TS-001
Tool Registration, Discovery and Activation

Abstract: This XMPP extension describes a way in which tools may register themselves such that other tools can discover them and, for tools using a proxy, a way to activate them.

Authors: Brian Bonnett, Les D'Souza, Kingston Duffie, Kenneth Green, Keith Kidd, Todd Law, Tom McBeath, Madhusudan Nanjanagud, Eric Miller, Kris Raney, Kate Xu

Status: Final

Revision: 01

Revision date: Jun 2011

Submission: ntaf ts-001

Legal Notices

This Specification has been created by the Network Test Automation Forum (NTAF). NTAF reserves the rights to at any time add to, amend, modify or withdraw all or any portion of this Specification.

Note: All parties who in any way intend to use this Specification for any purpose, are hereby put on notice that the possibility exists that practicing under this Specification may require the use of inventions covered by the patent rights held by third parties. By publication of this Specification NTAF makes no representation or warranty whatsoever, whether expressed or implied, that practicing under this will not infringe any third party rights, nor does NTAF make any representation or warranty whatsoever, whether expressed or implied, with respect to (1) any claim that has been or may be asserted by any third party, (2) the validity of any patent rights related to any such claim, (3) or the extent to which a license to use any such rights may or may not be available on reasonable and nondiscriminatory terms, or on any terms at all.

Revision History

Version	Date	By	Changes
01	2011-06-01		First release

Table of Contents

Revision History 3

1. Introduction 5
   Motivation 5
2. Preliminaries 5
   Prerequisites 5
   Initialization 5
3. Use Cases 6
   Ensuring the Existence of the Root Collection Node 6
   Registering a typical “Always-On” Tool 7
   Registering a Proxied Tool 9
   Fetching a Tool Registry 10
   Subscribing for Tool Registry Updates 11
   Activating a Tool 12
   Deactivating a Tool 14
   Removing a Leaf Node 14
   Removing a Tool from a Node 15
4. Implementation Notes 15
   Locale-specific registrations 15
   Tool Location 15
5. XML Schemas 16

1. Introduction

This document describes an XMPP extension that allows an application or tool to register itself in a way that other interested entities can discover its existence. It also describes the mechanism by which a tool can be activated so that its automation harnesses are available for use.

Motivation

Traditionally, XMPP has been used for human-to-human exchanges. For users to establish communication with one another, they find out about each other through various mechanisms that may not involve XMPP. For example, one user may send an email to another user telling them about their Jabber ID. Then that user uses an XMPP client and adds that user's JID to their contact list.

In the world of machine-to-machine or human-to-machine scenarios, this ad hoc approach to discovery is problematic. Provisioning each tool's roster with a list of all other relevant tools is time-consuming and error-prone. Instead, one wants "plug-and-play" among tools where each tool automatically discovers all other tools available in their environment and can take advantage of them without requiring manual provisioning.

In addition, there are several use cases where using a "proxy" to represent other automatable tools is advantageous. For example, certain tools may be very resource intensive (CPU, memory) to have always running. In such cases, one would like to use a lightweight proxy tool to represent them until they are actually required. In other use cases, a single proxy could represent a pool of similar tools that can be instantiated on demand. This specification defines a registry structure and activation procedure that facilitates these scenarios.

This extension will typically be used in conjunction with the tool automation harness extension described in a separate document (see Tool Automation Harness).

2. Preliminaries

Prerequisites

The extension described herein requires that the XMPP server support XMPP's publish-subscribe service (XEP-0060) as well as XMPP Collection Nodes (XEP-0248).

Initialization

When a tool is to be made available to others via NTAF, it is registered using XMPP's publish-subscribe service (XEP-0060). For each tool or set of tools, a leaf node underneath a single root collection node is created. The publish-subscribe service JID and the collection node on that server are application-specific. For NTAF-compliant tools, the default address to use is the XMPP service address (e.g., "mycorp.com") prefixed by "pubsub." and the default collection node to use is "ntaf.tools". Tools may be provisioned to use a different pubsub address and/or node name.

The collection node may already exist on the XMPP server if the user has created it in advance, or if another tool has done so, however it is not guaranteed that the node will be preexisting. Thus, when a tool registers, it is recommended that the tool check for the existence of the collection node, and create it if it is not found.

