XEP-0042: Jabber OOB Broadcast Service (JOBS)

Matthew Miller
mailto:linuxwolf@outer-planes.net
xmpp:linuxwolf@outer-planes.net

2003-04-11
Version 0.5

Status	Type	Short Name
Retracted	Standards Track	JOBS

A protocol for enabling uni-directional multicast data transfers out of band.

Contents

1 Overview

1.1 Introduction

Distributing data out-of-band (OOB) to one or more end-points is a requirement for many Jabber clients. The Jabber OOB Broadcast Service (JOBS) is a mechanism to allow end-points to open uni-directional data streams between each other, on top of which any number of applications can be built [1].

The aim of this document is to define a process of connecting a sender to one or more receivers through a secondary TCP port.

1.2 Design Concepts

As the name implies, JOBS is designed to enable multicast, uni-directional OOB connections. These connections are usually between a ”sender” client, a JOBS ”service”, and one or more ”receiver” clients. Each such set of connections is collectively called a session. JOBS is designed to allow a single service to handle multiple sessions over a single host/port combination.

To address a large number of typical uses efficiently, JOBS can multicast the data from a sender to multiple receivers. In order to keep the protcol as simple as possible, it only allows data to flow in one direction.

JOBS utilizes a ”two-band” authentication mechanism. This allows the end-points to know practically nothing about each other, yet still be assured that the OOB connection is really to/from the Jabber entity its intended to be. The authentication system is then backed-up with explicit authorization requests.

For the OOB portion, clients connect to the host/address and port of the JOBS service for a given session. Once connected, a client-initiated handshake process occurs, and (if successful), then data is routed from the sender’s connection to each receiver’s connection. The only point at which any error information may be conveyed over the OOB connection is during the handshake process.

Term	Description
OOB	Out-Of-Band. Any network connection that exists outside of the normal Jabber protocol traffic.
session	A session registered with a ”server” for clients to connect to.
client	An end-point on a JOBS session.
service	The JOBS Jabber component/server.
server	A particular host/port of the JOBS service.
sender	The client sending data.
receiver	A client receiving data.

[1]Possible applications include file transfer, audio/video streaming, and some gaming implementations.

2 Use Cases

2.1 Discovering the Service

Discovering support for JOBS involves either Jabber Browsing (XEP-0011) [2] or Service Discovery (XEP-0030) [3]. This determines if the end-point understands the JOBS protocol.

2.1.1 ”jabber:iq:browse”

To determine support for JOBS via jabber:iq:browse, look for an item with a nested <ns/> with a value of ”http://jabber.org/protocol/jobs”:

Listing 1: Browse request

```
<iq type='get' to='domain'>
  <query xmlns='jabber:iq:browse'/>
</iq>
```

Listing 2: Browse result

```
<iq type='get' to='sender@domain/res' from='domain'>
  ...
  <item xmlns='jabber:iq:browse'
      category='service'
      type='x-jobs'
      jid='jobs.domain'
      version='0.4'>
    <npath{}
  </item>
  ...
</iq>
```

2.2 Creating a Session

The JOBS protocol supports various scenarios to create sessions. Most of these scenarios allow an entity to determine the possible parameters to create a session with. To actually create a session, the (would-be) sender sends an ”iq-set” with a <session action=”create”/>. This returns the details of the newly created session, including the ID and OOB host/port.
This use-case can be completely ignored for true ”peer-to-peer” systems.

[2] XEP-0011: Jabber Browsing <http://xmpp.org/extensions/xep-0011.html>.
[3] XEP-0030: Service Discovery <http://xmpp.org/extensions/xep-0030.html>.

2.2.1 Simple

The simplest create request is:

Listing 3: Creation request

```
<iq type='set' from='sender@domain/res' to='jobs.domain' id='JOBS1'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='create'/>
</iq>
```

Listing 4: Creation result

```
<iq
    type='result' to='sender@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      status='pending'
      host='jobs.domain'
      id='01234567'
      port='12676'
      sender='sender@domain/res'
      buffer='0'
      expires='30'
      receivers='1'/>
</iq>
```

This creates a session between sender@domain/resource and any one receiver. At this point, the JOBS service is ready to accept connections for this session. The <session/> element describes the details for the session. The value returned in the "id" attribute is the JOBS session ID [4].

