XEP-0356: Privileged Entity

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr

2015-03-23
Version 0.2

Status	Type	Short Name
Experimental	Standards Track	NOT_YET_ASSIGNED

This specification provides a way for XMPP entities to have a privileged access to some other entities data

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

XMPP components are used for long through Jabber Component Protocol (XEP-0114) [1], but are quite limited: they have a restricted access to other entities data, similar to what a client can do. This is sufficient for components like gateways, but very limiting for more complex components like a PubSub service. The goal of this XEP is to allow a component or any entity to have a "privileged" status, and access some other entity data with the same privileges than the entity itself, that means manage an entity roster on its behalf, send <message/> or receive <presence/> stanzas in the name of the server.
Privileged entities have numerous advantages, including:

- a step forward in decentralization: it is possible for an entity to do tasks which were before reserved to server itself. For example, a privileged pubsub component can offer access model based on publisher's roster
- better integration of components: a gateway can add items to an entity roster itself
- possibility to overpass a server limitation (typically: incomplete Personal Eventing Protocol (XEP-0163) [2] implementation)
- quick development cycle: developers can implement the components they need without waiting for a new server release
- server agnostic

Privileged entity has been created with the main goal to create an external, server agnostic, PEP service. It is restricted to only a couple of features, see Acknowledgements section for more details.
This XEP is complementary to Namespace Delegation (XEP-0355) [3] (and works in a similar way), although they can be used together or separately. To build something like an external PEP service, it is necessary to use both XEPs.

2 Requirements

A privileged entity must be able to do what a PEP service can do and to access roster, so it must be able to (according to configuration):

- get and modify the roster of any entity managed by the server
- send a <message/> stanza on behalf of the server

[1] XEP-0114: Jabber Component Protocol <https://xmpp.org/extensions/xep-0114.html>.
[2] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[3] XEP-0355: Namespace Delegation <https://xmpp.org/extensions/xep-0355.html>.

- access <presence/> informations for entities in a managed entity's roster (and for managed entity itself)

The privilege mechanism MUST be totally transparent for the managed entities.

3 Glossary

- **Privileged entity** — the entity which has a privileged status.
- **Managed entity** — the entity that is managed by a privileged entity.

4 Accessing Roster

4.1 Server Allows Roster Access

Roster access is granted in the server configuration. Roster access can have 4 types:

- **none** — the entity is not allowed to access managed entity roster at all. This is usually the default value.
- **get** — the entity is allowed to send <iq/> stanzas of type *'get'* for the namespace 'jabber:iq:roster'.
- **set** — the entity is allowed to send <iq/> stanzas of type *'set'* for namespace 'jabber:iq:roster'.
- **both** — the entity is allowed to send <iq/> stanzas of type *'get'* and *'set'* for namespace 'jabber:iq:roster'.

4.2 Server Advertises Entity Of Allowed Permission

Once a privileged entity is authenticated and stream is started, the server send it a <message/> stanza with a <privilege/> elements which MUST have the namespace 'urn:xmpp:privilege:1'. This element contains <perm/> elements which MUST contain a 'access' attribute of the value "roster" and a 'type' attribute which must correspond to the type configured as specified in "Server Allows Roster Access" section

Listing 1: Server Advertises Roster Privilege

```
<message from='capulet.net' to='pubub.capulet.lit' id='12345'>
    <privilege xmlns='urn:xmpp:privilege:1'>
        <perm access='roster' type='both'/>
    </privilege>
</message>
```

Here *pubsub.capulet.lit* is allowed to do *get* and *set* operations on all entities managed by capulet.lit

4.3 Privileged Entity Manage Roster

Doing a *get* or *set* operation on the roster of a managed entity is done in the usual way (as described in RFC 6121 [4] section 2), except that the 'to' attribute is set to the attribute of the managed entity. The server MUST check that the privileged entity has right to *get* or *set* the roster of managed entity, and MUST return a <forbidden/> error if it is not the case:

Listing 2: Privileged Entity Get Managed Entity Roster

```
<iq id='roster1'
    from='pubsub.capulet.lit'
    to='juliet@example.com'
    type='get'
    id='roster1'>
        <query xmlns='jabber:iq:roster'/>
</iq>
```

The server then answers normally, as it would have done to the managed entity:

Listing 3: Server Answers To Privileged Entity

```
<iq id='roster1'
    from='juliet@example.com'
    to='pubsub.capulet.net'
    type='result'>
    <query xmlns='jabber:iq:roster' ver='ver7'>
        <item jid='nurse@example.com'/>
        <item jid='romeo@example.net'/>
    </query>
</iq>
```

5 Message Permission

5.1 Authorizing Messages

In some cases, it can be desirable to send notifications (e.g. PEP service), so the privileged entity must be able to send <message/> stanzas. This is allowed in server configuration in the same way as for roster permission. The permission type can have the following values:

[4] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.

- **none** — the entity is not allowed to send <message/> stanza in the name of the server. This is usually the default value.
- **outgoing** — the entity is allowed to send <message/> stanzas in the name of the server, according to following restrictions.

A privileged entity can then send message on the behalf either of the server or of a bare JID of an entity managed by the server (i.e. a bare jid with the same domain as the server), using Stanza Forwarding (XEP-0297) [5]. The <forwarded/> element MUST be a child of a <privilege/> element with a namespace of *'urn:xmpp:privilege:1'*, with the following restriction:

1. forwarded <message/> 'from' attribute MUST be a bare JID from the server, no resource is allowed

If this rule is violated, the server MUST return a <message/> error with condition <forbidden/>, as in RFC 6120 [6] section 8.3.3.4.

5.2 Advertising Permission

Server advertises "message" permission in the same way as for "roster" permission, except that 'access' attribute has the value of "message", and the 'type' attribute as a value of 'outgoing':

Listing 4: Server Advertises Roster And Message Privileges

```
<message from='capulet.net' to='pubub.capulet.lit' id='54321'>
    <privilege xmlns='urn:xmpp:privilege:1'>
        <perm access='roster' type='both'/>
        <perm access='message' type='outgoing'/>
    </privilege>
</message>
```

5.3 Sending Messages

Now that *pubsub.capulet.lit* is allowed, it can send messages using <forwarded/> elements.

Listing 5: privileged entity send a notificaction message

```
<message from='pubsub.capulet.lit' to='capulet.lit' id='notif1'>
    <privilege xmlns='urn:xmpp:privilege:1'>
        <forwarded xmlns='urn:xmpp:forward:0'>
            <message from='juliet@capulet.lit'
```

[5] XEP-0297: Stanza Forwarding <https://xmpp.org/extensions/xep-0297.html>.
[6] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

```
                id='foo'
                to='romeo@montague.lit/orchard'
                xmlns='jabber:client'>
                <event xmlns='http://jabber.org/protocol/pubsub#event'
                    >
                    <items node='http://jabber.org/protocol/tune'>
                        <item>
                            <tune xmlns='http://jabber.org/protocol/
                                tune'>
                                <artist>Gerald Finzi</artist>
                                <length>255</length>
                                <source>Music for "Love's␣Labors␣Lost"
                                    (Suite for small orchestra)</
                                    source>
                                <title>Introduction (Allegro vigoroso)
                                    </title>
                                <track>1</track>
                            </tune>
                        </item>
                    </items>
                </event>
                <delay xmlns='urn:xmpp:delay' stamp='2014-11-25
                    T14:34:32Z'/>
            </message>
        </forwarded>
    </privilege>
</message>
```

The server sees that forwarded message 'from' attribute (*juliet@capulet.lit*) is a bare JID of the server, and that outgoing message permission was granted; it can now send the notification:

Listing 6: server sends the notification as if it was originating from him

```
<message from='juliet@capulet.lit'
    id='bar'
    to='romeo@montague.lit/orchard'>
    <event xmlns='http://jabber.org/protocol/pubsub#event'>
        <items node='http://jabber.org/protocol/tune'>
            <item>
                <tune xmlns='http://jabber.org/protocol/tune'>
                    <artist>Gerald Finzi</artist>
                    <length>255</length>
                    <source>Music for "Love's␣Labors␣Lost" (Suite for
                        small orchestra)</source>
                    <title>Introduction (Allegro vigoroso)</title>
                    <track>1</track>
                </tune>
            </item>
        </items>
```

```
    </event>
    <delay xmlns='urn:xmpp:delay' stamp='2014-11-25T14:34:32Z'/>
</message>
```

6 Presence Permission

6.1 Managed Entity Presence

It can be often desirable for a privileged entity to have presence information of the managed entities (e.g. to know when to send them notificiations). As privileges must be transparent for the managed entity, this presence has to be sent by the server without modifying managed entity roster.
This is allowed in server configuration in the same way as for *roster* and *message* permissions. The "presence" type can have the following values:

- **none** — the entity is not allowed to access <presence/> informations at all. This is usually the default value.
- **managed_entity** — the entity is allowed to receive managed entity presence (see below).
- **roster** — the entity is allowed to receive presence informations of managed entity contacts, see Roster Presence section.

If the privilege is granted, the server MUST use a directed presence from the full jid of the managed entity, to the privileged entity, as specified in RFC 6121 [7] section 4.6, on the behalf of managed entity each time its presence information change.
Only presences with no 'type' attribute or with a 'type' attribute with the value "unavailable" are transmitted to the privileged entity, the server MUST NOT transmit <presence/> stanza of any other type.

6.2 Advertising Permission

Server advertises "presence" permission in the same way as for "roster" or "message" permissions, except that 'access' attribute has the value of "presence", and the 'type' attribute has a value of "managed_entity"

6.3 Server Send presence informations

Once the "presence" permission is granted, the server send presence informations:

[7] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.

Listing 7: server receives new presence from Juliet

```
<presence from='juliet@capulet.lit/balcony'
          id='presence1'
          xml:lang='en'>
    <show>chat</show>
    <status>Staying on the balcony</status>
</presence>
```

Listing 8: server redirects presence to privileged entity

```
<presence from='juliet@capulet.lit/balcony'
          to='pubsub.capulet.lit'
          id='presence1'
          xml:lang='en'>
    <show>chat</show>
    <status>Staying on the balcony</status>
</presence>
```

6.4 Roster Presence

In addition to "managed entity presence", a privileged entity may need to know when a contact in managed entity roster is online (for example, it's necessary for a PEP service because of the presence default access model).
As for other permissions, the access is granted in server's configuration, but there is an additional restriction: the privileged entity MUST have read permission on roster namespace (i.e. 'type' attribute in allowed <perm> of access *roster* MUST have a value of either **get** or **both**).
If the privilege is granted, the server MUST send to the privileged entity every presence information with no 'type' attribute or with a 'type' with a value of 'unavailable' that the privileged entity is receiving or would receive if it were available. It do it in the same way as for managing entity by using directed <presence/> from the full jid of the entity from which presence information has changed, to the privileged entity. If the managed entity is unavailable but the privileged entity is available, the server MUST send <presence/> stanza to the later anyway.
Having "roster" type for "presence" permission imply that you have also implicitly "managed_entity" type.
The server MUST reject the permission if the privileged entity doesn't have read permission on roster namespace.
Note: this permission should be given carefully, as it gives access to presence of potentially a lot of entities to the privileged entity (see security considerations).

6.5 Advertising Permission

Server advertises roster "presence" permission in the same way as for other permissions, except that the 'access' attribute has the value of "presence", and the 'type' attribute has a

value of ”roster”

Listing 9: Server Advertises Roster, Message, Managed Entity Presence and Roster Presence Privileges

```
<message from='capulet.net' to='pubub.capulet.lit' id='54321'>
    <privilege xmlns='urn:xmpp:privilege:1'>
        <perm access='roster' type='both'/>
        <perm access='message'/>
        <perm access='presence' type='roster'/>
    </privilege>
</message>
```

Note the presence of *roster* permission request.