3. Use Cases

Ensuring the Existence of the Root Collection Node

If the root collection node does not exist, the registering tool SHOULD create it, if the XMPP server supports doing so. If a tool does create the root node, there are a set of configuration items the tool MUST set. In short

- It must be a collection node
- Its parent collection must be the root node
- If children_association_policy is supported, it should be "open" *Note: the XMPP server administrator may use other policies at their discretion, but tools creating this automatically should default to "open"*
- The subscription type must be "all"
- The subscription depth must be "all"
- It should have an open access model
- It must be configured to deliver payloads
- It must be configured to notify when items are retracted
- It must be configured to persist items
- The publish model should be open

Example 1: Creating the root collection node

```
<iq id='yrx100-0' to='pubsub.mycorp.com' type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='ntaf.tools'/>
    <configure>
      <x type='submit' xmlns='jabber:x:data'>
        <field var='pubsub#node_type'>
          <value>collection</value>
        </field>
        <field var='pubsub#collection'>
          <value/>
        </field>
        <field var='pubsub#children'>
          <value/>
        </field>
        <field var='pubsub#children_association_policy'>
          <value>open</value>
        </field>
        <field var='pubsub#subscription type'>
          <value>all</value>
        </field>
        <field var='pubsub#subscription_depth'>
          <value>all</value>
        </field>
        <field type='list-single' var='pubsub#access model'>
          <value>open</value>
        </field>
        <field type='boolean' var='pubsub#deliver_payloads'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#notify retract'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#persist_items'>
          <value>1</value>
        </field>
        <field type='list-single' var='pubsub#publish_model'>
          <value>open</value>
        </field>
      </x>
    </configure>
  </pubsub>
</iq>
```


Registering a typical "Always-On" Tool

A tool that handles its own queries and activation will publish its information to a leaf node that is a child of the root collection node. If this is the first time the tool publishes its data, it must first create the node. The name of the node must be the full JID of that tool, including the resource.

The node created must meet the following criteria

- It must be a descendant of the root collection node. It is recommended for most circumstances that it be a direct child of the root collection node.
- It should be set to notify on item retraction
- It should be set to persist items
- It should have a max_items value of 1[1]
- In most cases, the access model should be open. However, a tool may choose to limit access.
- In most cases, the publish model should be "publishers", since only the tool will update its own registry.

The entry that is published to the registry contains a variety of information describing the tool:

- **toolType**: This is a URI that is intended to uniquely identify the type of tool. By using a URI, there is no chance of ambiguity between vendors and others about what type of tool this entry represents. Note that this is not intended to provide comprehensive information about the tool. For example, the same type of hardware-based tool may vary in its capabilities depending what components are installed into it. To be able to express this additional information, one can use the extension element (see below).
- **version**: This allows someone to see what version of the tool this entry represents. The format of the version field should consist of four integers (typically representing major, minor, version, maintenance, and build numbers) separated by periods (e.g., "4.1.0.24345").
- **harness**: Each tool may identify zero, one, or multiple automation harnesses that it supports. Each harness is identified using a URI. And for each harness, the entry must identify one or more session modes that it supports when using that harness.
- **documentation**: Each tool should include one or more documentation elements describing the tool. Each documentation element should include the same information, but potentially localized for different languages. (The xml:lang attribute for each document element identifies the language contained in that element.) The documentation element includes the name of the vendor (or other creator) of this tool, the name of the tool (appropriate for humans to understand), a multiline description providing more information about this tool, and a URI typically pointing to a web page providing more information about the tool.
- **published**: This is the timestamp when the particular registry entry was last updated. Publishers should typically update their registration each time that they start up – but not more than once per day unless the information has changed. (These updates help an administrator quickly identify tools that have not been active in a long time.)
- **location**: The location information allows someone viewing the registry to understand where the particular tool is physically located. Several elements within location provide different ways to describe that location information, as well as associated inventory information when appropriate. For more information, see Tool Location in the Implementation Notes section.
- **extension**: This element is optional, and multiple extensions are permitted. It provides an opportunity for the registrant to provide any additional vendor-specific information about the tool. These extension elements must each carry an XML namespace declaration so that there are no possible naming conflicts, and that a viewer of the registry can, in theory, reference a suitable schema definition based on that namespace. The contents of each extension are up to the vendor. However, it is recommended to limit

[1] This specification does not mandate the configuration of max_items with a specific value. But for most self-registering tools, a setting of "1" for max_items is strongly recommended as this ensures that the server will only keep the most current item within that node.

the amount of content in these extensions, as they may be published to many recipients – causing a potential performance issue for very large extensions.

