.
If the machine name asserted by a client is already taken by another machine on the network, the client MUST assert a different machine name, which SHOULD be formed by adding the character ”-” and digit ”1” to the end of the machine name string (e.g., ”pronto-1”), adding the character ”-” and digit ”2” if the resulting machine name is already taken (e.g., ”pronto-2”),

and similarly incrementing the digit until a unique machine name is constructed.
If the username asserted by a client is already taken by another application on the machine, the client MUST assert a different username, which SHOULD be formed by adding the character "-" and digit "1" to the end of the username string (e.g., "juliet-1"), adding the character "-" and digit "2" if the resulting username is already taken (e.g., "juliet-2"), and similarly incrementing the digit until a unique username is constructed.

3.1 TXT Record

DNS-SD enables service definitions to include a TXT record that specifies parameters to be used in the context of the relevant service type. The name of the TXT record is the same as that of the SRV record (i.e., "user@machine._presence._tcp.local."). The value of the TXT record is one or more strings, where each string is a parameter that usually takes the form of a key-value pair.
In the context of serverless messaging, the following rules apply:

1. The entire TXT record needs to comply with the suggested maximum TXT record size (see Section 6.3 of the DNS-SD specification).
2. A given key MUST NOT occur more than once in a given TXT record value (see Section 6.4 of the DNS-SD specification).
3. The first parameter in the TXT record value SHOULD be "txtvers" (see Section 6.7 of the DNS-SD specification).

The XMPP Registrar [10] maintains a registry of the parameters that can be used in the TXT record value for the _presence._tcp service type, as specified in the XMPP Registrar Considerations section of this document. Those parameters are not listed here. It is OPTIONAL to include any of these TXT record parameters, and an implementation MUST NOT fail (i.e., MUST enable serverless messaging) even if none of the parameters are provided by another entity. However, as mentioned the TXT record MUST be published (although its value MAY be a single zero byte).
Most of the registered TXT record parameters relate to human users, in which context certain parameters are of greater interest than others, e.g. "msg", "nick", and "status"; however, serverless messaging can be used by non-human entities (e.g., devices).
Note: See the Security Considerations section of this document regarding the inclusion of information that can have an impact on personal privacy (e.g., the "1st", "last", "nick", "email", and "jid" parameters).

[10] 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 <https://xmpp.org/registrar/>.

4 Discovering Other Users

In order to discover other users, a client sends an mDNS request for PTR records that match "_presence._tcp.local.". The client then receives replies from all machines that advertise support for serverless messaging. [11] In accordance with Section 13 of the DNS-SD specification, these replies can include the SRV, A/AAAA, and TXT records in the Additional Section of the DNS message (subject to the size limits described in Section 19 of the Multicast DNS specification).
The client MAY then find out detailed information about each machine by sending SRV and TXT queries [12] to "user@machine.local." for each machine; however, to preserve bandwidth, the client SHOULD NOT send these queries unless it is about to initiate communication with the other user, and it MUST cancel the queries after it has received a response).

5 Exchanging Presence

When the _presence._tcp service is used, presence is exchanged via the format described in the TXT Record section of this document. In particular, presence information is not pushed as in XMPP (see RFC 3921 [13]). Instead, clients listen for presence announcements from other entities on the local link or wide-area network. Recommended rates for sending updates can be found in the Multicast DNS specification.

6 Initiating an XML Stream

In order to exchange serverless messages, the initiator and recipient MUST first establish XML streams between themselves, as is familiar from RFC 6120.
First, the initiator opens a TCP connection at the IP address and port discovered via the DNS lookup for an entity and opens an XML stream to the recipient, which SHOULD include 'to' and 'from' address:

Listing 1: Initiator Opens a Stream

```
I: <?xml version='1.0'?>
   <stream:stream
            xmlns='jabber:client'
            xmlns:stream='http://etherx.jabber.org/streams'
            from='romeo@forza'
            to='juliet@pronto'
            version='1.0'>
```

[11] The replies will include a record corresponding to the client itself; the client MUST filter out this result.
[12] These questions MAY all be sent in one DNS query packet.
[13] RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc3921>.