2.2.2 With Parameters

When creating a session, parameters to <session/> can be supplied, explicitly requesting that certain parameters be met (such as buffer size, time to expire, and receiver limit). Since these parameters have lower- and upper-bounds specific to the JOBS service, a sender may need to determine these limits.
To determine the limits, the sender sends an "iq-get" with a <session action="create"/>:

Listing 5: Parameter statistics request

```
<iq id='JOBS0' type='get' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs' action='create'/>
</iq>
```

[4]The exact value of the ID is left to JOBS implementations.

Listing 6: Parameter statistics result

```
<iq id='JOBS0' type='result' to='sender@domain/res' from='jobs.domain'
   >
  <session xmlns='http://jabber.org/protocol/jobs'
      host='jobs.domain'
      port='12676'
      sender='sender@domain/res'
      buffer='0'
      expires='30'
      receivers='1'>
    <connect host='jobs.domain' port='12676'/>
    <limit  type='buffer'
        default='0'
        max='1024'
        min='0'/>
    <limit  type='expires'
        default='30'
        max='3600'
        min='5'/>
    <limit  type='receivers'
        default='1'
        max='15'
        min='1'/>
  </session>
</iq>
```

The returned <session/> is also prefilled with default values for all known parameters.
To create a session with specific parameters, the sender sends an "iq-set" as in the "simple" use-case, but then specifying the parameter values desired:

Listing 7: Creation request, with parameters

```
<iq id='JOBS1' type='set' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs' action='create'
      expires='-1'/>
</iq>
```

Listing 8: Creation result, with parameters

```
<iq type='result' to='sender@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      status='pending'
      host='jobs.domain'
      id='01234567'
      port='12676'
      sender='sender@domain/res'
      buffer='0'
      expires='-1'
      receivers='1'/>
```

```
</iq>
```

The above example creates a session that does not timeout. A JOBS service uses values from the default information set for any parameters that are missing.
Any parameters that exceed the minimums/maximums causes an error.

2.2.3 Form-based

In some cases, the session creation process requires an interface more suitable for human consumption. In such cases the JOBS protocol helps by allowing for contained elements governed by other namespaces. For form-based creation, a Data Forms (XEP-0004) [5] form can be embedded in the <session/>.
To create a session using forms, send a <session action="create"/> with an embedded <x xmlns="jabber:x:data"/>:

Listing 9: Creation form request

```
<iq type='get' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs' action='create'>
    <x xmlns='jabber:x:data'/>
  </session>
</iq>
```

Listing 10: Creation form result

```
<iq type='result' from='jobs.domain' to='sender@domain/res'>
  <session xmlns='http://jabber.org/protocol/jobs'
      host='jobs.domain'
      port='12676'
      buffer='0'
      expires='-1'
      receivers='1'>
    <x xmlns='jabber:x:data' type='form'>
      <instructions>Please specify values for the given fields.</
          instructions>
      <field var='hostport' type='select-single' label='JOBS␣host/port
          '>
        <option>jobs.domain:12676</option>
      </field>
      <field var='buffer' type='text' label='Buffer␣Size␣(in␣bytes)'><
          value>0</value></field>
      <field var='expires' type='text' label='Timeout␣(in␣seconds)'><
          value>30</value></field>
      <field var='receivers' type='text' label='Number␣of␣Recipients'>
          <value>1</value></field>
```

[5] XEP-0004: Data Forms <http://xmpp.org/extensions/xep-0004.html>.

```
    </x>
  </session>
</iq>
```

The exact fields present in the form are dependent upon the JOBS implementation. The form SHOULD allow a user to at least specify the <session/> attributes.
Using the form-based approach, the session is then created by sending a <session action='create'/> with a form submission (as defined for "jabber:x:data"):