Example 2. Registering a normal "always-on" tool via publish-subscribe

(Creating the node)

```
<iq id='yrx100-0' to='pubsub.mycorp.com' type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='acme_ascent@mycorp.com/1'/>
    <configure>
      <x type='submit' xmlns='jabber:x:data'>
        <field var='pubsub#collection'>
          <value>ntaf.tools</value>
        </field>
        <field var='pubsub#children'>
          <value/>
        </field>
        <field type='boolean' var='pubsub#deliver payloads'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#notify_retract'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#persist items'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#max_items'>
          <value>1</value>
        </field>
        <field type='list-single' var='pubsub#access_model'>
          <value>open</value>
        </field>
        <field type='list-single' var='pubsub#publish_model'>
          <value>publishers</value>
        </field>
      </x>
    </configure>
  </pubsub>
</iq>
```

(Publishing the tool information)

```
<iq type='set' from='acme ascent@mycorp.com/21' to='pubsub.mycorp.com' id='reg1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='acme_ascent@mycorp.com/1'>
      <item>
        <entry xmlns='http://ntaforum.org/registry'>
          <toolType>http://example.org/tools/acme/ascent/2.1</toolType>
          <version>1.2.0.24215</version>
          <published>2011-07-11T11:30:02-08:00</published>
          <harness xmlns='http://ntaforum.org/2011/harness'
                   name='http://example.org/acme/ascent/2.1'>
             <supportedMode>invisible_and_automated</supportedMode>
             <supportedMode>visible and interactive</supportedMode>
          </harness>
          <documentation xml:lang='en'>
             <vendor>Acme Test Tools</vendor>
             <name>Ascent</name>
             <description>A tool for demonstrating upward progress</description>
          </documentation>
          <location>
             <workspaceID>0d9374c4-aa1e-47f5-8b61-98bae0ad8ca1</workspaceID>
             <ipv4>10.155.2.151</ipv4>
             <hostname>jdoe43.mycorp.com</hostname>
             <description>2nd floor, bldg. 3, San Jose HQ</description>
          </location>
        </entry>
      </item>
```



```
    </publish>
  </pubsub>
</iq>
```

Registering a Proxied Tool

Some tools may be dormant or pooled in which case a proxy will handle certain operations on their behalf. This proxy will respond to harness-related queries and provide activation services for that tool. In this case, the registration describes the tool (or pool of equivalent tools) using an item within the leaf node. The leaf node represents the proxy. The items represent the menu of available tools handled by that proxy.

Note the absence of the max_items setting on the leaf node. That is because proxies may need to represent multiple tools and therefore may contain multiple items. For this reason, care must be taken that duplicates are avoided when adding items to these nodes.[2]

Example 3. Registering a proxy, and then a tool for which the proxy will take care of queries and activation

(Creating the node)

```
<iq id='yrx100-0' to='pubsub.mycorp.com' type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='acme proxy@mycorp.com/2'/>
    <configure>
      <x type='submit' xmlns='jabber:x:data'>
        <field var='pubsub#collection'>
          <value>ntaf.tools</value>
        </field>
        <field var='pubsub#children'>
          <value/>
        </field>
        <field type='boolean' var='pubsub#deliver_payloads'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#notify retract'>
          <value>1</value>
        </field>
        <field type='boolean' var='pubsub#persist_items'>
          <value>1</value>
        </field>
        <field type='list-single' var='pubsub#access_model'>
          <value>open</value>
        </field>
        <field type='list-single' var='pubsub#publish model'>
          <value>publishers</value>
        </field>
      </x>
    </configure>
  </pubsub>
</iq>
```

(Publishing the tool information)

```
<iq type='set' from='acme_sleeper@mycorp.com/4' to='pubsub.mycorp.com' id='reg2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node= acme_proxy@mycorp.com/2'>
      <item id='acme_proxy@mycorp.com/2#sleeper1'>
        <entry xmlns='http://ntaforum.org/registry'>
          <toolType>http://example.org/tools/acme/sleeper/1.1</toolType>
          <version>1.1.0.14153</version>
          <mode>dormant</mode>
          <proxy>acme_proxy@mycorp.com/2</proxy>
          <published>2011-07-11T11:32:41-08:00</published>
```

