XEP-0346: Form Discovery and Publishing

Kevin Smith
mailto:kevin.smith@isode.com
xmpp:kevin.smith@isode.com

2017-09-11
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification describes a series of conventions that allow the management of form templates and publishing of completed forms.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

There are many circumstances in which it is necessary for entities to 'fill out forms' to be consumed by other entities (such an example might be for reporting an accident in the workplace). This document provides a method by which an entity can discover which forms are available, fetch the templates and submit the completed forms, using standard XMPP Publish-Subscribe (XEP-0060) [1].
This is achieved by every form having a pair of pubsub nodes on the same service; one of the nodes contains the template form (the empty form that is to be completed) and the other is used for publishing completed forms.

2 Use Cases

2.1 Node naming

Pubsub nodes used for these forms are comprised of a standard prefix and an application-specific suffix. Templates and completed forms for the same form type have the same application-specific suffix, but a different prefix. The prefix for form templates is "fdp/template/" and for completed forms is "fdp/submitted/".
The application-specific suffix must be guaranteed to be unique to the application, so it is suggested to start with a domain under the application author's control; as such if Isode Ltd. were to use this approach for feedback on the Christmas party, a node ID might be formed as "fdp/template/isode.com/christmas-party-feedback/2013" to contain the form template and "fdp/submitted/isode.com/christmas-party-feedback/2013" for the submitted nodes.

2.2 Listing available templates

To find the templates present on a pubsub service, do a disco#items as described in Publish-Subscribe (XEP-0060) [2] section 5.2. Those items that have a node ID that starts with "fdp/template" are form templates.

Listing 1: User discovers available nodes

```
<iq id="34385937-3740-411d-a360-374e9ba73202" to="pubsub.stan.isode.
    net" type="get">
  <query xmlns="http://jabber.org/protocol/disco#items"/>
</iq>

<iq from='pubsub.stan.isode.net' to='sysop@stan.isode.net/4
    d67a58b4d16cc1d' type='result' id='34385937-3740-411d-a360-374
    e9ba73202'>
```

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

```
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <item jid='pubsub.stan.isode.net' node='fdp/submitted/stan.isode.
       net/accidentreport' name='AccidentReport'/>
    <item jid='pubsub.stan.isode.net' node='fdp/template/stan.isode.
       net/accidentreport' name='AccidentReport'/>
  </query>
</iq>
```

In the above example there is one form available, ”stan.isode.net/accidentreport”

2.3 Fetching a template

To fetch a template, first identify the node that the template is stored in and then request the last published item for that node, as in Publish-Subscribe (XEP-0060) [3] 6.5.5

Listing 2: User fetches a template

```
<iq id="fb73efb3-356e-43ce-98a3-f76d86132745" to="pubsub.stan.isode.
   net" type="get">
  <pubsub xmlns="http://jabber.org/protocol/pubsub">
    <items max_items="1" node="fdp/template/stan.isode.net/
       accidentreport" xmlns="http://jabber.org/protocol/pubsub"/>
  </pubsub>
</iq>

<iq from='pubsub.stan.isode.net' to='sysop@stan.isode.net/4
   d67a58b4d16cc1d' type='result' id='fb73efb3-356e-43ce-98a3-
   f76d86132745'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='fdp/template/stan.isode.net/accidentreport'>
      <item id='version01'>
        <x xmlns='jabber:x:data' type='form'>
          <instructions>Please fill in all fields to complete this
             accident report.</instructions>
          <title>Accident report form.</title>
          <field type='list-single' label='Place␣of␣fall:' var='Place'
             >
            <option label="Kitchen">
              <value>Kitchen</value>
            </option>
            <option label="Conference␣Room">
              <value>Conference Room</value>
            </option>
            <option label="Engineering">
              <value>Engineering</value>
            </option>
```

[3] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

```
          </field>
          <field type='text-single' label='Type of injury:' var='Type'
              >
          </field>
          <field type='boolean' label='Ambulance needed?' var='
              Ambulance'>
          </field>
        </x>
      </item>
    </items>
  </pubsub>
</iq>
```

2.4 Template format

The format of the template should be a standard Data Forms (XEP-0004) [4] form; this can be extended with Data Forms Validation (XEP-0122) [5] and/or Data Forms Layout (XEP-0141) [6] as needed by individual form applications. The template will be the pubsub item.

2.5 Submitting a completed form

When the form has been completed, the resultant Data Forms (XEP-0004) [7] payload shall be published to the completed items node corresponding to the form type using the protocol in Publish-Subscribe (XEP-0060) [8] 7.1. The node ID for publishing completed forms corresponds to the node the template was stored in, with the fdp/template prefix replaced with the fdp/submitted prefix.

