# XEP-0159: Spim-Blocking Control


Ian Paterson
mailto:ian.paterson@clientside.co.uk
xmpp:ian@zoofy.com

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/



2006-10-30
Version 0.3

| Status | Type | Short Name |
| --- | --- | --- |
| Deferred | Standards Track | TO BE ASSIGNED |



This document defines an XMPP protocol extension that enables clients to control how their servers may block spim that is addressed to them. It specifies a default privacy list fall-through action.

# Legal

## Copyright



## Permissions



## Warranty



## Liability



## Conformance

# Contents

# 1 Introduction

## 1.1 Motivation

The appearance of large public IM services based on XMPP Core [1] and XMPP IM [2] makes it desirable to implement protocols that *discourage* the sending of large quantities of instant messaging spam (a.k.a. "spim"). Spim could be generated by XMPP clients connected to legitimate servers or by XMPP servers with virtual clients, where the malicious entities are hosted on networks of "zombie" machines. Spim is defined here as any type of unsolicited XMPP stanza sent by a "robot" and delivered to a human, including messages and subscription requests. Spim has the potential to disrupt people even more than spam, because each message interrupts the receiver (humans typically filter SPAM in batch mode).
Spim blocking is more efficiently performed on the receiving server for several reasons:

- The sending server may be controlled by the spimmer.
- Client implementations are simplified.
- Client-to-server bandwidth is saved.
- Consistency with the stanza blocking protocol specified in Privacy Lists (XEP-0016) [3].
- The use of interactive spim recognition techniques (like CAPTCHA Forms (XEP-0158) [4]) does not leak information about the destination client's presence.
- If the destination client is not online when the stanza is sent, then all spim would need to be stored by the server until the user comes back online to decide if it is spim.
- Furthermore, if the sending client is no longer online when the stanza is received, then it would not be possible for the receiving client to use interactive spim recognition techniques.

However, no automated spim recognition techniques work perfectly all the time. This document is designed to give users control over the spim recognition their servers perform on their behalf.

## 1.2 Requirements

Clients should be able to:

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[3] XEP-0016: Privacy Lists <https://xmpp.org/extensions/xep-0016.html>.
[4] XEP-0158: CAPTCHA Forms <https://xmpp.org/extensions/xep-0158.html>.

- disable spim recognition for senders that have been falsely identified as spimmers
- stop spim from senders that spim recognition is failing to block
- exempt from spim recognition stanzas sent by:
  - specific users
  - users of specific domains
  - members of specific roster groups
  - presence subscribers
- rely on the server to automatically exempt from spim recognition stanzas sent by other existing correspondents

## 1.3 Approach

The stanza blocking protocol defined in XEP-0016 enables a client to control *explicitly* which senders its server must block stanzas from. Such explicit blocking is suitable for privacy control, but not for filtering spim.
This document contradicts an *assumption* expressed in the standard blocking protocol in order to extend client control to spim blocking. More specifically, it simply defines a spim recognition privacy-list fall-through action that is different from the 'allow' default assumed in XEP-0016. [5]

## 1.4 Note on Spim Recognition

The various spim recognition procedures that may be employed by the server are beyond the scope of this document. No single measure can differentiate all spim perfectly. It is RECOMMENDED that servers implement a combination of complementary spim recognition techniques (and other anti-spim techniques [6]).
For example, a server could employ traffic and reputation analysis to filter the majority of spim, and use CAPTCHA Forms to identify the remainder (feeding what it learns back to the traffic and reputation analysis).

[5] The language used in XEP-0016 appears to be deliberately vague (informative rather than normative) in order to permit other default fall-through actions. The protocol defined in this document is therefore (arguably) compatible with XEP-0016.

[6] Other examples of anti-spim policies and protocols include: requiring a user to pass a robot challenge before registering a new account, invite-only and/or out-of-band user account registration, providing a standard protocol for reporting spim to both the servers involved, server-to-server connection dialback, karma (client-to-server and server-to-server), legal agreements not to send spim during user account registration, and IP blocking.

## 2 Discovery

A client MAY confirm that its server supports Spim-Blocking Control using Service Discovery (XEP-0030) [7].

Listing 1: Initial Service Discovery Information Request

```
<iq type='get'
    from='victim@mydomain.com/laptop'
    to='mydomain.com'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 2: Server Indicates Support

