



XMPP

XEP-0191: Simple Communications Blocking

Peter Saint-Andre
<mailto:stpeter@jabber.org>
<xmpp:stpeter@jabber.org>
<https://stpeter.im/>

2007-02-15
Version 1.1

Status	Type	Short Name
Draft	Standards Track	blocking

This document specifies an XMPP protocol extension for simple communications blocking.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 - 2011 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, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1	Introduction	1
2	Requirements	1
3	Relationship to Privacy Lists	1
4	JID Matching	2
5	Use Cases	3
5.1	User Discovers Support	3
5.2	User Retrieves Block List	3
5.3	User Blocks Contact	4
5.4	User Unblocks Contact	6
5.5	User Unblocks All Contacts	7
6	Implementation Notes	7
7	Security Considerations	8
8	IANA Considerations	8
9	XMPP Registrar Considerations	8
9.1	Protocol Namespaces	8
10	XML Schema	8
10.1	blocking	8
10.2	blocking:errors	10
11	Acknowledgements	10

1 Introduction

RFC 3921¹ includes an XMPP protocol extension for communications blocking, which has since been moved to Privacy Lists². Unfortunately, because the privacy lists extension is quite complex, it has not been widely implemented in servers and has been implemented virtually not at all in clients. This is problematic, since the ability to block communications with selected users is an important feature for an instant messaging system (and is required by RFC 2779³). However, the full power of privacy lists is not needed in order to block communications, so this document proposes a much simpler blocking protocol that meets the requirement specified in RFC 2779 and can be implemented much more easily in Jabber/XMPP clients and servers.

2 Requirements

The requirements for simple communications blocking are straightforward:

1. A user must be able to block communications with a specific contact.
2. A user should be able to determine which contacts are blocked.
3. A user should be able to unblock communications with a specific contact.

3 Relationship to Privacy Lists

The simple communications blocking protocol specified herein is intended to be a user-friendly "front end" to a subset of the functionality defined by the privacy lists protocol (XEP-0016). If a service deploys both privacy lists and simple communications blocking, the service MUST use the same back-end data store for both protocols. (Note: Wherever possible, this document attempts to define a protocol that is fully consistent with XEP-0016; if a particular aspect of functionality is not specified herein, the relevant text in XEP-0016 shall be taken to apply.)

A service SHOULD map the blocklist to the default privacy list, where each blocked JID is represented as a privacy list item of type "jid" and action "deny".⁴ If this is done and none of the user's clients ever use the privacy lists protocol, then the blocklist will always apply. This mapping has the following implications:

¹RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <<http://tools.ietf.org/html/rfc3921>>.

²XEP-0016: Privacy Lists <<http://xmpp.org/extensions/xep-0016.html>>.

³RFC 2779: A Model for Presence and Instant Messaging <<http://tools.ietf.org/html/rfc2779>>.

⁴An implementation MUST NOT block communications from one of a user's resources to another, even if the user happens to define a rule that would otherwise result in that behavior.

1. If all of a user's clients always use simple communications blocking, then the default privacy list will be equivalent to the blocklist and the default privacy list will be a kind of "virtual list" (in the sense that it is never modified directly by any of the clients).
2. If one of a user's clients uses privacy lists instead of blocklists and modifies the default privacy list by removing a blocked JID or blocking a new JID, then that change will be reflected in the blocklist.
3. If one of a user's clients uses privacy lists and does anything but block or unblock a JID, then that change will not be reflected in the blocklist (since blocklists cannot represent anything except blocked JIDs).
4. If one of a user's clients removes the default privacy list and substitutes a new list for the old list, the blocked JIDs in the new default privacy list (if any) will become the new blocklist.
5. If one of a user's clients makes active something other than the default privacy list, the user may receive communications from contacts who are blocked in the default list.

Because of the potential for confusion between block lists and privacy lists, it is NOT RECOMMENDED for a client to request both the block list and privacy lists in the same session. The priority of blocked (jid+deny) items in the privacy list SHOULD be such that they come first in the privacy list.

4 JID Matching

Matching of JIDs as specified in the 'jid' attribute of the <item/> element SHOULD proceed in the following order (this is consistent with XEP-0016):

1. <user@domain/resource> (only that resource matches)
2. <user@domain> (any resource matches)
3. <domain/resource> (only that resource matches)
4. <domain> (the domain itself matches, as does any user@domain or domain/resource)

5 Use Cases

5.1 User Discovers Support

In order for a client to discover whether its server supports the protocol defined herein, it MUST send a [Service Discovery](#)⁵ information request to the server:

Listing 1: Service discovery request

```
<iq from='juliet@capulet.com/chamber' to='capulet.com' type='get' id='
  disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info' />
</iq>
```

If the server supports the protocol defined herein, it MUST return a feature of "urn:xmpp:blocking":