Listing 3: User submits the completed form

```
<iq id="fc2e69de-4626-47d3-b04b-1cd6b4049b3f" to="pubsub.stan.isode.
    net" type="set">
  <pubsub xmlns="http://jabber.org/protocol/pubsub">
    <publish node="fdp/submitted/stan.isode.net/accidentreport" xmlns=
        "http://jabber.org/protocol/pubsub">
      <item id="" xmlns="http://jabber.org/protocol/pubsub">
        <x type="submit" xmlns="jabber:x:data">
          <title>Accident report form.</title>
          <field label="Place of fall:" type="list-single" var="Place"
              >
            <value>Kitchen</value>
          </field>
```

[4] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[5] XEP-0122: Data Forms Validation <https://xmpp.org/extensions/xep-0122.html>.
[6] XEP-0141: Data Forms Layout <https://xmpp.org/extensions/xep-0141.html>.
[7] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[8] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

```
          <field label="Type of injury:" type="text-single" var="Type"
              >
            <value>Ankle</value>
          </field>
          <field label="Ambulance needed?" type="boolean" var="
              Ambulance">
            <value>1</value>
          </field>
        </x>
      </item>
    </publish>
  </pubsub>
</iq>
```

2.6 Monitoring completed form

An entity can monitor completed forms by subscribing to the completed form node, as described in Publish-Subscribe (XEP-0060) [9] 6.1.

Listing 4: User subscribes to a completed form node

```
<iq id="0f66fbcf-6148-40ed-a084-a3a5e2a71329" to="pubsub.stan.isode.
    net" type="set">
  <pubsub xmlns="http://jabber.org/protocol/pubsub">
    <subscribe jid="sysop@stan.isode.net/763be9c30f8ee893" node="fdp/
        submitted/stan.isode.net/accidentreport" xmlns="http://jabber.
        org/protocol/pubsub"/>
  </pubsub>
</iq>
```

2.7 Publishing form templates

Form templates are made available by publishing them to the template node for that form using the protocol in Publish-Subscribe (XEP-0060) [10] 7.1.

Listing 5: Administrator publishes a form template

```
<iq id="309d632d-fa68-4004-8611-d11cc5074d66" to="pubsub.stan.isode.
    net" type="set">
  <pubsub xmlns="http://jabber.org/protocol/pubsub">
    <publish node="fdp/template/stan.isode.net/accidentreport">
      <item id="version01">
        <x xmlns="jabber:x:data" type="form">
```

[9] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[10] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

```
          <instructions>Please fill in all fields to complete this
              accident report.</instructions>
          <title>Accident report form.</title>
          <field type="list-single" label="Place of fall:" var="Place"
              >
            <option label="Kitchen">
              <value>Kitchen</value>
            </option>
            <option label="Conference Room">
              <value>Conference Room</value>
            </option>
            <option label="Engineering">
              <value>Engineering</value>
            </option>
          </field>
          <field type="text-single" label="Type of injury:" var="Type"
              >
        </field>
          <field type="boolean" label="Ambulance needed?" var="
              Ambulance">
        </field>
      </x>
    </item>
  </publish>
</pubsub>
</iq>
```

2.8 Node configuration

Template nodes must be configured to support at least one persistent item. Both templates and published nodes need to be configured with appropriate access to publish and subscribe. Deployments may be configured such that entities are allowed to read the templates and submit completed forms, but not to read the completed forms.

3 Determining Support

All of the form activity happens over standard pubsub, so initial discovery is of the pubsub domain. A pubsub domain supporting these forms will have an additional disco identity (additional to the standard pubsub identity) with a category of 'pubsub' and a type of 'urn:xmpp:fdp:0'.

Listing 6: Entity Queries Pubsub Service Regarding Supported Features

```
<iq type='get'
    from='sysop@stan.isode.net/763be9c30f8ee893'
```

```
    to='pubsub.stan.isode.net'
    id='feature1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 7: Pubsub Service Includes FDP in its identities

```
<iq type='result'
    from='pubsub.stan.isode.net'
    to='sysop@stan.isode.net/763be9c30f8ee893'
    id='feature1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='pubsub' type='service'/>
    <identity category='pubsub' type='urn:xmpp:fdp:0'/>
    <feature var='http://jabber.org/protocol/pubsub'/>
  </query>
</iq>
```

Discovery of the template forms or completed form nodes happens using the protocol described in Use Cases.

4 Security Considerations

This document introduces no security considerations beyond those in Publish-Subscribe (XEP-0060) [11].

5 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [12].

6 XMPP Registrar Considerations

Include the "urn:xmpp:fdp:0" namespace in the registry of protocol namespaces. Include "urn:xmpp:fdp:0" as an additional type in the pubsub category of service discovery identities.

[11] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

[12] 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 XML Schema

As this document only defines semantics for existing protocol, additional schemas are not required.

8 Acknowledgements

Thanks to Matthew Wild, Richard Maudsley and Alex Clayton.