Listing 11: Creation request (form-based)

```
<iq type='set' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs' action='create'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='hostport'><value>jobs.domain:12676</value></field>
      <field var='buffer'><value>0</value></field>
      <field var='expires'><value>300</value></field>
      <field var='receivers'><value>1</value></field>
    </x>
  </session>
</iq>
```

Listing 12: Creation result (form-based)

```
<iq type='result' from='jobs.domain' to='sender@domain/res'>
  <session xmlns='http://jabber.org/protocol/jobs'
      status='pending'
      host='jobs.domain'
      id='01234567'
      port='12676'
      sender='sender@domain/res'
      buffer='0'
      expires='300'
      receivers='1'/>
</iq>
```

2.3 Inviting Receivers

Once the session is created, the sender invites receivers to connect. The sender can invite receivers either directly, or via the JOBS service. Most invitations are distributed via <message/>.

2.3.1 Inviting Directly

The sender can invite receivers directly. This is done using a <message/>:

Listing 13: Invitation message (direct)

```
<message from='sender@domain/res' to='receiver@domain/res'>
  <body>Let's␣connect!</body>
␣␣<session␣xmlns='http://jabber.org/protocol/jobs'
␣␣␣␣␣␣host='jobs.domain'
␣␣␣␣␣␣id='01234567'
␣␣␣␣␣␣port='12676'
␣␣␣␣␣␣sender='sender@domain/res'
␣␣␣␣␣␣buffer='0'
␣␣␣␣␣␣expires='30'
␣␣␣␣␣␣receivers='1'/>
</message>
```

When inviting directly, the <session/> MUST contain enough information for a receiver to connect OOB. The required information is:

- host
- id
- port

2.3.2 Inviting via JOBS

Alternatively, a sender can invite receivers via the JOBS service. This is also done using a <message/>, with a <session action="notify"/> containing one or more <item action="invite" type="connection"/>:

Listing 14: Invitation message (to JOBS service)

```
<message from='sender@domain/res' to='jobs.domain'>
  <body>Let's␣connect!</body>
␣␣<session␣xmlns='http://jabber.org/protocol/jobs'
␣␣␣␣␣␣action='notify'
␣␣␣␣␣␣id='01234567'>
␣␣␣␣<item␣type='connection'␣action='invite'>receiver@domain</item>
</message>
```

This results in the JOBS service sending the <message/> to each <item/>. Any additional elements (such as a <body/>) are passed onto those invited:

Listing 15: Invitation message (from JOBS service)

```
<message from='jobs.domain' to='receiver@domain'>
  <body>Let's␣connect!</body>
␣␣<session␣xmlns='http://jabber.org/protocol/jobs'
␣␣␣␣␣␣action='notify'
```

```
␣␣␣␣␣␣status='pending'
␣␣␣␣␣␣host='jobs.domain'
␣␣␣␣␣␣id='01234567'
␣␣␣␣␣␣port='12676'
␣␣␣␣␣␣sender='sender@domain/res'
␣␣␣␣␣␣buffer='0'
␣␣␣␣␣␣expires='30'
␣␣␣␣␣␣receivers='1'>
␣␣␣␣<item␣type='connection'␣action='invite'/>
</message>
```

2.4 Retrieving Session Information

At any time, a client can request information about sessions for a JOBS service. The request can be directed for "all" sessions, or a specific session[6].

2.4.1 Service-wide

A client can request all the sessions for a JOBS service by sending an "iq-get" containing a <session action="info"/> with no ID:

Listing 16: Information request (all sessions)

```
<iq type='get' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='info'/>
</iq>
```

The JOBS service responds with all the sessions within the "iq-result". This is the only case where a result can have more than one <session/>.