Note: If the initiator supports stream features and the other stream-related aspects of XMPP 1.0 as specified in RFC 6120, then it SHOULD include the version='1.0' flag as shown in the previous example.
The recipient then responds with a stream header as well:

Listing 2: Recipient Sends Stream Header Response

```
R: <?xml version='1.0'?>
   <stream:stream
            xmlns='jabber:client'
            xmlns:stream='http://etherx.jabber.org/streams'
            from='juliet@pronto'
            to='romeo@forza'
            version='1.0'>
```

If both the initiator and recipient included the version='1.0' flag, the recipient SHOULD also send stream features as specified in RFC 6120:

Listing 3: Recipient Sends Stream Features

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

The exchange of stream headers results in an unencrypted and unauthenticated channel between the two entities. See the Security Considerations section of this document regarding methods for authenticating and encrypting the stream.

7 Exchanging Stanzas

Once the streams are established, either entity then can send XMPP message or IQ stanzas by specifying 'to' and 'from' addresses using the logical addresses: [14]

Listing 4: Sending a Message

```
<message from='romeo@forza' to='juliet@pronto'>
  <body>M'lady, I would be pleased to make your acquaintance.</body>
</message>
```

[14] The to and from addresses MUST be of the form "user@machine" as discovered via SRV (this is the <Instance> portion of the Service Instance Name).

Listing 5: A Reply

```
<message from='juliet@pronto' to='romeo@forza'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

8 Ending an XML Stream

To end the chat, either party closes the XML stream:

Listing 6: Ending the Chat

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

The other party MUST then also close the stream in the other direction:

Listing 7: Closing the Stream

```
I: </stream:stream>
```

The closing party (i.e., the party that sent the first closing stream tag) then MUST close the TCP connection between them.
Note: The closing party might receive additional stanzas from the other party after sending its closing stream tag and before receiving a closing stream tag from the other party (e.g., because of network latency or because the other party has messages queued up for delivery when it receives the closing party's closing stream tag). Therefore, the closing party needs to be prepared to handle such messages, which it SHOULD do by presenting them to the controlling user (if any).

9 Going Offline

In order to go offline, a link-local entity MUST send a Multicast DNS "Goodbye" packet for the user's PTR record as described in Section 11.2 of the Multicast DNS specification. As a result, all other entities on the local network will receive a Multicast DNS "Remove" event, at which point they MUST cancel any outstanding TXT, SRV, or NULL record queries for the offline user.

10 Discovering Capabilities

Because serverless communication does not involve the exchange of XMPP presence, it is not possible to use Entity Capabilities (XEP-0115) [15] for capabilities discovery. Therefore, it

[15] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

is RECOMMENDED to instead include the node, hash, and ver TXT record parameters (and OPTIONAL to include the ext parameter). The values of these parameters MUST be the same as the values for the 'node', 'hash', 'ver', and 'ext' attributes that are advertised for the application in normal XMPP presence (if any) via the Entity Capabilities protocol as described in XEP-0115.
As with Entity Capabilities over native XMPP networks, a client might not know the Service Discovery (XEP-0030) [16] features associated with the 'ver' value advertised by another entity. However, in the case of serverless messaging there is no way for the client to discover the entity's supported features without initiating an XML stream to that entity and then sending a Service Discovery information ("disco#info") request over the negotiated stream.
Unfortunately, full stream negotiation (including TLS and SASL if appropriate) can require a large number of packets. Therefore, as an optimization, it is RECOMMENDED for the receiving entity in a serverless XML stream negotiation to include its disco#info data (including node) as a stream feature, as shown in the following examples.

Listing 8: Initiator Opens a Stream

```
I: <?xml version='1.0'?>
   <stream:stream
            xmlns='jabber:client'
            xmlns:stream='http://etherx.jabber.org/streams'
            from='romeo@forza'
            to='juliet@pronto'
            version='1.0'>
```

Listing 9: Recipient Sends Stream Header Response

