XEP-0146: Remote Controlling Clients

Remko Tronçon
http://el-tramo.be/

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

2006-03-23
Version 1.0

Status	Type	Short Name
Active	Informational	rc

This document specifies recommended best practices for remote controlling clients using Ad-Hoc Commands.

Legal

Copyright

Permissions

Warranty

Liability

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/extensions/ipr-policy.shtml> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1 Introduction

When one has multiple clients at different locations logged in simultaneously, it is often desirable to control these clients from the client you are currently using. There are a number of common tasks one might want to perform remotely on clients: change the status of the client, forward all received unread messages to this client, and so on. Therefore, it makes sense to define a protocol for performing these tasks.
This document describes a protocol to perform a set of common tasks on a remote client, by specifying a profile of Ad-Hoc Commands [1].

2 Requirements

This document addresses the following requirements:

- Enable users to perform a set of common tasks on a remote client.
- Re-use existing XMPP and Jabber protocols wherever possible.

3 Discovery

A client MUST advertise any remote controlling commands it supports via Service Discovery [2] (as described in XEP-0050: Ad-Hoc Commands). Entity Capabilities [3] can be used to query capability of remote controlling commands in a client.

4 Use Cases

This document defines a profile of XEP-0050: Ad-Hoc Commands that enables a user to perform the following tasks on a remote client:

1. Change status
2. Forward unread messages residing at the remote client to the local client
3. Change run-time options
4. Accept pending file transfer requests
5. Leave groupchats

[1] XEP-0050: Ad-Hoc Commands <http://xmpp.org/extensions/xep-0050.html>.
[2] XEP-0030: Service Discovery <http://xmpp.org/extensions/xep-0030.html>.
[3] XEP-0115: Entity Capabilities <http://xmpp.org/extensions/xep-0115.html>.

Although this document aims to define common use cases for remote controlling clients, an implementation or deployment MAY support any subset and MAY support additional commands not defined herein.
Note: The text that follows assumes that implementors have read and understood XEP-0050: Ad-Hoc Commands.

4.1 Change Status

It is common to forget changing the status of a resource when leaving the client for a longer period. When realizing this while at another location, it might be desirable to change the status from there, to avoid contacts thinking that resource is attended and sending it messages.

Listing 1: Local Client Requests to Set the Status of a Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='set-status-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='http://jabber.org/protocol/rc#set-status'/>
</iq>
```

Unless an error occurs (see the Error Handling section below), the service SHOULD return the appropriate form.

Listing 2: Remote Client Replies with a Form to Set its Status

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='set-status-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#set-status'
           sessionid='set-status:20040727T0337Z'
           status='executing'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Change Status</title>
      <instructions>Choose the status and status message</instructions
          >
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field label='Status'
             type='list-single'
```

```
            var='status'>
        <required/>
        <value>online</value>
        <option label='Chat'>
            <value>chat</value>
        </option>
        <option label='Online'>
            <value>online</value>
        </option>
        <option label='Away'>
            <value>away</value>
        </option>
        <option label='Extended Away'>
            <value>xa</value>
        </option>
        <option label='Do Not Disturb'>
            <value>dnd</value>
        </option>
        <option label='Invisible'>
            <value>invisible</value>
        </option>
        <option label='Offline'>
            <value>offline</value>
        </option>
      </field>
      <field label='Priority'
             type='text-single'
             var='status-priority'>
        <value>5</value>
      </field>
      <field label='Message'
             type='text-multi'
             var='status-message'/>
    </x>
  </command>
</iq>
```

Listing 3: Local Client Submits Set Status Form to Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='set-status-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#set-status'
           sessionid='set-status:20040727T0337Z'>
    <x xmlns='jabber:x:data' type='form'>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
```

```
      </field>
      <field type='list-single' var='status'>
        <value>xa</value>
      </field>
      <field type='text-single' var='status-priority'>
        <value>-1</value>
      </field>
      <field type='text-multi' var='status-message'>
        <value>In my chamber.</value>
      </field>
    </x>
  </command>
</iq>
```

If the 'status-priority' variable is omitted, the client SHOULD NOT change the priority of the client

Listing 4: Remote Client Informs Local Client of Completion

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='set-status-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#set-status'
           sessionid='set-status:20040727T0337Z'
           status='completed'/>
</iq>
```

Notification of completion MAY include the processed data in a data form of type 'result'.