6.6 Privileged Entity Receive Roster Presences

Listing 10: server receives new presence from Romeo, which is in Juliet's roster

```
<presence from='romeo@montaigu.lit/orchard'/>
```

Listing 11: server sends the presence as usually, but also to the privileged entity

```
<presence from='romeo@montaigu.lit/orchard'
          to='juliet@capulet.lit'/>
<presence from='romeo@montaigu.lit/orchard'
          to='pubsub.capulet.lit'/>
```

7 Business Rules

1. For ”presence” access, if a privileged entity is connected after first <presence/> stanzas have been received, the server MUST send it all the <presence/> stanzas with no 'type' attribute it would have had if it was connected first (in other words: all the presences informations for connected entities it has access to).

2. For ”presence” access, if a privileged entity is supposed to received several time the same <presence/> stanza, the server SHOULD send it only once. For example: if *pubsub.capulet.lit* has a ”presence” access with a ”roster” type for *capulet.lit*, and *juliet@capulet.lit* and *nurse@capulet.it* both have *romeo@montague.lit* in their roster. When romeo is available, *pubsub.capulet.lit* shoud have its <presence/> stanza only once (instead of 2 times).

8 Security Considerations

1. Privileged entitiy has access to sensitive data, and can act as the server itself, permissions should be granted carefuly, only if you absolutely trust the entity.
2. Roster presence is particulary sensitive, because presence informations of whole rosters are shared.
3. Generaly, the server MUST NOT allow the privileged entity to do anything that the managed entity could not do.

9 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [8].

10 XMPP Registrar Considerations

10.1 Protocol Namespaces

The XMPP Registrar [9] includes 'urn:xmpp:privilege:1' in its registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).

- urn:xmpp:privilege:1

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

11 XML Schema

[8] 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/>.

[9] 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/>.

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:privilege:1'
    xmlns='urn:xmpp:privilege:1'
    elementFormDefault='qualified'>

  <xs:element name='privilege'>
      <xs:complexType>
          <xs:element name='perm'
                      maxOccurs='unbounded'>
              <xs:complexType>
                  <xs:attribute name='access' use='required' type='
                     xs:string'/>
                      <xs:simpleType base='xs:NMTOKEN'>
                          <xs:enumeration value='roster'/>
                          <xs:enumeration value='message'/>
                          <xs:enumeration value='presence'/>
                      </xs:simpleType>
                  <xs:attribute name='type' use='required'>
                      <xs:simpleType base='xs:NMTOKEN'>
                          <xs:enumeration value='none'/>
                          <xs:enumeration value='get'/>
                          <xs:enumeration value='set'/>
                          <xs:enumeration value='both'/>
                          <xs:enumeration value='outgoing'/>
                          <xs:enumeration value='managed_entity'/>
                          <xs:enumeration value='roster'/>
                      </xs:simpleType>
                  </xs:attribute>
              </xs:complexType>
          </xs:element>
      </xs:complexType>
</xs:element>

</xs:schema>
```

12 Acknowledgements

Thanks to Sergey Dobrov, Dave Cridland, Steven Lloyd Watkin, Lance Stout, Johannes Hund, Kurt Zeilenga and Kevin Smith for their feedbacks. Thanks to Adrien Cossa for his typos/style corrections.

Privileged entity was initialy written to be a generic identity based access control (IBAC) which allows an entity to access sensitive data. After a discussion on standard mailing list, it has been decided to restrict the current XEP to immediate needs to build an external PEP service,

and to implement separately an Attribute Based Access Control (ABAC) which is more modern, generic and flexible. This XEP is still interesting for being easy to implement and doing the job.