```
R: <?xml version='1.0'?>
   <stream:stream
            xmlns='jabber:client'
            xmlns:stream='http://etherx.jabber.org/streams'
            from='juliet@pronto'
            to='romeo@forza'
            version='1.0'>
```

Listing 10: Recipient Sends Stream Features

```
R: <stream:features>
     <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
       <optional/>
     </starttls>
     <query xmlns='http://jabber.org/protocol/disco#info'
            node='http://code.google.com/p/exodus#QgayPKawpkPSDYmwT/
                WM94uAlu0='>
       <identity category='client' name='Exodus 0.9.1' type='pc'/>
       <feature var='http://jabber.org/protocol/caps'/>
```

[16] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
      <feature var='http://jabber.org/protocol/disco#info'/>
      <feature var='http://jabber.org/protocol/disco#items'/>
      <feature var='http://jabber.org/protocol/muc'/>
    </query>
  </stream:features>
```

If the initiating entity was connecting to the receiving entity only to perform a Service Discovery query, it SHOULD then end the stream:

Listing 11: Initiating Entity Terminates XML Stream

```
I: </stream:stream>
```

Listing 12: Receiving Entity Mirrors Stream Termination

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

11 Implementation Notes

11.1 Multiple Network Interfaces

Devices that use serverless messaging can have multiple network interfaces. As a result, it is possible to discover the same entity multiple times. Even if a client discovers the same presence name on multiple network interfaces, it MUST show only one entity in the serverless roster. In addition, because local IP addresses can be dynamically re-assigned, the client SHOULD NOT store the IP address to be used for communication when it discovers that address in the initial DNS lookup phase; instead, it SHOULD delay sending the Multicast DNS query until the client is ready to communicate with the other entity.

11.2 Buddy Icons

If an entity has an associated icon (e.g., a user avatar or photo), its client SHOULD publish the raw binary data for that image via a DNS NULL record of the following form:

```
_presence._tcp.local. IN NULL raw-binary-data-here
```

Note: In accordance with RFC 1035 [17], the data MUST be 65535 octets or less.
After retrieving the "phsh" value from a Buddy's TXT record, a client SHOULD search its local picture database to learn the last recorded picture hash value for an entity and then compare it to the "phsh" value in the TXT record. If the values are equal, the client SHOULD use the local copy of the icon. If the picture hash values are not equal, the client SHOULD issue a Multicast DNS NULL record query to retrieve the new icon. After retrieving the NULL record,

[17] RFC 1035: Domain Names - Implementation and Specification <http://tools.ietf.org/html/rfc1035>.

the client SHOULD replace the old ”phsh” value in the picture database with the new ”phsh” value and save the icon to disk. If the client needs to send a Multicast DNS query in order to retrieve the icon, it MUST cancel the NULL record query immediately after receiving a response containing the new picture data.
If a user changes their picture, the user’s client MUST update the NULL record with the contents of the new picture, calculate a new picture hash, and then update the ”phsh” value in the TXT record with the new hash value. Since all users ”logged into” serverless presence are monitoring for TXT record changes, they will see that the ”phsh” value was changed; if they wish to view the new icon, their clients SHOULD issue a new Multicast DNS query to retrieve the updated picture.

11.3 Port

The port used for serverless messaging MAY be any unassigned port number, as determined by the messaging application on the device. The chosen port MUST be specified in the SRV record and applications MUST use the port specified in the SRV record. However, the chosen port SHOULD also be specifed in the ”port.p2pj” TXT record for backwards-compatibility with older implementations, and if included the port specified in the TXT record MUST be the same as the port specified in the SRV record.

11.4 Wide-Area Networks

Serverless messaging via the _presence._tcp DNS SRV service type is not limited to local networks, since it is possible to advertise this service type via Wide-Area DNS-SD as described at <http://www.dns-sd.org/iChatWideArea.html>. Although the protocol is most commonly used on local networks, there is nothing intrinsic to the protocol that limits its use to peers on the same link, and it also works between any two peers that can discover each other via any profile of DNS-SD (whether local or wide-area). Naturally, the DNS records used in Wide-Area DNS-SD will not contain the ”.local.” domain, since the records are not intended for use over a local link.