Listing 2: Service discovery response

```
<iq from='capulet.com' to='juliet@capulet.com/chamber' type='result'
  id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:blocking' />
    ...
  </query>
</iq>
```

5.2 User Retrieves Block List

In order for a client to request a user's list of blocked contacts (e.g., in order to determine whether to unblock a contact), it shall send an IQ-get with no 'to' address (thus handled by the user's server) containing a <blocklist/> element qualified by the 'urn:xmpp:blocking' namespace:

Listing 3: Client requests blocklist

```
<iq type='get' id='blocklist1'>
  <blocklist xmlns='urn:xmpp:blocking' />
</iq>
```

If the user has any contacts in its blocklist, the server MUST return an IQ-result containing a <blocklist/> element that in turn contains one child <item/> element for each blocked contact:

⁵XEP-0030: Service Discovery <<http://xmpp.org/extensions/xep-0030.html>>.

Listing 4: Server returns blocklist with items

```
<iq type='result' id='blocklist1'>
  <blocklist xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
    <item jid='iago@shakespeare.lit' />
  </blocklist>
</iq>
```

If the user has no contacts in its blocklist, the server MUST return an IQ-result containing an empty <blocklist/> element:

Listing 5: Server returns empty blocklist

```
<iq type='result' id='blocklist1'>
  <blocklist xmlns='urn:xmpp:blocking' />
</iq>
```

A client SHOULD retrieve the block list after authenticating with its server and before completing any blocking or unblocking operations.

5.3 User Blocks Contact

In order for a user to block communications with a contact, the user's client shall send an IQ-set with no 'to' address (thus handled by the user's server) containing a <block/> element qualified by the 'urn:xmpp:blocking' namespace, where the JID to be blocked is encapsulated as the 'jid' attribute of the <item/> child element:

Listing 6: Block command

```
<iq from='juliet@capulet.com/chamber' type='set' id='block1'>
  <block xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
  </block>
</iq>
```

If the server can successfully process the block command, it MUST return an IQ-result:

Listing 7: Block command is successful

```
<iq type='result' id='block1' />
```

The server MUST also send an IQ-set to all of the user's resources that have requested the blocklist, containing the blocked item(s):

Listing 8: Block "push"

```
<iq to='juliet@capulet.com/chamber' type='set' id='push1'>
  <block xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
  </block>
</iq>

<iq to='juliet@capulet.com/balcony' type='set' id='push2'>
  <block xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
  </block>
</iq>
```

If the `<block/>` element does not contain at least one `<item/>` child element, the server **MUST** return a `<bad-request/>` error. The `<block/>` element **MAY** contain more than one `<item/>` child. Other standard XMPP stanza errors also apply; see [XMPP Core](#)⁶ and [Error Condition Mappings](#)⁷.

When the user blocks communications with the contact, the user's server **MUST** send unavailable presence information to the contact (but only if the contact is allowed to receive presence notifications from the user in accordance with the rules defined in RFC 3921).

Once the user has blocked communications with the contact, the user's server **MUST NOT** deliver any XML stanzas from the contact to the user. The block remains in force until the user subsequently unblocks communications with the contact (i.e., the duration of the block is potentially unlimited and applies across sessions).

If the contact attempts to send a stanza to the user (i.e., an inbound stanza from the user's perspective), the user's server shall handle the stanza according to the following rules:

- For presence stanzas (including notifications, subscriptions, and probes), the server **MUST NOT** respond and **MUST NOT** return an error.
- For message stanzas, the server **SHOULD** return an error, which **SHOULD** be `<service-unavailable/>`.
- For IQ stanzas of type "get" or "set", the server **MUST** return an error, which **SHOULD** be `<service-unavailable/>`. IQ stanzas of other types **MUST** be silently dropped by the server.

If the foregoing suggestions are followed, the user will appear offline to the contact.

If the user attempts to send an outbound stanza to the contact, the user's server **MUST NOT** route the stanza to the contact but instead **MUST** return a `<not-acceptable/>` error containing an application-specific error condition of `<blocked/>` qualified by the `'urn:xmpp:blocking:errors'` namespace:

⁶RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

⁷XEP-0086: Error Condition Mappings <http://xmpp.org/extensions/xep-0086.html>.