Listing 17: Information result (all sessions)

```
<iq type='result' from='jobs.domain' to='sender@domain/res'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='info'
      status='active'
      host='jobs.domain'
      id='01234567'
      port='12676'
      buffer='0'
      expires='30'
      receivers='1'>
    <item type='connection' action='accept'>sender@domain/res</item>
```

[6]Who is allowed to perform this action is left up to the JOBS service implementation.

```
  </session>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='info'
      status='pending'
      host='jobs.domain'
      id='87654321'
      port='12676'
      buffer='0'
      expires='30'
      receivers='1'/>
</iq>
```

2.4.2 Session-specific

Alternatively, a client can request the information for a specific session by sending an ”iq-get” containing a <session action=”info”/> with the ID:

Listing 18: Information request (session-specific)

```
<iq type='get' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='info'
      id='01234567'/>
</iq>
```

The service responds with an ”iq-result” of just the requested session.

Listing 19: Information result (session-specific)

```
<iq type='result' from='jobs.domain' sender='sender@domain/res'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='info'
      status='active'
      host='jobs.domain'
      id='01234567'
      port='12676'
      buffer='0'
      expires='30'
      receivers='1'>
    <item type='connection' action='accept'>sender@domain/res</item>
  </session>
</iq>
```

2.5 Connecting OOB

2.5.1 Initiating and Authenticating

When a client connects (sender or receivers), a client-initiated handshake takes place. The purpose of this handshake is to authenticate the OOB connection, in relation to the client's JID. This authentication utilizes both in-band and OOB packets.
To start the handshake, the client sends an "init" packet on its established connection:

Listing 20: Client INIT

```
jobs/0.4 init
session-id: 01234567
client-jid: sender@domain/res
```

If the session exists, and the client's JID is not automatically rejected, the JOBS service responds with an auth-challenge packet, containing an unique, arbitrary token:

Listing 21: Server AUTH-CHALLENGE

```
jobs/0.4 auth-challenge
confirm: SID00001234
```

Once received, the client then sends an "iq-set" containing a <session action="authenticate"/>, which itself contains an <item type='auth' action='confirm'/> with this confirm key:

Listing 22: Authentication request (from Client)

```
<iq type='set' to='jobs.domain'>
    <session xmlns='http://jabber.org/protocol/jobs'
        action='authenticate'
        id='01234567'>
      <item type='auth' action='confirm'>SID00001234</item>
    </session>
  </iq>hehe
```

The service then compares this confirm key to that sent with the "auth-challenge" OOB packet. If this matches correctly, and the service determines this connection is authorized, the session will respond with a <session action="authenticate"/> containing a <item type="auth" action="accept"/> with the accept key:

Listing 23: Authentication result (from Server)

```
<iq type='result' from='jobs.domain' to='sender@domain/res'>
    <session xmlns='http://jabber.org/protocol/jobs'
        action='authenticate'
        status='pending'
        id='01234567'>
```

```
      <item type='auth' action='accept'>SID88884321</accept>
    </session>
  </iq>
```

At this point, the client responds on the OOB data stream with an ”auth-response” packet:

Listing 24: Client AUTH-RESPONSE

```
jobs/0.4 auth-response
  accept: SID88884321
```

If the connection is accepted, the JOBS service sends a ”connected” packet:

Listing 25: Server CONNECTED

```
jobs/0.4 connected
```

and after this, the data transfer occurs. If this connection is the sender, they may start sending data now (regardless if receivers are connected). If this connection is a receiver, the sender's data immediately follows the terminating ”newline”.

2.5.2 Authorizing

Authenticating ensures the OOB connection matches a particular JID. Authorizing ensures to the service that receiver is allowed to be connected to the session. To determine if the session connection should be accepted or rejected, the JOBS service first checks if the JID matches the sender. This matches against the ”full” JID, including node, domain, and resource. If this connection is the sender, it is allowed. Otherwise, the service confirms the connection with the sender.
If a confirmation is required, the service sends an ”iq-get” to the sender, with a <session action=”authorize”/> containing an <item type”connection” action=”confirm”/> with the full JID of the receiver:

Listing 26: Confirmation request

```
<iq type='get' to='sender@domain/res' id='JOBS5' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='authorize'
      status='active'
      id='01234567'>
    <item type='connection' action='confirm'>receiver@domain/res</item
       >
  </session>
</iq>
```

One or more <item type="connection" action="confirm/> elements, each specifying a JID to accept/reject. To accept (or reject) a connection, the sender responds with an "iq-result", wrapping each JID in either an <item type="connection" action="accept"/> or <item type="connection" action="reject"/>.