11.5 User Interface

The presence name to be used for display in a serverless ”roster” SHOULD be obtained from the <Instance> portion of the received PTR record for each user; however, the client MAY instead display a name or nickname derived from the TXT record if available.
A client MAY require user approval before allowing a human user to chat with other users over serverless messaging.

12 Internationalization Considerations

RFC 1035 does not allow characters outside the US-ASCII [18] character range in DNS A records. Therefore the ”machine” portion of an A record as used for serverless messaging MUST NOT contain characters outside the US-ASCII character range.
Although RFC 2317 and RFC 2782 do not allow characters outside the US-ASCII character range in PTR and SRV records respectively, Section 4.1 of DNS-Based Service Discovery recommends support for UTF-8-encoded Unicode characters in the <Instance> portion of Service Instance Names, which in serverless messaging is the ”user@machine” portion of the PTR or SRV record. This document adheres to the recommendation in DNS-Based Service Discovery. However, as mentioned above, the ”machine” portion of the <Instance> portion MUST NOT contain characters outside the US-ASCII range.
Although RFC 1464 does not allow characters outside the US-ASCII character range in TXT records, Section 6.5 of DNS-Based Service Discovery mentions support for UTF-8-encoded Unicode characters in text record values (e.g., values of the TXT ”msg” name). This document adheres to the recommendation in DNS-Based Service Discovery.

13 Security Considerations

13.1 Authentication and Encryption

XMPP networks use TLS (RFC 5246 [19]) for channel encryption, SASL (RFC 4422 [20]) for authentication, and the Domain Name System (RFC 1034 [21]) for weak validation of server hostnames; these technologies help to ensure the identity of sending entities and to encrypt XML streams. By contrast, zero-configuration networking uses dynamic discovery and asserted machine names as the basis of sender identity. Therefore, serverless messaging does not result in authenticated identities in the same way that XMPP itself does, nor does it provide for an encrypted channel between entities.
To secure communications between serverless entities, it is RECOMMENDED to negotiate the use of TLS and SASL for the XML stream as described in RFC 6120. However, subject to client configuration and local service policies, an entity MAY accept an unauthenticated and unencrypted channel, in which case the client SHOULD warn the human user that the channel is unauthenticated and unencrypted.

[18] Coded Character Set - 7-bit American Standard Code for Information Interchange (American National Standards Institute X3.4, 1986).
[19] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[20] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.
[21] RFC 1034: Domain Names - Concepts and Facilities <http://tools.ietf.org/html/rfc1034>.

13.2 Stanza Injection

Because of fundamental differences between a true XMPP network and a serverless client "mesh", entities communicating via serverless messaging MUST NOT attempt to inject serverless traffic onto an XMPP network and an XMPP server MUST reject communications until an entity is properly authenticated in accordance with the rules defined in RFC 6120. However, a client on a serverless mesh MAY forward traffic to an XMPP network after having properly authenticated on such a network (e.g., to forward a message received on a serverless client mesh to a contact on an XMPP network).

13.3 TXT Record Parameters

Because there is no mechanism for validating the information that is published in DNS TXT records, it is possible for clients to "poison" this information (e.g., by publishing email addresses or Jabber IDs that are controlled by or associated with other users).

13.4 Private Information

The TXT record parameters optionally advertised as part of this protocol MAY result in exposure of privacy-sensitive information about a human user (such as full name, email address, and Jabber ID). A client MUST allow a user to disable publication of this personal information (e.g., via client configuration).

14 IANA Considerations

DNS-SD service type names are not yet managed by the Internet Assigned Numbers Authority (IANA) [22]. Section 19 of DNS-Based Service Discovery proposes an IANA allocation policy for unique application protocol or service type names. Until the proposal is adopted and in force, Section 19 points to <http://www.dns-sd.org/ServiceTypes.html> regarding registration of service type names for DNS-SD.
Before this specification was written, there was an existing registration for the "presence" service type, with registration information as follows:

1. Short name: presence
2. Long name: iChat AV
3. Responsible person: Jens Alfke <jens at apple.com>

[22] 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/>.