[2] Some versions of the OpenFire XMPP server's implementation of PubSub exhibit the annoying behavior of accepting new items into a leaf node when there is overlap in the ID of that item. Instead of overwriting the original item with the same ID, the server adds a new item and assigns it an auto-generated unique ID.


```
          <harness xmlns='http://ntaforum.org/2011/harness'
                   name='http://example.org/acme/sleeper/1.1>
             <supportedMode>invisible and automated</supportedMode>
             <supportedMode>visible_and_interactive</supportedMode>
          </harness>
          <documentation xml:lang='en'>
             <vendor>Acme Test Tools</vendor>
             <name>Sleeper</name>
             <description>A tool for whiling away the hours</description>
          </documentation>
          <location>
             <workspaceID>0e2222c4-aa1e-47f5-8b61-98bae0ad8ca1</workspaceID>
             <ipv4>10.155.2.151</ipv4>
             <hostname>jdoe43.mycorp.com</hostname>
             <description>2nd floor, bldg. 3, San Jose HQ</description>
          </location>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

Fetching a Tool Registry

Tools may want to find out about what tools have registered. To do this, they can use the standard publish-subscribe mechanisms to accomplish this.[3] If the XMPP server supports item retrieval on collection nodes, all Items may be retrieved in a single call.

Example 4. Fetching a list of available tools from a collection node

```
<iq type='get' from='client1@mycorp.com/1' to='pubsub.mycorp.com' id='q1'>
   <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='ntaf.tools'/>
   </pubsub>
</iq>

<iq type='result' to='client1@mycorp.com/1' from='pubsub.mycorp.com' id='q1'>
   <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='acme_ascent@mycorp.com/1'>
         <item id='JHGBFftydvbFDT'>
           <entry xmlns='http://ntaforum.org/registry'>
             <toolType>http://example.org/tools/acme/ascent/2.1</toolType>
             <version>1.2.0.24215</version>
             <published>2011-07-11T11:30:02-08:00</published>
             <harness xmlns='http://ntaforum.org/2011/harness'
                      name='http://example.org/acme/ascent/2.1'>
                <supportedMode>invisible and automated</supportedMode>
                <supportedMode>visible_and_interactive</supportedMode>
             </harness>
             <documentation xml:lang='en'>
                <vendor>Acme Test Tools</vendor>
                <name>Ascent</name>
                <description>A tool for demonstrating upward progress</description>
             </documentation>
             <location>
                <workspaceID>0f1414c4-aa1e-47f5-8b61-98bae0ad8ca1</workspaceID>
                <ipv4>10.155.2.133</ipv4>
                <hostname>server3.mycorp.com</hostname>
             </location>
           </entry>
         </item>
        </items>
        <items node='acme_proxy@mycorp.com/2'>
         <item id='acme_proxy@mycorp.com/2#sleeper1'>
```

[3] Note that the XMPP publish-subscribe standard provides a set of rich mechanisms for fetching and dealing with published information that is not covered in detail here. For more details, see XEP-0060.


```
          <entry xmlns='http://ntaforum.org/registry'>
            <toolType>http://example.org/tools/acme/sleeper/1.1</toolType>
            <version>1.1.0.14153</version>
            <mode>dormant</mode>
            <proxy>acme_proxy@mycorp.com/2</proxy>
            <published>2011-07-11T11:32:41-08:00</published>
            <harness xmlns='http://ntaforum.org/2011/harness'
                     name='http://example.org/acme/sleeper/1.1>
               <supportedMode>invisible and automated</supportedMode>
               <supportedMode>visible_and_interactive</supportedMode>
            </harness>
            <documentation xml:lang='en'>
               <vendor>Acme Test Tools</vendor>
               <name>Sleeper</name>
               <description>A tool for whiling away the hours</description>
            </documentation>
            <location>
               <workspaceID>0d9374c4-aa1e-47f5-8b61-98bae0ad8ca1</workspaceID>
               <ipv4>10.155.2.153</ipv4>
               <hostname>server1.mycorp.com</hostname>
            </location>
          </entry>
        </item>
      </items>
   </pubsub>
</iq>
```

Not all XMPP servers support this optional feature, however (for example, openfire does not, and will return a feature-not-implemented error on such a request.) In the case that the server does not, it is necessary to first query the list of child nodes, and then query the children individually.

Example 5. Fetching a list of child nodes from a collection node