4.2 Forward Unread Messages Residing at a Remote Client

A user might want to forward all the unread messages residing at the remote client to the local client (e.g. when the remote client was accidentally left on-line, and has received messages in the meantime).
For example, suppose Romeo sends a message to Juliet, thinking she is still on her balcony. The balcony client receives the message:

Listing 5: Remote Client Receives Message

```
<message from='romeo@example.com/orchard'
         to='juliet@example.com/balcony'>
  <subject>Just saying hi</subject>
  <body>Hello Juliet!</body>
</message>
```

However, Juliet is in her chamber, so she doesn't know about the message (yet). Realizing she left her balcony client unattended, she sends a request to the remote client to forward all unread messages.

Listing 6: Local Client Requests to Forward Unread Messages Currently Residing at the Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='forward-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='http://jabber.org/protocol/rc#forward'
           sessionid='forward:20040727T0337Z'/>
</iq>
```

The client forwards all unread messages to the local client, adding information about the origin of the message (using the 'ofrom' Extended Stanza Addressing [4] address, and the Delayed Delivery [5] timestamp of the original message). The chamber client receives both these messages and a confirmation that the command was completed.

Listing 7: Remote Client Forwards All Unread Messages to Local Client

```
<message from='juliet@example.com/balcony'
         to='juliet@example.com/chamber'>
  <subject>Just saying hi</subject>
  <body>Hello Juliet!</body>
  <addresses xmlns='http://jabber.org/protocol/address'>
    <address type='ofrom' jid='romeo@example.com/orchard'/>
  </addresses>
  <delay xmlns='urn:xmpp:delay'
         from='juliet@capulet.com/balcony'
         stamp='2002-09-10T23:41:07Z'/>
</message>
```

Listing 8: Remote Client Informs Local Client of Completion

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='forward-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#forward'
           sessionid='forward:20040727T0337Z'
           status='completed'/>
</iq>
```

[4] XEP-0033: Extended Stanza Addressing <http://xmpp.org/extensions/xep-0033.html>.
[5] XEP-0203: Delayed Delivery <http://xmpp.org/extensions/xep-0203.html>.

A client MAY provide a more fine-grained implementation, e.g. by presenting the requester an extra form to select which messages have to be forwarded.

4.3 Change Run-Time Options

It might be desirable to remotely set some run-time options of a client. For example, when neighbours complain about the sounds your client makes while you're at another location, you could turn the sounds off at the remote client.

Listing 9: Local Client Requests to Change Options of a Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='set-options-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='http://jabber.org/protocol/rc#set-options'/>
</iq>
```

Unless an error occurs (see the Error Handling section below), the service SHOULD return the appropriate form.

Listing 10: Remote Client Replies with a Form to Set its Options

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='set-options-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
             node='http://jabber.org/protocol/rc#set-options'
             sessionid='set-options:20040727T0337Z'
             status='executing'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Set Options</title>
      <instructions>Set the desired options</instructions>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field label='Play sounds'
             type='boolean'
             var='sounds'>
        <value>1</value>
      </field>
      <field label='Automatically Go Offline when Idle'
             type='boolean'
```

```
            var='auto-offline'>
        <value>0</value>
      </field>
      <field label='Automatically Open New Messages'
            type='boolean'
            var='auto-msg'>
        <value>0</value>
      </field>
      <field label='Automatically Accept File Transfers'
            type='boolean'
            var='auto-files'>
        <value>0</value>
      </field>
      <field label='Automatically Authorize Contacts'
            type='boolean'
            var='auto-auth'>
        <value>0</value>
      </field>
    </x>
  </command>
</iq>
```