Listing 9: Error: contact is blocked

```

<message type='error' from='romeo@montague.net' to='juliet@capulet.com'
  >
  <body>Can you hear me now?</body>
  <error type='cancel'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <blocked xmlns='urn:xmpp:blocking:errors' />
  </error>
</message>

```

5.4 User Unblocks Contact

In order for a user to unblock communications with a contact, the user's client shall send an IQ-set with no 'to' address (thus handled by the user's server) containing an <unblock/> element qualified by the 'urn:xmpp:blocking' namespace, where the JID to be unblocked is encapsulated as the 'jid' attribute of the <item/> child element:

Listing 10: Unblock contact command

```

<iq type='set' id='unblock1'>
  <unblock xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
  </unblock>
</iq>

```

If the server can successfully process the unblock command, it MUST return an IQ-result:

Listing 11: Unblock contact command is successful

```

<iq type='result' id='unblock1' />

```

The server MUST also send an IQ-set to all of the user's resources that have requested the blocklist, containing the unblocked item(s):

Listing 12: Unblock "push"

```

<iq to='juliet@capulet.com/chamber' type='set' id='push3'>
  <unblock xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
  </unblock>
</iq>

<iq to='juliet@capulet.com/balcony' type='set' id='push4'>
  <unblock xmlns='urn:xmpp:blocking'>
    <item jid='romeo@montague.net' />
  </unblock>
</iq>

```

When the user unblocks communications with the contact, the user's server MUST send the user's current presence information to the contact (but only if the contact is allowed to receive presence notifications from the user in accordance with the rules defined in RFC 3921). After the user has unblocked communications with the contact, the user's server MUST deliver any subsequent XML stanzas from the contact to the user.

5.5 User Unblocks All Contacts

In order for a user to unblock communications with all contacts, the user's client shall send an IQ-set with no 'to' address (thus handled by the user's server) containing an empty <unblock/> element qualified by the 'urn:xmpp:blocking' namespace:

Listing 13: Unblock all command

```
<iq type='set' id='unblock2'>
  <unblock xmlns='urn:xmpp:blocking' />
</iq>
```

If the server can successfully process the unblock command, it MUST return an IQ-result:

Listing 14: Unblock all command is successful

```
<iq type='result' id='unblock2' />
```

The server MUST also send an IQ-set to all of the user's resources that have requested the blocklist, containing notification of global unblocking:

Listing 15: Unblock all "push"

```
<iq to='juliet@capulet.com/chamber' type='set' id='push5'>
  <unblock xmlns='urn:xmpp:blocking' />
</iq>

<iq to='juliet@capulet.com/balcony' type='set' id='push6'>
  <unblock xmlns='urn:xmpp:blocking' />
</iq>
```

Once the user has unblocked communications with all contacts, the user's server MUST deliver any XML stanzas from those contacts to the user.

6 Implementation Notes

When a server receives a block command from a user, it MAY cancel any existing presence subscriptions between the user and the blocked user and MAY send a message to the blocked

user; however, it is RECOMMENDED to deploy so-called "polite blocking" instead (i.e., to not cancel the presence subscriptions or send a notification). Which approach to follow is a matter of local service policy.

A service MAY also filter blocking users out of searches performed on user directories (see, for example, [Jabber Search](#)⁸); however, that functionality is out of scope for this specification.

7 Security Considerations

If properly implemented, this protocol extension does not introduce any new security concerns above and beyond those defined in RFC 3920 and RFC 3921.

8 IANA Considerations

No interaction with the [Internet Assigned Numbers Authority \(IANA\)](#)⁹ is required as a result of this specification.

9 XMPP Registrar Considerations

9.1 Protocol Namespaces

The [XMPP Registrar](#)¹⁰ includes 'urn:xmpp:blocking' and 'urn:xmpp:blocking:errors' in its registry of protocol namespaces (see <http://xmpp.org/registrar/namespaces.html>).

10 XML Schema

10.1 blocking

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

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:blocking'
  xmlns='urn:xmpp:blocking' />
```

⁸XEP-0055: Jabber Search <http://xmpp.org/extensions/xep-0055.html>.

⁹The 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/>.

¹⁰The 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/>.

```
    elementFormDefault='qualified'>

<xs:annotation>
  <xs:documentation>
    The protocol documented by this schema is defined in
    XEP-0191: http://www.xmpp.org/extensions/xep-0191.html
  </xs:documentation>
</xs:annotation>

<xs:element name='block'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='item' minOccurs='1' maxOccurs='unbounded' />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name='unblock'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='item' minOccurs='0' maxOccurs='unbounded' />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name='blocklist'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='item' minOccurs='0' maxOccurs='unbounded' />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name='item'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='empty'>
        <xs:attribute name='jid' type='xs:string' use='required' />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:simpleType name='empty'>
  <xs:restriction base='xs:string'>
    <xs:enumeration value='' />
  </xs:restriction>

```

```
</xs:simpleType>
</xs:schema>
```

10.2 blocking:errors

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:blocking:errors'
  xmlns='urn:xmpp:blocking:errors'
  elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0191: http://www.xmpp.org/extensions/xep-0191.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='blocked' type='empty' />

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value='' />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

11 Acknowledgements

Thanks to Valerie Mercier, Maciek Niedzielski, Kevin Smith, and Remko Tronçon for their feedback.