```
<iq type='get' to='pubsub.mycorp.com' id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#items'
         node='ntaf.tools'/>
</iq>


<iq type='result' from='pubsub.mycorp.com' id='disco1' to='client1@mycorp.com/1'>
  <query xmlns='http://jabber.org/protocol/disco#items' node='ntaf.tools'>
    <item node='acme ascent@mycorp.com/1' name='' jid='pubsub.mycorp.com'/>
    <item node='acme_proxy@mycorp.com/2' name='' jid='pubsub.mycorp.com'/>
  </query>
</iq>
```

Subscribing for Tool Registry Updates

Some tools may want to keep track as other tools register. If so, they can accomplish this by subscribing to changes on either the root collection node (for notifications about all tool updates,) or directly on the specified node (for notifications only about that specific tool.) Generally speaking, tools that have subscribed should unsubscribe when they go offline to useless notifications.

Example 6. Subscribing for changes to the publish-subscribe node contents, and later unsubscribing

```
<iq type='set' from='client1@mycorp.com/1' to='pubsub.mycorp.com' id='sub1'>
   <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <subscribe node='ntaf.tools' jid='client1@mycorp.com/1'/>
   </pubsub>
</iq>

...

(New tool registration added to pubsub list here)
```



```
<message from='pubsub.mycorp.com' to='client1@mycorp.com' id='n1'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='foo@mycorp.com/100''>
       <item id='KJHFbnktBHtbkkuyg'>
          <entry xmlns='http://ntaforum.org/registry'>
             <toolType>http://example.org/tools/other/bigboy/4.0</toolType>
             <version>4.0.0.1</version>
             <published>2011-07-13T08:24:09-05:00</published>
             <documentation xml:lang='en'>
                <vendor>Other Tools, Inc.</vendor>
                <name>Big Boy</name>
                <description>Another interesting tool</description>
             </documentation>
             <location>
                <workspaceID>0a2434c4-aa1e-47f5-8b61-98bae0ad8ca1</workspaceID>
                <ipv4>10.155.2.101</ipv4>
                <hostname>lab4.mycorp.com</hostname>
             </location>
          </entry>
       </item>
    </items>
  </event>
</message>

...
```

(Tool is closing… Unsubscribing…)

```
<iq type='set' from='client1@mycorp.com/1' to='pubsub.mycorp.com' id='sub1'>
   <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <unsubscribe node='ntaf.tools' jid='client1@mycorp.com/1'/>
   </pubsub>
</iq>
```

Activating a Tool

In order to establish a session with a tool, one must first activate that tool. To do this, the requester issues an activate request to the JID found in the leaf node containing the item representing that tool. This activation request identifies the tool to be activated (using the ID of one of the items under that leaf node) and one or more harnesses supported by that tool that the requester intends to use, along with the mode or modes of that harness that may be needed. This activation procedure enables both proxy and non-proxy use cases. Proxies will activate the requested tool (if needed) and return a JID for the requester to use to open a session with the activated tool. Non-proxy tools may do nothing on activation and simply return their own JID. But the requester need not handle these cases any differently.

In this first example, an always-on tool has its own leaf node in the registry. When asked to activate, it responds immediately and returns its own JID for opening a session. In this case, one of the two supported modes on the only harness associated with that tool is included.

Example 8. Activating an always-on tool

```
<iq type='set' to='acme_ascent@mycorp.com/1'from='client1@mycorp.com/1' id='a1'>
   <activate xmlns='http://ntaforum.org/2011/activation'
             toolId='JHGBFftydvbFDT'>
      <harness xmlns='http://ntaforum.org/2011/harness'
                     name='http://example.org/acme/ascent/2.1'>
         <supportedMode>invisible and automated</supportedMode>
      </harness>
   </activate>
</iq>

<iq type='result' from='acme_ascent@mycorp.com'to='client1@mycorp.com/1' id='a1'>
   <activate xmlns='http://ntaforum.org/2011/activation'
             toolId='JHGBFftydvbFDT'/>
      <result>pass</result>
      <jid>acme_ascent@mycorp.com</jid>
```