Listing 27: Confirmation result (accept)

```
<iq id='JOBS5' type='result' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='authorize'
      id='01234567'>
    <item type='connection' action='accept'>receiver@domain/res</item>
  </session>
</iq>
```

Listing 28: Confirmation result (reject)

```
<iq id='JOBS5' type='result' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='authorize'
      id='01234567'>
    <item type='connection' action='reject'>receiver@domain/res</item>
  </session>
</iq>
```

If the connection is rejected, the service drops the connection, and notifies the sender and receiver of the dropped connection.

2.5.3 Dropping

The sender may drop a connection at any time. To drop a connection, the sender sends an "iq-set" with the <session/> containing the "connection" to drop:

Listing 29: Drop request

```
<iq type='set' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='notify'
      id='01234567'>
    <item type='connection' action='drop'>receiver@domain/res</item>
  </session>
</iq>
```

If the connection is successfully dropped, the service returns an "iq-result":

Listing 30: Drop result

```
<iq type='result' from='jobs.domain' to='sender@domain/res'>
  <session  xmlns='http://jabber.org/protocol/jobs'
      status='active'
      id='01234567'/>
</iq>
```

The service also sends notification messages to the sender and the JID of the dropped connection (detailed in the "Being Notified about Events" section).

2.6 Deleting a Session

Sessions are deleted either by timeout or explicitly. Sessions are deleted by timeout automatically under certain conditions. Sessions can also be deleted explicity by their senders, at any time. Regardless of the method of deletion, a notice is sent to all connected.
This use-case can be completely ignored for true "peer-to-peer" systems.

2.6.1 Expiring

The exact conditions that expire a session are mostly up to the implementation. At a minimum, a session SHOULD be expired when there are less than two connections, and the "expires" time is reached.

2.6.2 Deleting Explicitly

To explictly delete a session, the sender sends an "iq-set" containing a <session action="delete"/>:

Listing 31: Delete request

```
<iq id='JOBS50' type='set' to='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='delete'/>
</iq>
```

Listing 32: Delete result

```
<iq id='JOBS50' type='result' to='sender@domain/res' from='jobs.domain
   '>
  <session xmlns='http://jabber.org/protocol/jobs'
      status='closed'
      id='01234567'/>
</iq>
```

2.7 Being Notified about Events

2.7.1 Connection Accepted

When a connection is accepted, the service sends a "notify" message to the sender and (if appropriate) the accepted receiver, with a <item type='connection' action='accept'/>:

Listing 33: "Connection Accepted" message

```
<message to='sender@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='notify'
      status='pending'
      id='01234567'>
    <item type='connection' action='accept'/>
  </session>
</message>
```

If the notification is not about the recipient of the message, then the <item/> contains the JID this notification pertains to.

2.7.2 Connection Rejected

When a connection is rejected, the service sends a "notify" message to the sender and (if appropriate) the accepted receiver, with a <item type='connection' action='reject'/>:

Listing 34: "Connection Rejected" message

```
<message to='receiver@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='notify'
      status='pending'
      id='01234567'>
    <item type='connection' action='reject'/>
  </session>
</message>
```

If the notification is not about the recipient of the message, then the <item/> contains the JID this notification pertains to.

2.7.3 Connection Dropped

When a connection is dropped, the service sends a "notify" message to the sender and (if appropriate) the accepted receiver, with a <item type='connection' action='drop'/>:

Listing 35: "Connection Dropped" message

```
<message to='receiver@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='notify'
      status='pending'
      id='01234567'>
    <item type='connection' action='drop'/>
  </session>
</message>
```

If the notification is not about the recipient of the message, then the <item/> contains the JID this notification pertains to.

