XEP-0062: Packet Filtering

Robert Norris
mailto:rob@cataclysm.cx
xmpp:rob@cataclysm.cx

2021-03-04
Version 0.2.2

Status	Type	Short Name
Deferred	Informational	Not yet assigned

A framework for packet filtering rules.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 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 <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction **1**
- 1.1 Definitions 1

2 Structure **2**

3 Filter modules **3**

4 Protocol **4**
- 4.1 Module discovery 4
- 4.2 Setting the ruleset 5
- 4.3 Retrieving the ruleset 6

5 Error Codes **7**

6 Implementation notes **8**

7 Security Considerations **8**

8 IANA Considerations **8**

9 JANA Considerations **8**

1 Introduction

Traditionally, the jabberd [1] server included an internal server module called ”mod_filter”, which provided a packet filtering facility for users. That service had the following problems:

- The service and protocol were undocumented, apart from some documentation reverse-engineered from the source code.
- The processing requirements that the service required made it unusable for large installations.
- Bugs in the service often caused the Jabber server to crash.

The most common use for this service was to provide server-side blocklists. Unforuntately, mod_filter was overpowered even by this relatively simple form of packet filtering (matching the sending JID and dropping the packet if necessary), so this need has been neatly filled by Privacy Lists (XEP-0016) [2].
However, packet filtering (as opposed to the subset of JID blocking) is still of use, for the following tasks:

- Setting up automatic responses to messages (e.g., ”vacation” messages).
- Redirecting packets to another JID.
- Modifying the contents of a packet in-transit (e.g., language translation, adding <x/> data).
- Force incoming messages to be stored offline (e.g., for low-bandwidth clients).

This document proposes a modular, extensible framework for specifying packet filtering rules. This document does not, however, propose any specific filter conditions or actions - that is left to other proposals.
The framework itself operates in the ”http://jabber.org/protocol/filter” namespace.

1.1 Definitions

The following definitions are used throughout this document:

- ruleset - a set of filtering rules.
- rule - a set of conditions with an associated action.

[1] The jabberd server is the original server implementation of the Jabber/XMPP protocols, first developed by Jeremie Miller, inventor of Jabber. For further information, see <http://jabberd.org/>.
[2] XEP-0016: Privacy Lists <https://xmpp.org/extensions/xep-0016.html>.

- condition - an expression (or set of expressions) that, when applied to a packet, is either true or false.
- action - a task that may be performed on a packet.

2 Structure

A single rule is be expressed in XML like so:

Listing 1: XML representation of a rule

```
<rule description='natural-language␣description␣of␣rule'>
  <condition>[conditionexpr]</condition>
  <action>[actionspec]</action>
</rule>
```

A rule is processed by applying its condition to the packet. If the condition is true, then the action is taken. The "description" attribute is provided so a rule generator can assign a meaningful and user-readable description of a rule.
A ruleset is be expressed in XML like so:

Listing 2: XML representation of a ruleset

```
<ruleset>
  <rule description='rule␣description'>
    <condition>[conditionexpr]</condition>
    <action>[actionspec]</action>
  </rule>
  <rule description='rule␣description'>
    <condition>[conditionexpr]</condition>
    <action>[actionspec]</action>
  </rule>
  <rule description='rule␣description'>
    <condition>[conditionexpr]</condition>
    <action>[actionspec]</action>
  </rule>
</ruleset>
```

A ruleset is processed by applying each rule to the packet, one at a time. Processing of the ruleset stops after the first matching rule is found and its action taken, *unless* the "continue" attribute is found on the matched rule, in which case the remaining rules get processed as though the current rule did not match. If no rules match, packet processing continues as though no rules were specified.
If the <condition/> element contains no condition expression, then the rule matches all packets.
A ruleset does not have to contain any rules:

Listing 3: Empty ruleset

```
<ruleset/>
```

Conditions may be aggregated using AND or OR modifiers, like so:

Listing 4: Aggregated condition

```
<condition>
  <and>
    [conditionexpr1]
    [conditionexpr2]
    <or>
      [conditionexpr3]
      [conditionexpr4]
    </or>
  </and>
</condition>
```

The above example is equivalent to ”conditionexpr1 AND conditionexpr2 AND (conditionexpr3 OR conditionexpr4)”.
No such aggregation exists for actions - only a single action expression may be included within an <action/> element.

3 Filter modules

A filter module is a document that defines conditions and/or actions that can be use by this framework. Each module should have its own namespace, and should clearly define the effect of each condition and action it defines.
Consider a hypothetical module that defines conditions that match packets based on their header information. It might use the namespace ”http://jabber.org/protocol/filter/header” and might define the following conditions:

- <to/> - true when the CDATA of this element matches the ”to” attribute of the packet.
- <from/> - true when the CDATA of this element matches the ”from” attribute of the packet.
- <type/> - true when the CDATA of this element matches the ”type” attribute of the packet.

Equally, consider a hypothetical module that defines actions for redirecting messages. It might use the namespace ”http://jabber.org/protocol/filter/redirect” and might define the following conditions:

- <redirect/> - redirects the packet to the JID specified in the CDATA of this element.
- <copy/> - sends a copy of the packet to the JID specified in the CDATA of this element, while giving the original packet to the user.

These two modules might be combined to produce a ruleset like the following:

Listing 5: Using modules in a ruleset

```
<ruleset>
  <rule description='Send␣messages␣from␣my␣friend␣to␣my␣home␣account␣
      to␣be␣dealt␣with␣later'>
    <condition>
      <from xmlns='http://jabber.org/protocol/filter/header'>
          friend@theirisp.com</from>
    </condition>
    <action>
      <redirect xmlns='http://jabber.org/protocol/filter/redirect'>
          me@home.com</redirect>
    </action>
  </rule>
</ruleset>
```

Using modules in this way enables this framework to be easily extended to support new types of filtering, as well as enabling site administrators to select the types of functionallity that are best suited to their site.