```
      <activationRef/>
   </activate>
</iq>

<iq type='set' to='acme_ascent@mycorp.com'from='client1@mycorp.com/1' id='o1'>
   <open xmlns='http://ntaforum.org/2011/harness'
         harness='http://example.org/acme/ascent/2.1'
         mode='visible_and_interactive'>
      <activationRef/>
   </open>
</iq>

...
```

In this next example, a proxy is responsible for a tool that is "dormant" – i.e., that is not running until it is activated by the proxy. In this case, the proxy receives the activation request. It responds immediately indicating that the activation is pending. Then it proceeds to activate the dormant tool and, when complete, returns the activation request ID (i.e., the packet ID from the activation request) and the JID for that tool to the requester who then proceeds to open a session accordingly.

The activationRef field (if any) from the activate response MUST be copied into the corresponding element in the open request on the activated tool. This provides a mechanism for the activator to associate the activation with the open request.

These are returned in a message, in this case issued by the proxy. Note, however, that the requester cannot assume that the sender of the activation notification will come from the same JID that the request was issued to. (This enables scenarios where the activated tool itself, for example, issues the activation notification once it is ready.) The requester must depend on the requestId in the notification to make the proper association.

Example 9. Activating a dormant tool

```
<iq type='set' to='acme_proxy@mycorp.com/2' from='client1@mycorp.com/1' id='a5'>
   <activate xmlns='http://ntaforum.org/2011/activation'
             toolId='acme_sleeper@mycorp.com#sleeper1'/>
</iq>

<iq type='result' from='acme_proxy@mycorp.com/2' to='client1@mycorp.com/1' id='a5'>
   <activate xmlns='http://ntaforum.org/2011/activation'
             toolId='acme_proxy@mycorp.com#sleeper1'/>
      <result>pending</result>
   </activate>
</iq>

(Activation occurs here)

...

<message from='acme_proxy@mycorp.com/2' to='client1@mycorp.com/1' id='an21'>
   <activate xmlns='http://ntaforum.org/2011/activation'
             toolId='acme_proxy@mycorp.com#sleeper1'
             requestId='a5'/>
      <result>pass</result>
      <jid>acme_sleeper@mycorp.com/4</jid>
      <activationRef>432345</activationRef>
   </activate>
</iq>

<iq type='set' to='acme_sleeper@mycorp.com/4' from='client1@mycorp.com/1' id='o1'>
   <open xmlns='http://ntaforum.org/2011/harness'
         harness='http://example.org/acme/sleeper/1.2'
         mode='visible and interactive'>
      <activationRef>432345</activationRef>
   </open>
</iq>

...
```


Deactivating a Tool

Having activated a tool, the requester will typically use that activated tool to open harness sessions. (For more information, see Tool Automation Harness specification.) When the requester has finished using the activated tool – i.e., has closed all harness sessions used on that tool – it is required to deactivate the tool to indicate to the activator that it will not need the tool any further.

Deactivation is performed by sending a deactivate packet. For an always-on tool, the deactivate packet is sent to the tool. For a tool which can be dormant and uses a proxy, the packet is sent to both the proxy and the tool. This allows the proxy to deactivate the tool or the tool to deactivate itself, whichever better suits the proxy-tool design being implemented. The packet recipient is free to handle deactivation according to the use case. For simple always-on standalone tools, the deactivation can effectively be ignored. For proxy scenarios, the tool may be shut down or kept around for some amount of time in case it is needed again soon. So the "deactivate" packet is to be interpreted only as an indication that the requester no longer needs to use the tool for additional sessions.

In the example shown below, the tool activated in Example 9 is now deactivated. The activation reference from the activation is provided. Both the proxy and the tool receive a deactivate packet, and each one sends back a reply accordingly. The deactivate reply should be sent back immediately to the requester, indicating that it was received. The packet recipient may take additional time after this point to perform any long-running operations (such as shutting down the tool).[4]

Example 10. Deactivating a tool

```
<iq type='set' to='acme proxy@mycorp.com/2' from='client1@mycorp.com/1' id='deact1'>
   <deactivate xmlns='http://ntaforum.org/2011/activation'>
      <activationRef>432345</activationRef>
   </deactivate>
</iq>