4. Defined TXT keys: txtvers, port.p2pj, phsh, vc, 1st, AIM, msg, status, last

On 2007-05-14, the XMPP Registrar submitted the following proposed modification to the existing registration, which was accepted on 2007-05-30:

1. Short name: presence
2. Long name: Link-Local Messaging
3. Responsible person: XMPP Registrar <registrar at xmpp.org>
4. Protocol URL: http://www.xmpp.org/extensions/xep-0174.html
5. Primary transport protocol: _tcp
6. TXT record URL: http://www.xmpp.org/registrar/linklocal.html

15 XMPP Registrar Considerations

15.1 Link-Local Messaging TXT Record Parameters Registry

The XMPP Registrar [23] maintains a registry of parameter strings contained in the TXT record advertised for serverless messaging (see <https://xmpp.org/registrar/linklocal.html>).

15.1.1 Registration Process

In order to submit new values to this registry, the registrant shall define an XML fragment of the following form and either include it in the relevant XMPP Extension Protocol or send it to the email address <registrar@xmpp.org>:

```
<param>
  <name>The name of the parameter as used a key-value pair.</name>
  <desc>A natural-language description of the parameter.</desc>
  <status>
    The requirements status of the record. Should be one of:
      - required
      - recommended
      - optional
      - deprecated
      - obsolete
 </status>
</param>
```

[23] 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 <https://xmpp.org/registrar/>.

The registrant can register more than one parameter at a time, each contained in a separate <record/> element.