```
<iq type='result'
    from='mydomain.com'
    to='victim@mydomain.com/laptop'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='im' type='server'/>
    ...
    <feature var='http://www.xmpp.org/extensions/xep-0159.html#node'/>
    ...
  </query>
</iq>
```

## 3 Spim Blocking

This section specifies *server* functionality *not* defined in XEP-0016. This document will not reach Draft status until Server-Based Privacy Rules has been modified to permit this functionality.

### 3.1 Correspondents Lists

A server that supports this protocol MAY maintain a separate [8] list of each user's correspondents. Each list contains all the bare JIDs the user has either sent a stanza to or received a stanza from (over the past few weeks or months).
Note: When it blocks a stanza, the server MUST NOT add the 'from' attribute of the stanza to the correspondents list.
Note: The lists of correspondents have a very different function from rosters. Edits are initiated by the server not the client, allowing the lists to be completely transparent to clients.

[7] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[8] If the server were to maintain a single unified list of the correspondents of all its users, then spimmers would only need to pass a single spim recognition test before being allowed to send spim to *all* the server's users.

## 3.2 Fall-Through Procedure

The server SHOULD perform the following procedures whenever it receives a stanza that falls through the active privacy list of the user it is addressed to without being either allowed or denied:

1. If the server maintains correspondents lists, and if the bare JID in the 'from' attribute of the received stanza is on the addressed user's correspondents list, then the server SHOULD allow the stanza to be delivered to the user.
2. Otherwise, the server SHOULD perform one or more spim recognition procedures (not defined in this document).
3. Depending on the result of those procedures, it SHOULD then either allow, deny or delay the delivery of the stanza without informing the sender.

## 3.3 Delay Procedure

When a spim recognition procedure delays delivery of a stanza the server SHOULD store it temporarily.
While delivery is being delayed:

- The server SHOULD allow or deny delivery of the stanza immediately (and without informing the sender) when any subsequent changes to the user's privacy list, correspondents list or roster explicitly either allow or deny the delivery of the stanza. [9]
- The server MAY allow or deny delivery of the stanza at any time (for reasons not defined in this document).

Once delivery of a stanza has been delayed for an implementation-specific length of time, or an implementation-specific number of stanzas from the same sender (or same sending server) are being delayed, the server SHOULD deny delivery of the stanza without informing the sender.
A good example of a delayed spim recognition procedure is when servers use the CAPTCHA Forms protocol to confirm whether or not a client is a spim robot before denying or allowing the delivery of a stanza from a *new correspondent*. [10] [11]

[9] For example, the first time a stanza is delivered the correspondents list will change, typically triggering the immediate delivery of any other delayed stanzas.
[10] The very occasional inconvenience of responding to a challenge is small and perfectly acceptable -- especially when compared to the countless robot-generated interruptions people might otherwise have to filter every day.
[11] If a human user fails such a robot challenge then his client SHOULD give him the option to resend the stanza immediately.

# 4 Configuring Privacy Lists

This informative section requires no client or server functionality beyond that defined in XEP-0016.

## 4.1 Exempting Trusted Groups from Spim Blocking

### 4.1.1 Users on Roster

The client SHOULD use the 'subscription' type to exclude all JIDs on the user's roster from spim blocking (see the items with order 20, 30 and 40 in the example below).

### 4.1.2 Transports

At least in the medium term, clients that use non-XMPP protocols cannot be expected to support interactive spim recognition techniques (like CAPTCHA Forms). So, if its server uses interactive techniques, the client MAY use the 'jid' type to ensure its server does not block stanzas arriving from the transports the user has registered with (see the item with order 50 in the example below).

### 4.1.3 Users of Trusted Servers

If a user believes spim will not be sent by users of a particular server (e.g. the user's own corporate server), then the client MAY use the 'jid' type to exclude all these users from spim blocking (see the item with order 60 in the example below).

Listing 3: Exempting Users from Spim Blocking

```
<iq type='set' from='victim@mydomain.com/laptop'>
  <query xmlns='jabber:iq:privacy'>
    <list name='normal'>
      ...
      <item type='subscription'
            value='both'
            action='allow'
            order='20'/>
      <item type='subscription'
            value='to'
            action='allow'
            order='30'/>
      <item type='subscription'
            value='from'
            action='allow'
            order='40'/>
```

```
      ...
      <item type='jid'
            value='yahoo.transport.org'
            action='allow'
            order='50'/>
      ...
      <item type='jid'
            value='mydomain.com'
            action='allow'
            order='60'/>
      ...
    </list>
  </query>
</iq>
```

## 4.2 Exempting Individual Users from Spim Blocking

No spim recognition techniques are perfect. Senders are sometimes falsely identified as spim bots. (For example, when a server sends CAPTCHA Forms, but the client does not support that protocol.)
In these cases the user MAY ask out-of-band the person he is trying to communicate with to allow communications in one of the following ways:

- simply send him a message, so her server adds him to her correspondents list
- add his JID to her active privacy list with action='allow'
- add him to her roster (either in a group whose stanzas are allowed in her active privacy list, or with a subscription type that is allowed [12])

## 4.3 Explicitly Blocking Spim

If stanzas from a spim robot running on a zombie domain somehow manage to get past the server's spim recognition then the client MAY use the 'jid' type to block one or all JIDs from the domain. (XEP-0016 already enables this functionality.)

Listing 4: Blocking Spim from a Specific Domain

```
<iq type='set' from='victim@mydomain.com/laptop'>
  <query xmlns='jabber:iq:privacy'>
    <list name='normal'>
      ...
      <item type='jid'
            value='spimmer.com'
```

[12] The subscription type SHOULD never be 'none' because XEP-0016 specifies that, for the purposes of blocking, all JIDs not on the roster also have the subscription type 'none'.

```
            action='deny'
            order='70'/>
      ...
    </list>
  </query>
</iq>
```

### 4.4 Completely Disabling Spim Blocking

A client MAY disable server-side spim blocking by ensuring the default fall-through action is never applied. It does this simply by including an explicit fall-through item in its active privacy list.

Listing 5: Client Disables Spim Blocking

```
<iq type='set' from='victim@mydomain.com/laptop'>
  <query xmlns='jabber:iq:privacy'>
    <list name='normal'>
      ...
      <item action='allow' order='999'/>
    </list>
  </query>
</iq>
```

Note: Before a server with existing accounts deploys this protocol, it MAY ensure all users' privacy lists have an explicit fall-through item. As a result, spim recognition would be disabled until users choose to switch it on.

## 5 Security Considerations

If a server implements this protocol and its security is compromised, then the attacker may be able to access the list of all previous correspondants for every user. People are unable to delete their own lists and they may not even be aware that they exist. So servers MUST ensure they protect this sensitive information very carefully.

## 6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [13].

[13] 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/>.

# 7 XMPP Registrar Considerations

## 7.1 Well-Known Service Discovery Node

Until this specification advances to a status of Draft, its associated service discovery node shall be "http://www.xmpp.org/extensions/xep-00158.html#node"; upon advancement of this specification, the XMPP Registrar [14] shall issue a permanent identifier in accordance with the process defined in Section 4 of XMPP Registrar Function (XEP-0053) [15].

[14] 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/>.

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