<iq type='result' to='client1@mycorp.com/1' from='acme_proxy@mycorp.com/2' id='deact1'>
   <deactivate xmlns='http://ntaforum.org/2011/activation'/>
</iq>

<iq type='set' to='acme_proxy@mycorp.com#sleeper1' from='client1@mycorp.com/1' id='deact2'>
   <deactivate xmlns='http://ntaforum.org/2011/activation'>
      <activationRef>432345</activationRef>
   </deactivate>
</iq>

<iq type='result' to='client1@mycorp.com/1' from='acme_proxy@mycorp.com#sleeper1' id='deact2'>
   <deactivate xmlns='http://ntaforum.org/2011/activation'/>
</iq>
```

Removing a Leaf Node

A tool's registration may be removed by simply deleting its node. (In cases where a tool is handled by a proxy that may be handling other tools, then the item may be deleted while leaving the leaf node in place.)

Example 11. Removing a leaf node

```
<iq type='set'
    to='pubsub.mycorp.com'
    id='delete1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <delete node='acme_ascent@mycorp.com/1'/>
  </pubsub>
```

[4] If an entity learns via XMPP presence information that a requester has gone offline, it should assume that any tools activated by that requesting entity and any sessions opened by that entity are no longer going to be used. In other words, a requester going offline is equivalent to sending close packets for all of its active sessions and deactivate packets for all of its activated tools.


```
</iq>
```

Removing a Tool from a Node

In the case where a leaf node contains multiple items corresponding to different tools, then it may become necessary to remove an individual item within a leaf node. This is done using the standard XMPP mechanism by issuing a retract request.

Example 12. Removing a tool item

```
<iq type='set'
    to='pubsub.mycorp.com'
    id='delete2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='acme_proxy@mycorp.com/2'>
        <item id='acme_proxy@mycorp.com/2#sleeper1'/>
    </retract>
  </pubsub>
</iq>
```

4. Implementation Notes

Locale-specific registrations

The schema for the entry in the tool registry allows the registrant to include as many sets of documentation as it wants – each supporting a different locale. In this way, clients can choose among the supported locales for the human-oriented information describing the tool.

If the tool itself is locale-specific, then that locale should be listed first among its documentation listings in the registry. The tool is locale-specific if, for example, its user interface contains human-readable text (like labels, etc.) that are specific to one locale.

Tool Location

Part of the registration entry for a tool is the location element. This is useful for identifying where a registered tool is located. This may be useful for inventory and asset management purposes. But it is also important for interoperability reasons in some cases.

One particular case is when humans are involved in the use of the tools. In that case, the requesting tool (a test authoring tool, for example) is going to initiate an interactive session on behalf of a user with another tool of a certain type. In this case, the choice of which instance of a given type of tool to use is not arbitrary. The user would like to be able to see the user interfaces for the two tools on the same display. For that reason, it is important for one tool to be able to determine whether it is co-resident with another tool. To be co-resident does not only mean that the other tool is installed on the same PC, but also that it is available within the same user's workspace – so that it can share the display with the first tool.

To that end, the workspaceID element was added into the location information in the registry entry. This is meant to uniquely identify a specific user's environment on a specific machine. If an NTAF tool sees that another tool has an identical workspaceID in its registry entry, then it can be assured that it is accessible in the same display context.

When registering itself, a tool must populate the workspaceID using the following procedure:

- The tool first checks to see if a workspaceID is already available. This is done by looking for a file located in the user's "NTAF" folder (see below) called "workspaceID.txt". The contents of this file are read as

characters using UTF-8 encoding and the full contents are used to populate the workspaceID element in the location element.

- If the NTAF folder is missing, then it is created (see below).
- If the workspaceID.txt file is missing, then it is created and populated with a new universally unique identifier (UUID) encoded as a UTF-8 string. (See http://en.wikipedia.org/wiki/Universally_Unique_Identifier.) Note that most programming languages provide a convenient method for generating UUIDs (or GUIDs).
- This is then used to populate the workspaceID element.

The location of the NTAF folder depends on the host operating system.

- On UNIX-based systems (including Linux), the ntaf folder is placed inside the user's home directory. Therefore, the file will be located at ~/.ntaf/workspaceID.txt.
- On Apple systems, /Users/<user>/Library/Application Support/NTAF/workspaceID.txt.
- On Windows systems,the NTAF folder is located inside the user's Local Application Data folder, whose location depends on the version of Windows:
  - Windows7: \Users\<user>\AppData\local\NTAF\workspaceID.txt
  - Windows XP: \Documents and Settings\<user>\Local Settings\Application Data\NTAF\workspaceID.txt.
  - Windows Server 2003: c:\Users\<user>\AppData\Local\Application Data\NTAF\workspaceID.txt

5. XML Schemas

The following XML Schema Definition (XSD) describes the content of the payload inside the publish-subscribe item that is added in the tool registry's node.

