XEP-0039: Statistics Gathering

Paul Curtis
mailto:pcurtis@terrapin.com
xmpp:pcurtis@www.terrapin.com

Russell Davis
mailto:ukscone@burninghorse.com
xmpp:ukscone@burninghorse.com

Ryan Eatmon
mailto:reatmon@jabber.org
xmpp:reatmon@jabber.org

David Sutton
mailto:dsutton@legend.co.uk
xmpp:peregrine@legend.net.uk

2002-11-05
Version 0.6.0

Status	Type	Short Name
Deferred	Standards Track	N/A

A protocol to enable gathering statistics from Jabber servers and components.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

As things currently stand it is not possible to obtain statistics from a jabber component or server without resorting to parsing the various log files. This makes it extremely difficult to obtain statistics that are of any use in real world situations. This document attempts to rectify this situation by defining a new namespace that would be used to obtain statistics from a component or server so that they may be manipulated and used in a useful manner. For the purposes of this namespace a statistic is anything that maybe expressed in a numeric form, such as the uptime of a server, the **number** of registered users and the number of packets sent. Things such as a list of currently online users or a list of registered users are beyond the scope of this namespace and properly belong within browse or disco.

2 Description

2.1 Namespace

This is a pretty simple namespace. It consists of a <stat/> tag with three attributes. name, units and value.

```
<query xmlns='http://jabber.org/protocol/stats'>
        <stat name='' units='' value=''/>
      </query>
```

There is one variation in the case of an error invalidating one or more errors in a single returned query that does not actually invalidate the whole query.

```
<query xmlns='http://jabber.org/protocol/stats'>
        <stat name=''><error code=''>...</error></stat>
      </query>
```

2.2 Name Attribute

The name of the statistic. The format for this attribute is the generic statistic type such as bandwidth, users, time etc. followed by a '/' character and then then the name of the actual statistic. For example bandwidth/packets-in, time/uptime and users/online. This will be assigned and administered by JANA[1].

[1] See Name Registration

2.3 Units Attribute

This is the units type of the statistic. As with the name attribute it will be assigned and administered by JANA[2]It is suggested that JANA use where appropriate commonly accepted international standards when assigning unit types i.e. seconds, grams, meters, bytes etc.

2.4 Value Attribute

This is the actual returned value of the queried statistic. The value returned is in multiples of the unit described by the units attribute.

2.5 Query

To query a component or server a client sends an iq packet of the type 'get' to the component or server. The component or server responds with the *list* of statistics that it supports.

```
send:     <iq type='get' to='component'>
            <query xmlns='http://jabber.org/protocol/stats'/>
          </iq>

     recv:     <iq type='result' from='component'>
            <query xmlns='http://jabber.org/protocol/stats'>
              <stat name='time/uptime'/>
              <stat name='users/online'/>
                    .
                    .
                    .
              <stat name='bandwidth/packets-in'/>
              <stat name='bandwidth/packets-out'/>
            </query>
          </iq>
```

Once a client knows which statistics a component or server supports it may now request the actual statistics by sending an iq packet of the type 'get' containing a request for the specific statistics and sending that to the component or server.

```
send:     <iq type='get' to='component'>
            <query xmlns='http://jabber.org/protocol/stats'>
              <stat name='time/uptime'/>
              <stat name='users/online'/>
              <stat name='bandwidth/packets-in'/>
              <stat name='bandwidth/packets-out'/>
            </query>
```

[2]See Name Registration

```
                </iq>


      recv:     <iq type='result' from='component'>
                  <query xmlns='http://jabber.org/protocol/stats'>
                    <stat name='time/uptime' units='seconds'    value=
                        '3054635'/>
                    <stat name='users/online' units='users' value='365
                        '/>
                    <stat name='bandwidth/packets-in' units='packets'
                        value='23434'/>
                    <stat name='bandwidth/packets-out' units='packets'
                         value='23422'/>
                  </query>
                </iq>
```

2.6 Errors

If an error occurs with one or more of the requests for statistics the component or server should return one of the following error codes.