2.7.4 Session Deleted

When a session is deleted, any clients connected to the session are immediately disconnected. The "notify" message is sent to the sender and any receivers still connected, with the <session action="notify"/> containing an <item type="status"/>:

Listing 36: "Session Deleted (explicitly)" message

```
<message to='sender@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='notify'
      status='closed'
      id='01234567'>
    <item type='status' action='delete'/>
  </session>
</message>
```

Listing 37: "Session Deleted (timed out)" message

```
<message to='sender@domain/res' from='jobs.domain'>
  <session xmlns='http://jabber.org/protocol/jobs'
      action='notify'
      status='closed'
      id='01234567'>
    <item type='status' action='expire'/>
  </session>
</message>
```

The reason the session is deleted is specified by the action attribute. A value of "delete" means it was explicitly deleted. A value of "expire" means it timed out.

3 Formal Description

3.1 In-Band Protocol

3.1.1 DTD

```
<!ELEMENT session ( ( item | ( connect? , limit+ ) )* | #PCDATA* ) >
<!ATTLIST session
    action   ( 'authenticate' | 'authorize' | 'create' | 'delete' | '
       info' | 'notify' )  #OPTIONAL
    status   ( 'active' | 'closed' | 'in-use' | 'pending' )
                                    #OPTIONAL
    host      CDATA #OPTIONAL
    id        CDATA #OPTIONAL
    port      CDATA #OPTIONAL
    sender    CDATA #OPTIONAL
    buffer    CDATA #OPTIONAL
    expires   CDATA #OPTIONAL
    receivers CDATA #OPTIONAL
>

<!ELEMENT item ( #PCDATA )* >
<!ATTLIST item
    action   ( 'accept' | 'confirm' | 'delete' | 'drop' | 'expire' | '
       invite' | 'reject')  #OPTIONAL
    type     ( 'auth' | 'connection' | 'status' )
                                                #OPTIONAL
>

<!ELEMENT connect () >
<!ATTLIST connect
    host  CDATA #REQUIRED
    port  CDATA #REQUIRED
>

<!ELEMENT limit () >
<!ATTLIST limit
    default CDATA #REQUIRED
    max     CDATA #REQUIRED
    min     CDATA #REQUIRED
>
```

3.1.2 <session/> Element

The <session/> element is the core element to the protocol. This element provides both information about a session and the action applied to it. It has a large number of attributes, and contains zero or more <item/> elements, zero or more <connect/> elements, and zero or three <limit/> elements. It may also contain elements governed by other namespaces.

The ”action” attribute specifies the action to apply or being applied to the session. From clients, this attribute MUST be specified. From the service this attribute MAY be specified (to prevent ambiguity). The value of ”action” MUST be one of the following:

Value	Description
authenticate	Authenticating one or more connections.
authorize	Authorizing one or more connections.
create	Create a new session.
delete	Delete an existing session.
notify	Notification about the session.

The ”status” attribute specifies the current status of the session. This attribute MUST NOT be present if the session does not have an identifier (i.e. does not yet exist). Only the service can provide this attribute. The value of ”status” MUST be one of the following:

Value	Description
active	The session is active, but not yet in use.
closed	The session has closed.
in-use	The session is in use (e.g. data is being transferred).
pending	The session is ready, but not yet active (e.g. not enough connections).