```
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'
           xmlns:r='http://ntaforum.org/2011/registry'
           xmlns:h='http://ntaforum.org/2011/harness'
           xmlns:xml='http://www.w3.org/XML/1998/namespace'
           targetNamespace='http://ntaforum.org/2011/registry'
           elementFormDefault='qualified'
           attributeFormDefault='unqualified'>
    <xs:import namespace='http://www.w3.org/XML/1998/namespace' schemaLocation='xml-1998.xsd'/>
    <xs:import namespace='http://ntaforum.org/2011/harness' schemaLocation='harness.xsd'/>
    <xs:element name='entry'>
        <xs:complexType>
            <xs:sequence>
                <xs:element name='tooltype' type='xs:anyURI'/>
                <xs:element name='version' type='xs:string'/>
                <xs:element ref='h:harness' minOccurs='0' maxOccurs='unbounded'/>
                <xs:element name='documentation' maxOccurs='unbounded'>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name='vendor' type='xs:string'/>
                            <xs:element name='name' type='xs:string'/>
                            <xs:element name='description' type='xs:string'/>
                            <xs:element name='helpUri' type='xs:anyURI'/>
                        </xs:sequence>
                        <xs:attribute ref='xml:lang' use='required'/>
                    </xs:complexType>
                </xs:element>
                <xs:element name='published' type='xs:dateTime'/>
                <xs:element name='location'>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name='workspaceID' type='xs:string'/>
                            <xs:element name='ipv4' type='xs:string'
                                        minOccurs='0' maxOccurs='unbounded'/>
                            <xs:element name='ipv6' type='xs:string'
                                        minOccurs='0' maxOccurs='unbounded'/>
                            <xs:element name='hostname' type='xs:string' minOccurs='0'/>
```



```
                            <xs:element name='serialNumber' type='xs:string' minOccurs='0'/>
                            <xs:element name='assetNumber' type='xs:string' minOccurs='0'/>
                            <xs:element name='description' type='xs:string' minOccurs='0'/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name='extension' minOccurs='0' maxOccurs='unbounded'>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:any processContents='skip'/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Following is the schema for the activation request that is sent from a client to a proxy requesting that a dormant tool be activated, and the corresponding response.

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'
           xmlns:a='http://ntaforum.org/2011/activation'
           xmlns:h='http://ntaforum.org/2011/harness'
           targetNamespace='http://ntaforum.org/2011/activation'
           elementFormDefault='qualified'
           attributeFormDefault='unqualified'>
    <xs:import namespace='http://ntaforum.org/2011/harness' schemaLocation='harness.xsd'/>
    <xs:element name='activate'>
        <xs:complexType>
            <xs:sequence>
                <xs:element name='result' type='a:activationResult'/>
                <xs:element name='message' type='xs:string' minOccurs='0'/>
                <xs:element name='jid' type='xs:string' minOccurs='0'/>
                <xs:element name='activationRef' type='xs:string' minOccurs='0'/>
                <xs:element name='timestamp' type='xs:dateTime' minOccurs='0'/>
                <xs:element ref='h:harness' maxOccurs='unbounded'/>
            </xs:sequence>
            <xs:attribute name='toolId' type='xs:string' use='required'/>
            <xs:attribute name='requestId' type='xs:string'/>
        </xs:complexType>
    </xs:element>
    <xs:simpleType name='activationResult'>
        <xs:restriction base='xs:string'>
            <xs:enumeration value='pass'/>
            <xs:enumeration value='fail'/>
            <xs:enumeration value='pending'/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name='deactivate'>
        <xs:complexType>
            <xs:sequence>
                <xs:element name='activationRef' type='xs:string'/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