Code	String	Reason
401	Unauthorized	Querying JID is not authorized to perform that query
404	Not Found	The statistic was not found for some reason
501	Not Implemented	Although statistic is advertised as available it has not been implemented
503	Service Unavailable	Statistic is temporarily unavailable

Because we wish to be able to collect groups of statistics within a single returned packet errors must be handled in a two tier way with authorization and core errors that would render **all** the statistics meaningless being indicated with a type='error' in the returned packet.

```
<iq type='error' from='component'>
          <query xmlns='http://jabber.org/protocol/stats'>
            <error code='401'>Not Authorized</error>
          </query>
        </iq>
```

Errors in a query that only invalidate one or more of the requested statistics are indicated with an </error> tag embedded inside the </stat> tag.

```
<iq type='result' from='component'>
          <query xmlns='http://jabber.org/protocol/stats'>
            <stat name='time/uptime units='seconds' value='4534'/>
            <stat name='bandwidth/packets-in'><error code='503'>
    Service Unavailable</error></stat>
            <stat name='bandwidth/packets-out'><error code='503'>
    Service Unavailable</error></stat>
          </query>
        </iq>
```

3 Implementation

3.1 Name Registration

All statistic names, returned data units types and other pertinent statistic information will be assigned and registered with the Jabber Naming Authority in the category ***stat*** Unfortunately at this time such a body does not exist so we will have to rely on component and server authors diligently researching to ensure that their desired name is not already in use and that they adequately document the returned units type and anything else that would normally be registered. Hopefully by the time this document is formally adopted a central naming authority for the Jabber protocol will be in place and functional and authors will be then able to register their names.

Stat	Description	Returned Units
registered name	description of statistic/reason	unit type returned by query

3.2 Core Statistics

Although components and servers are free to support whichever statistics they feel are justified for their particular component or server it is suggested that the following set of three core statistics are implemented by all components and servers.

- <stat name='time/uptime'/>
- <stat name='bandwidth/packets-in'/>
- <stat name='bandwidth/packets-out'/>

Stat	Description	Returned Units
time/uptime	uptime of component or server	Seconds

Stat	Description	Returned Units
bandwidth/packets-in	packets received by component or server	packets
bandwidth/packets-out	packets transmitted by component or server	packets

3.3 Human readable labels

For several reasons the **http://jabber.org/protocol/stats** namespace does not support human readable labels for the returned values. Generally the application querying the statistic should already know what the statistic is and in what units the value is returned. However if the application really wants some form of human readable label for the returned value although not an optimal solution or even recommended by the authors of this document it should be safe for it to convert the value of the units attribute into a string and use that as a label for the returned statistic value.

3.4 Namespace query authorization

In most cases the **http://jabber.org/protocol/stats** would be tied to the component or servers admin JID so that only that JID may query the statistics however there are advantages to having a three tier system where some statistics are available to all JIDs, some to an arbitrary JID listed in the configuration file and all available to the listed admin JID. As the first case can be emulated by the second I propose that when implemented the **http://jabber.org/protocol/stats** namespace is configured to use the three tier method of authorizing queries.

3.5 SNMP

Supporting industry accepted standards and procedures [3] within the Jabber protocol is highly desirable as long as it does not restrict the flexibility or functionality of Jabber. So while the **http://jabber.org/protocol/stats** namespace seems to fall within the domain of the SNMP and CIM standards amongst others and large jabber installations would find direct support an appealing prospect when tying jabber into their existing network information and management systems the jabber:iq:namespace will not do this. Because Jabber is an XML based protocol, conversion of the returned data to other formats including those required for SNMP, CIM etc. should be relatively simple. Not supporting industry standards is not without advantages. By leaving the **http://jabber.org/protocol/stats** as a *pure* jabber namespace we allow ourselves to obtain not only commonly used statistics but also some unusual ones as well. For example imagine a jabber enabled vending machine, by remaining free of the

[3] For more details on SNMP and CIM

encumberence and limitations of SNMP etc. we can directly (if allowed by the controlling component) query the fluid ounces dispensed, most popular beverage and the amount of money that has been collected. Finally although it is unlikely to occur by staying true to our roots and avoiding direct SNMP support we avoid any possibility of conflict with other network management agents such as net(ucd)-snmp.

4 Realworld Examples

TBD

5 DTD

5.1 DTD in English

TBD

5.2 DTD

TBD