Listing 11: Local Client Submits Set Options Form to Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='set-options-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#set-options'
           sessionid='set-options:20040727T0337Z'>
    <x xmlns='jabber:x:data' type='form'>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field type='boolean' var='sounds'>
        <value>0</value>
      </field>
      <field type='boolean' var='auto-offline'>
        <value>0</value>
      </field>
      <field type='boolean' var='auto-msg'>
        <value>0</value>
      </field>
      <field type='boolean' var='auto-files'>
        <value>0</value>
      </field>
      <field type='boolean' var='auto-auth'>
        <value>0</value>
```

```
      </field>
    </x>
  </command>
</iq>
```

The remote client sets the values of the options to their requested value. If a variable is omitted, the client SHOULD NOT change the value of the corresponding option.

Listing 12: Remote Client Informs Local Client of Completion

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='set-options-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#set-options'
           sessionid='set-options:20040727T0337Z'
           status='completed'/>
</iq>
```

Notification of completion MAY include the processed data in a data form of type 'result'.

4.4 Accept Pending File Transfer Requests

Listing 13: Local Client Requests to Accept Pending File Transfer Requests on the Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='accept-files-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='http://jabber.org/protocol/rc#accept-files'/>
</iq>
```

Unless an error occurs (see the Error Handling section below), the service SHOULD return the appropriate form.

Listing 14: Remote Client Replies with a Form Containing Pending File Transfers

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='accept-files-1'
    xml:lang='en'>
```

```
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#accept-files'
           sessionid='set-status:20040727T0337Z'
           status='executing'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Pending File Transfers</title>
      <instructions>Select the pending file transfers to accept</
         instructions>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field label='Files'
             type='list-multi'
             var='files'>
        <required/>
        <option label='ballad.ogg (romeo@example.com)'>
          <value>romeo@example.com/orchard:1</value>
        </option>
        <option label='picture.jpg (romeo@example.com)'>
          <value>romeo@example.com/orchard:2</value>
        </option>
        <option label='challenge.txt (mercutio@example.com)'>
          <value>mercutio@example.com/orchard:1</value>
        </option>
      </field>
    </x>
  </command>
</iq>
```

Listing 15: Local Client Submits Form to Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='accept-files-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#accept-files'
           sessionid='accept-files:20040727T0337Z'>
    <x xmlns='jabber:x:data' type='form'>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field type='list-multi' var='files'>
        <value>romeo@example.com/orchard:2</value>
      </field>
    </x>
  </command>
</iq>
```

The remote client accepts the selected file transfers, and informs the local client of completion.

Listing 16: Remote Client Informs Local Client of Completion

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='accept-files-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#accept-files'
           sessionid='accept-files:20040727T0337Z'
           status='completed'/>
</iq>
```

4.5 Leave Groupchats

Listing 17: Local Client Requests the Remote Client to Leave Groupchats

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='leave-groupchats-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='http://jabber.org/protocol/rc#leave-groupchats'/>
</iq>
```

Listing 18: Remote Client Replies with a Form with a List of Groupchats it is currently in

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='leave-groupchats-1'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
             node='http://jabber.org/protocol/rc#leave-groupchats'
             sessionid='leave-groupchats:20040727T0337Z'
             status='executing'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Leave Groupchats</title>
      <instructions>Choose the groupchats you want to leave</
          instructions>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field label='Groupchats'
             type='list-multi'
```

```
            var='groupchats'>
        <required/>
        <option label='juliet on jdev@conference.jabber.org'>
            <value>jdev@conference.jabber.org/juliet</value>
        </option>
        <option label='juliette on jdev@conference.jabber.org'>
            <value>jdev@conference.jabber.org/juliette</value>
        </option>
        <option label='juliet on girlsonly@jabber.com'>
            <value>girlsonly@jabber.com/juliet</value>
        </option>
      </field>
    </x>
  </command>
</iq>
```

Listing 19: Local Client Submits Form to Remote Client

```
<iq from='juliet@example.com/chamber'
    to='juliet@example.com/balcony'
    type='set'
    id='leave-groupchats-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#leave-groupchats'
           sessionid='leave-groupchats:20040727T0337Z'>
    <x xmlns='jabber:x:data' type='form'>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/rc</value>
      </field>
      <field type='list-multi' var='groupchats'>
        <value>jdev@conference.jabber.org/juliet</value>
        <value>girlsonly@jabber.com/juliet</value>
      </field>
    </x>
  </command>
</iq>
```

The remote client leaves the requested groupchats, and informs the local client of completion.