The ”host” attribute specifies the OOB hostname for the session. This attribute SHOULD be specified when possible. The value of this attribute can either be the ”raw” dotted-decimal address or a fully-qualified domain name.
The ”id” attribute identifies the session. This attribute is required for all uses of <session/> except the request to create a session. This value is any string that the service and clients can use to uniquely identify it.
The ”port” attribute specifies the OOB port number for the session. This attribute SHOULD be specified when possible.
The ”sender” attribute specifies the JID of the sender. This attribute SHOULD be specified when possible. The value of this attribute MUST be the full JID of the sender, including node and resource (if possible).
The ”buffer” attribute specifies the size of a temporary transfer buffer. This attribute MAY be present at any time, and SHOULD be presented by the service wherever possible. The value of this attribute MUST be a non-negative number. A value of 0 means there is no buffer. This value has limits defined by the ”jobs:buffer” parameter statistic.
The ”expires” attribute specifies the number of seconds before this session times out. This attribute MAY be present at any time, and SHOULD be presented by the service wherever

possible. The value of this attribute MUST be either a positive number or -1. A value of -1 means this session does not expire. This value has limits defined by the ”jobs:expires” parameter statistic.
The ”receivers” attribute specifies the maximum number of receivers this session can have. this attribute MAY be present at any time, and SHOULD be presented by the service wherever possible. The value of this attribute MUST be either a positive number of -1. A value of -1 means this session can (theoretically) have any number of receivers. This value has limits defined by the ”jobs:receivers” parameter statistic.

3.1.3 <item/> Element

The <item/> element is used for detailed information about specific items of a session. It is used to contain authentication keys, to define connections, and provide more detailed status for a session. It has attributes for the type of item and the action associated with this item. This element contains only character data.
The ”action” attribute specifies the action to apply or being applied to this item. From clients, this attribute SHOULD be specified. From the service, this attribute MUST be specified (to prevent ambiguity). The value of ”action” MUST be one of the following:

Value	Description	Notes
accept	The item is accepted.	This value MUST only be used when the type is ”auth” or ”connection”.
confirm	The item needs confirmation.	This value MUST only be used when the type is ”auth” or ”connection”.
delete	The item is deleted.	This value MUST only be used when the type is ”status”.
drop	The item is dropped.	This value MUST only be used when the type is ”connection”.
expire	The item has expired.	This value MUST only be used when the type is ”status”.
invite	The item is invited to the session.	This value MUST only be used when the type is ”connection”.
reject	The item is rejected.	This value MUST only be used when the type is ”auth” or ”connection”.

The ”type” attribute specifies the type of item. This attribute MUST be present. The value of ”type” MUST be one of the following:

Value	Description
auth	The item pertains to authentication keys.

Value	Description
connection	The item details a session connection. The CDATA is the JID that is connected.
status	The item details a session status event.

3.1.4 <connect/> Element

The <connect/> element specifies a valid host/port combination for a session. An instance of this element MUST be present for each host/port combination possible. This element SHOULD only be present when information on creating sessions is requested. It has attributes to define the OOB hostname and port number. This element is empty.
The ”host” attribute specifies the OOB hostname. This attribute MUST be present. The value is either the ”raw” dotted-decimal IP address, or the fully-qualified domain name.
The ”port” attribute specifies the OOB port number. This attribute MUST be present. The value MUST be a positive integer in the range (0 < port <= 1024).

3.1.5 <limit/> Element

The <limit/> element specifies a valid host/port combination for a session. An instance of this element MUST be present for each ”type”. This element SHOULD only be present when information on creating sessions is requested. It has attributes to define the type of limit, the default value, the minimum value, and the maximum value. This element is empty.
The ”type” attribute specifies the type of limit. This attribute MUST be present. Each type corresponds to an attribute of <session/>. The value of ”type” MUST be one of the following:

Value	Description
buffer	The buffer size limits. The units for ”default”, ”max”, and ”min” are bytes.
expires	The expires time limits. The units for ”default”, ”max”, and ”min” are seconds.
receivers	The receiver count limits. The units for ”default”, ”max”, and ”min” are number of connections.

The ”default” attribute specifies the default value for this limit. This attribute MUST be present. The value of ”default” MUST be a number.
The ”max” attribute specifies the maximum value for this limit. This attribute MUST be present. The value of ”max” MUST be a number. A value of -1 means there is no maximum value.
The ”min” attribute specifies the minimum value for this limit. This attribute MUST be present. The value of ”min” MUST be a number. A value of -1 means there is no minimum value.