15.1.2 Initial Registration

The following submission registers parameters in use as of June 2007. Refer to the registry itself for a complete and current list of parameters (this specification might or might not be revised when new parameters are registered).

```
<param>
  <name>1st</name>
  <desc>The given or first name of the user.</desc>
  <status>optional</status>
</param>

<param>
  <name>email</name>
  <desc>
    The email address of the user; can contain a space-separated list
    of more than one email address.
  </desc>
  <status>optional</status>
</param>

<param>
  <name>ext</name>
  <desc>
    A space-separated list of extensions; the value of this record
       MUST
    be the same as that provided via normal XMPP presence (if
       applicable)
    in the 'ext' attribute specified in Entity Capabilities (XEP-0115)
       .
  </desc>
  <status>optional</status>
</param>

<param>
  <name>hash</name>
  <desc>
    The hashing algorithm used to generated the 'ver' attribute in
    Entity Capabilities (XEP-0115) and therefore the ver parameter
    in Link-Local Messaging.
  </desc>
  <status>recommended</status>
</param>
```

```
<param>
  <name>jid</name>
  <desc>
    The Jabber ID of the user; can contain a space-separated list of
    more than one JID.
  </desc>
  <status>recommended</status>
</param>

<param>
  <name>last</name>
  <desc>The family or last name of the user.</desc>
  <status>optional</status>
</param>

<param>
  <name>msg</name>
  <desc>
    Natural-language text describing the user's state. This is
␣␣␣␣equivalent to the XMPP &lt;status/&gt;; element.
␣␣</desc>
␣␣<status>optional</status>
</param>

<param>
␣␣<name>nick</name>
␣␣<desc>A friendly or informal name for the user.</desc>
␣␣<status>recommended</status>
</param>

<param>
␣␣<name>node</name>
␣␣<desc>
␣␣␣␣A unique identifier for the application; the value of this record
    MUST
␣␣␣␣be the same as that provided via normal XMPP presence (if
    applicable)
␣␣␣␣in the 'node' attribute specified in Entity Capabilities (XEP
    -0115).
␣␣</desc>
␣␣<status>recommended</status>
</param>

<param>
␣␣<name>phsh</name>
␣␣<desc>
␣␣␣␣The SHA-1 hash of the user's avatar icon or photo. This SHOULD be
    requested using mDNS in unicast mode by sending a DNS query to the
    mDNS multicast address (224.0.0.251 or its IPv6 equivalent
```

```
        FF02::FB).
    The client SHOULD keep a local cache of icons keyed by hash. If
        the
    phsh value is not in the cache, the client SHOULD fetch the
        unknown
    icon and then cache it. Implementations SHOULD also include logic
        for
    expiring avatar icons.
  </desc>
  <status>optional</status>
</param>

<param>
  <name>port.p2pj</name>
  <desc>
    The port for serverless communication. This MUST be the same as
        the
    value provided for SRV lookups. Clients MUST use the port
        discovered
    via SRV lookups and MUST ignore the value of this parameter.
        However,
    clients SHOULD advertise this parameter if it is important to
        ensure
    backwards-compatibility with some existing implementations. (Note:
         In
    some existing implementations this value was hardcoded to "5298".)
  </desc>
  <status>deprecated</status>
</param>

<param>
  <name>status</name>
  <desc>
    The presence availability of the user. Allowable values are "avail
        ",
    "away", and "dnd", which map to mere XMPP presence (the user is
    available) and the XMPP &lt;show/&gt; values of "away" and "dnd",
    respectively; if the status record is not included, the status
        SHOULD
    be assumed to be "avail".
  </desc>
  <status>recommended</status>
</param>

<param>
  <name>txtvers</name>
  <desc>
    The version of the TXT record supported by the client. For
        backwards
```

```
    compatibility this is hardcoded at "1". This parameter SHOULD be
       the
    first one provided, in accordance with the DNS-SD specification.
  </desc>
  <status>deprecated</status>
</param>

<param>
  <name>vc</name>
  <desc>
    A flag advertising the user's␣ability␣to␣engage␣in␣audio␣or␣video
␣␣␣␣conferencing.␣If␣the␣user␣is␣able␣to␣engage␣in␣audio␣conferencing,
␣␣␣␣the␣string␣MUST␣include␣the␣"A"␣character.␣If␣the␣user␣is␣able␣to
␣␣␣␣engage␣in␣video␣conferencing,␣the␣string␣MUST␣include␣the␣"V"
␣␣␣␣character.␣If␣the␣user␣is␣able␣to␣engage␣in␣conferencing␣with␣more
␣␣␣␣than␣one␣participant,␣the␣string␣MUST␣include␣the␣"C"␣character.␣
    If
␣␣␣␣the␣user␣is␣not␣currently␣engaged␣in␣an␣audio␣or␣video␣conference,
␣␣␣␣the␣string␣MUST␣include␣the␣"!"␣character.␣The␣order␣of␣characters
␣␣␣␣in␣the␣string␣is␣immaterial.␣NOTE:␣This␣flag␣is␣included␣only␣for
␣␣␣␣backwards-compatibility;␣implementations␣SHOULD␣use␣the␣node,␣ver,
␣␣␣␣and␣ext␣parameters␣for␣more␣robust␣capabilities␣discovery␣as␣
    described
␣␣␣␣in␣the␣Discovering␣Capabilities␣section␣of␣XEP-0174.
␣␣</desc>
␣␣<status>optional</status>
</param>

<param>
␣␣<name>ver</name>
␣␣<desc>
␣␣␣␣A␣hashed␣string␣that␣defines␣the␣XMPP␣service␣discovery␣(XEP-0030)
␣␣␣␣identity␣of␣the␣application␣and␣the␣XMPP␣service␣discovery␣
    features
␣␣␣␣supported␣by␣the␣application;␣the␣value␣of␣this␣record␣MUST␣be␣the
␣␣␣␣same␣as␣that␣provided␣via␣normal␣XMPP␣presence␣(if␣applicable)␣in
␣␣␣␣the␣'ver'␣attribute␣specified␣in␣Entity␣Capabilities␣(XEP-0115).
␣␣</desc>
␣␣<status>recommended</status>
</param>
```

16 Acknowledgements

Thanks to Emanuele Aina, Jens Alfke, Marco Barisione, Stuart Cheshire, Justin Karneges, Marc Krochmal, Eric St. Onge, and Sjoerd Simons for their input. Some of the explanatory concepts

were loosely borrowed from SIP URI Service Discovery using DNS-SD [24].

[24] SIP URI Service Discovery using DNS-SD <http://tools.ietf.org/html/draft-lee-sip-dns-sd-uri>. Work in progress.