Listing 20: Remote Client Informs Local Client of Completion

```
<iq from='juliet@example.com/balcony'
    to='juliet@example.com/chamber'
    type='result'
    id='leave-groupchats-2'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='http://jabber.org/protocol/rc#leave-groupchats'
```

```
            sessionid='leave-groupchats:20040727T0337Z'
            status='completed'/>
</iq>
```

5 Error Handling

Several error conditions are possible when an entity sends a command request to the service, as defined in the following table. If one of these errors occurs, the service MUST return an error stanza to the requesting entity.

Condition	Cause
<feature-not-implemented/>	The specific command is not supported (even though the ad-hoc commands protocol is)
<forbidden/>	The requesting entity does not have sufficient privileges to perform the command
<service-unavailable/>	The ad-hoc commands protocol is not supported

For the syntax of these errors, see Error Condition Mappings [6]. Naturally, other errors may be returned as well.

6 Implementation Notes

Implementations of this protocol MAY add or remove fields to forms as they see fit. For example, when setting the status of a remote client that supports multiple accounts, the client may choose to add a boolean field to allow the user to specify whether the status change should be applied globally or only to the receiving account.
Implementations MAY also introduce extra forms for commands. For example, when forwarding unread messages, a client could return a form containing a list of short descriptions of unread messages, allowing the user to select the messages he wants to forward.

7 Security Considerations

The ability to complete the tasks specified herein MUST NOT be granted to users who lack privileges to control a client. A sensible access policy is to only allow remote controlling by other resources of the same account used by the client. If other accounts are to be able to remote control the client, the client needs more complex access right management.

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

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [7].

9 XMPP Registrar Considerations

9.1 Protocol Namespaces

The XMPP Registrar includes 'http://jabber.org/protocol/rc' in its registry of protocol namespaces (see <http://xmpp.org/registrar/namespaces.html>).

9.2 Field Standardization

Field Standardization for Data Forms [8] defines a process for standardizing the fields used within Data Forms scoped by a particular namespace (see also <http://xmpp.org/registrar/formtypes.html>). The reserved fields for the 'http://jabber.org/protocol/rc' namespace are specified below.

```
<form_type>
  <name>http://jabber.org/protocol/rc</name>
  <doc>XEP-0146</doc>
  <desc>Forms used for remote controlling clients</desc>
  <field var='auto-auth'
         type='boolean'
         label='Whether to automatically authorize subscription
            requests'/>
  <field var='auto-files'
         type='boolean'
         label='Whether to automatically accept file transfers'/>
  <field var='auto-msg'
         type='boolean'
         label='Whether to automatically open new messages'/>
  <field var='auto-offline'
         type='boolean'
         label='Whether to automatically go offline when idle'/>
  <field var='sounds'
         type='boolean'
         label='Whether to play sounds'/>
  <field var='files'
         type='list-multi'
```

[7] 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/>.

[8] XEP-0068: Field Data Standardization for Data Forms <http://xmpp.org/extensions/xep-0068.html>.

```
        label='A list of pending file transfers'/>
  <field var='groupchats'
        type='list-multi'
        label='A list of joined groupchat rooms'/>
  <field var='status'
        type='list-single'
        label='A presence or availability status'>
    <option label='Chat'>
        <value>chat</value>
    </option>
    <option label='Online'>
        <value>online</value>
    </option>
    <option label='Away'>
        <value>away</value>
    </option>
    <option label='Extended Away'>
        <value>xa</value>
    </option>
    <option label='Do Not Disturb'>
        <value>dnd</value>
    </option>
    <option label='Invisible'>
        <value>invisible</value>
    </option>
    <option label='Offline'>
        <value>offline</value>
    </option>
  </field>
  <field var='status-message'
        type='text-multi'
        label='The status message text'/>
  <field var='status-priority'
        type='text-single'
        label='The new priority for the client'/>
</form_type>
```

10 XML Schema

Because the protocol defined here is a profile of XEP-0050: Ad-Hoc Commands, no schema definition is needed.