3.1.6 Possible Errors

Code	Message	Cause
400	Bad Request	The JOBS service did not understand the request.
403	Forbidden	The JOBS service cannot accept the authentication request from the requesting JID.
404	Not Found	The JOBS service could not find the given session.
406	Not Acceptable	The authentication is not valid.

Code	Message	Cause
400	Bad Request	The JOBS service did not understand the request.
403	Forbidden	The JOBS service cannot accept the authorization request from the requesting JID.
404	Not Found	The JOBS service could not find the given session.
406	Not Acceptable	The authorization is not valid.

Code	Message	Cause
400	Bad Request	The JOBS service did not understand the request.
403	Forbidden	The JOBS service cannot accept any creation requests from this JID.
406	Server Not Acceptable	The JOBS service cannot accept any creation requests using the requested <server/> parameters.
406	Restrictions Not Acceptable	The JOBS service cannot accept any creation requests using the requested <accept/>, <confirm/>, and/or <reject/> parameters.
503	Service Unavailable	The JOBS service cannot accept any additional sessions at this time. Future requests may be accepted.

Code	Message	Cause
400	Bad Request	The JOBS service did not understand the request.
403	Forbidden	The JOBS service cannot accept deletion requests from the requesting JID.
404	Not Found	The JOBS service could not find a given session.

Code	Message	Cause
400	Bad Request	The JOBS service did not understand the request.
403	Forbidden	The JOBS service denied the connection for any reason.
404	Not Found	The JOBS service could not find a given connection and/or JID.
406	Not Acceptable	The JOBS service denied the notify for some reason.
504	Remote Server Timeout	The JOBS connection timed out.

3.2 OOB Protocol

The OOB protocol consists of a series of hanshaking headers, then the normal data transfer process. The syntax of the hanshake packets is similar to HTTP and SIP, in that it includes a "version and method" line, followed by zero or more "headers". The end of a packet is marked by two adjacent carriage returns (i.e. a single "empty" line). The primary difference is with the first line ("version and method"), where the protocol name and version precede the method.

3.2.1 Header EBNF

```
jobs-oob :=          header CRLF ?(BINDATA)
header :=            command-line CRLF *( header-field CRLF )

command-line :=      jobs-version-tag SP method

jobs-version-tag := "jobs" "/" *DIGIT 1*( "." *DIGIT )
method :=            "init" | "auth-challenge" | "auth-response" | "
   connected" | "error"

header-field :=      header-name ":" *1(SP header-value)
```

```
header-name :=       NMCHAR *1( ( "." | "-" | NMCHAR ) NMCHAR)
header-value :=      TEXT

name-char :=         UALPHA | LALPHA | DIGIT

CRLF :=              CR LF

TEXT :=              <any OCTECT except CTLs>
OCTET :=             <any 8-bit sequence of data>
CTL :=               <any US-ASCII control character (octets 0 - 31)
    and DEL (127)>
UALPHA :=            <any US-ASCII uppercase letter "A".."Z">
LALPHA :=            <any US-ASCII lowercase letter "a".."z">
DIGIT :=             <any US-ASCII digit "0".."9">
SP :=                <US-ASCII SP, space (32)>
CR :=                <US-ASCII CR, carriage return (13)>
LF :=                <US-ASCII LF, linefeed (10)>
BINDATA :=           <Arbitrary Binary Data>
```

3.2.2 State Details

Comand-line (example)	jobs/0.4 init
Expected header-fields	"session-id" - The JOBS session ID "client-jid" - The (full) client JID

Comand-line (example)	jobs/0.4 auth-challenge
Expected header-fields	confirm - A unique <item type="auth" action="confirm"/> token

Comand-line (example)	jobs/0.4 auth-response
Expected header-fields	accept - A unique <item type="auth" action="accept"/> token

Comand-line (example)	jobs/0.4 connected
Expected header-fields	NONE

Comand-line (example)	jobs/0.4 error
Expected header-fields	error-code - The HTTP-like error code error-msg - Detailed error message

3.2.3 Possible Errors