4 Protocol

It will not always be appropriate for a service to provide a Jabber-based interface to its filter settings (e.g., in the case of an XML router, it will almost always be more appropriate to limit the specification of rules and rulesets to the router configuration). However, this will be appropriate sometimes (e.g., a session manager providing per-user packet filtering). In these cases, the following protocol should be used.

4.1 Module discovery

An entity may find out if a service supports filtering, and the modules it supports, by issuing a discovery request to the service:

Listing 6: Module discovery

```
<iq type='get' to='jabber.org' 'disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 7: Module discovery response

```
<iq type='result' to='jabber.org' id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='server' type='jabber' name='Jabber server'/>
    <feature var='http://jabber.org/protocol/filter'/>
    <feature var='http://jabber.org/protocol/filter/header'/>
    <feature var='http://jabber.org/protocol/filter/redirect'/>
    <feature var='...'/>
  </query>
</iq>
```

4.2 Setting the ruleset

An entity may set the filter ruleset for an entity (which may be itself) like so:

Listing 8: Setting the ruleset

```
<iq type='set' to='rob@cataclysm.cx' id='filter1'>
  <ruleset xmlns='http://jabber.org/protocol/filter'>
    <rule description='Send messages from my friend to my home account
       to be dealt with later'>
      <condition>
        <from xmlns='http://jabber.org/protocol/filter/header'>
            friend@theirisp.com</from>
      </condition>
      <action>
        <redirect xmlns='http://jabber.org/protocol/filter/redirect'>
            me@home.com</redirect>
      </action>
    </rule>
    <rule description='Copy messages from this spammer to our abuse
       address'>
      <condition>
        <from xmlns='http://jabber.org/protocol/filter/header'>
            spammer@badsite.com</from>
      </condition>
      <action>
        <copy xmlns='http://jabber.org/protocol/filter/redirect'>
            abuse@company.com</redirect>
      </action>
    </rule>
  </ruleset>
</iq>
```

On success, the service returns:

Listing 9: Setting the ruleset result

```
<iq type='result' from='rob@cataclysm.cx' id='filter1'/>
```

On error, the server returns the original packet with an appropriate error:

Listing 10: Setting the ruleset failure

```
<iq type='error' to='rob@cataclysm.cx' id='filter1'>
  <ruleset xmlns='http://jabber.org/protocol/filter'>
    <rule description='Send␣messages␣from␣my␣friend␣to␣my␣home␣account
        ␣to␣be␣dealt␣with␣later'>
      <condition>
        <from xmlns='http://jabber.org/protocol/filter/header'>
            friend@theirisp.com</from>
      </condition>
      <action>
        <redirect xmlns='http://jabber.org/protocol/filter/redirect'>
            me@home.com</redirect>
      </action>
    </rule>
    <rule description='Copy␣messages␣from␣this␣spammer␣to␣our␣abuse␣
        address'>
      <condition>
        <from xmlns='http://jabber.org/protocol/filter/header'>
            spammer@badsite.com</from>
      </condition>
      <action>
        <copy xmlns='http://jabber.org/protocol/filter/redirect'>
            abuse@company.com</redirect>
      </action>
    </rule>
  </ruleset>
  <error code='403'>Forbidden</error>
</iq>
```

4.3 Retrieving the ruleset

An entity may retrieve the filter ruleset for an entity (which may be itself) like so:

Listing 11: Requesting the ruleset

```
<iq type='get' id='filter2'>
  <ruleset xmlns='http://jabber.org/protocol/filter'/>
</iq>
```

If the requesting entity has permissions to view the ruleset, the server must return the ruleset to the entity:

Listing 12: Retrieving the ruleset result

```
<iq type='error' to='rob@cataclysm.cx' id='filter2'>
  <ruleset xmlns='http://jabber.org/protocol/filter'>
    <rule description='Send␣messages␣from␣my␣friend␣to␣my␣home␣account
       ␣to␣be␣dealt␣with␣later'>
      <condition>
        <from xmlns='http://jabber.org/protocol/filter/header'>
           friend@theirisp.com</from>
      </condition>
      <action>
        <redirect xmlns='http://jabber.org/protocol/filter/redirect'>
           me@home.com</redirect>
      </action>
    </rule>
    <rule description='Copy␣messages␣from␣this␣spammer␣to␣our␣abuse␣
       address'>
      <condition>
        <from xmlns='http://jabber.org/protocol/filter/header'>
           spammer@badsite.com</from>
      </condition>
      <action>
        <copy xmlns='http://jabber.org/protocol/filter/redirect'>
           abuse@company.com</redirect>
      </action>
    </rule>
  </ruleset>
</iq>
```

On error, the server returns the original packet with an appropriate error:

Listing 13: Retrieving the ruleset failure

```
<iq type='error' to='rob@cataclysm.cx' id='filter2'>
  <ruleset xmlns='http://jabber.org/protocol/filter'/>
  <error code='403'>Forbidden</error>
</iq>
```

5 Error Codes

Possible errors are:

Code	Text	Description
403	Forbidden	The sender does not have permission to modify the ruleset for this entity.
404	Not Found	The entity does not exist.

6 Implementation notes

Ruleset processing should be the first thing that a service does when it receives a packet - even before processing privacy rules per XEP-0016.
Rules must be processed in the order they are given, and must be returned to a requesting entity in the same order.

7 Security Considerations

There are no security features or concerns related to this proposal.

8 IANA Considerations

This document requires no interaction with the IANA.

9 JANA Considerations

No namespaces or parameters need to be registered with JANA as a result of this document.