ejabberd 1.1.4

Installation and Operation Guide

September 5, 2007

ejabberd Development Team

Contents

1 Introduction

`ejabberd` is a free and open source instant messaging server written in Erlang[1].

`ejabberd` is cross-platform, distributed, fault-tolerant, and based on open standards to achieve real-time communication.

`ejabberd` is designed to be a rock-solid and feature rich XMPP server.

`ejabberd` is suitable for small deployments, whether they need to be scalable or not, as well as extremely big deployments.

[1] `http://www.erlang.org/`

1.1 Key Features

`ejabberd` is:

- Cross-platform: `ejabberd` runs under Microsoft Windows and Unix derived systems such as Linux, FreeBSD and NetBSD.
- Distributed: You can run `ejabberd` on a cluster of machines and all of them will serve the same Jabber domain(s). When you need more capacity you can simply add a new cheap node to your cluster. Accordingly, you do not need to buy an expensive high-end machine to support tens of thousands concurrent users.
- Fault-tolerant: You can deploy an `ejabberd` cluster so that all the information required for a properly working service will be replicated permanently on all nodes. This means that if one of the nodes crashes, the others will continue working without disruption. In addition, nodes also can be added or replaced 'on the fly'.
- Administrator Friendly: `ejabberd` is built on top of the Open Source Erlang. As a result you do not need to install an external database, an external web server, amongst others because everything is already included, and ready to run out of the box. Other administrator benefits include:
  - Comprehensive documentation.
  - Straightforward installers for Linux, Mac OS X, and Windows. IMPROVED
  - Web interface for administration tasks.
  - Shared Roster Groups.
  - Command line administration tool. IMPROVED
  - Can integrate with existing authentication mechanisms.
  - Capability to send announce messages.
- Internationalized: `ejabberd` leads in internationalization. Hence it is very well suited in a globalized world. Related features are:
  - Translated in 12 languages. IMPROVED
  - Support for IDNA[2].
- Open Standards: `ejabberd` is the first Open Source Jabber server claiming to fully comply to the XMPP standard.
  - Fully XMPP compliant.
  - XML-based protocol.
  - Many JEPs supported[3].

[2] http://www.ietf.org/rfc/rfc3490.txt
[3] http://ejabberd.jabber.ru/protocols

1.2 Additional Features

Moreover, `ejabberd` comes with a wide range of other state-of-the-art features:

- Modular
  - Load only the modules you want.
  - Extend `ejabberd` with your own custom modules.
- Security
  - SASL and STARTTLS for c2s and s2s connections.
  - STARTTLS and Dialback s2s connections.
  - Web interface accessible via HTTPS secure access.
- Databases
  - Native MySQL support.
  - Native PostgreSQL support.
  - Mnesia.
  - ODBC data storage support.
  - Microsoft SQL Server support. NEW
- Authentication
  - LDAP and ODBC. IMPROVED
  - External Authentication script.
  - Internal Authentication.
- Others
  - Compressing XML streams with Stream Compression (JEP-0138[4]).
  - Interface with networks such as AIM, ICQ and MSN.
  - Statistics via Statistics Gathering (JEP-0039[5]).
  - IPv6 support both for c2s and s2s connections.
  - Multi-User Chat[6] module with logging. IMPROVED
  - Users Directory based on users vCards.
  - Publish-Subscribe[7] component.
  - Support for virtual hosting.
  - HTTP Polling[8] service.
  - IRC transport.

[4] http://www.jabber.org/jeps/jep-0138.html
[5] http://www.jabber.org/jeps/jep-0039.html
[6] http://www.jabber.org/jeps/jep-0045.html
[7] http://www.jabber.org/jeps/jep-0060.html
[8] http://www.jabber.org/jeps/jep-0025.html

2 Installation from Source

2.1 Installation Requirements

2.1.1 'Unix-like' operating systems

To compile `ejabberd` on a 'Unix-like' operating system, you need:

- GNU Make
- GCC
- libexpat 1.95 or higher
- Erlang/OTP R9C-2 or higher
- OpenSSL 0.9.6 or higher (optional)
- Zlib 1.2.3 or higher (optional)
- GNU Iconv 1.8 or higher (optional, not needed on systems with GNU libc)

2.1.2 Windows

To compile `ejabberd` on a Windows flavour, you need:

- MS Visual C++ 6.0 Compiler
- Erlang/OTP R9C-2 or higher[9]
- Expat 1.95.7 or higher[10]
- GNU Iconv 1.9.1[11] (optional)
- Shining Light OpenSSL[12] (to enable SSL connections)
- Zlib 1.2.3 or higher[13]

2.2 Obtaining `ejabberd`

Released versions of `ejabberd` can be obtained from
http://www.process-one.net/en/projects/ejabberd/download.html.

The latest development version can be retrieved from the Subversion repository.

```
svn co http://svn.process-one.net/ejabberd/trunk ejabberd
```

[9] http://erlang.org/download.html
[10] http://sourceforge.net/project/showfiles.php?group_id=10127&package_id=11277
[11] http://www.gnu.org/software/libiconv/
[12] http://www.slproweb.com/products/Win32OpenSSL.html
[13] http://www.zlib.net/

2.3 Compilation

2.3.1 'Unix-like' operating systems

Compile `ejabberd` on a 'Unix-like' operating system by executing:

```
./configure
make
su
make install
```

These commands will:

- install `ejabberd` into the directory `/var/lib/ejabberd`,
- install the configuration file into `/etc/ejabberd`,
- create a directory called `/var/log/ejabberd` to store log files.

Note: if you want to use an external database, you need to execute the configure script with the option(s) `--enable-odbc` or `--enable-odbc --enable-mssql`. See section 4 for more information.

2.3.2 Windows

- Install Erlang emulator (for example, into `C:\Program Files\erl5.3`).
- Install Expat library into `C:\Program Files\Expat-1.95.7` directory.

  Copy file `C:\Program Files\Expat-1.95.7\Libs\libexpat.dll` to your Windows system directory (for example, `C:\WINNT` or `C:\WINNT\System32`)
- Build and install the Iconv library into the directory `C:\Program Files\iconv-1.9.1`.

  Copy file `C:\Program Files\iconv-1.9.1\bin\iconv.dll` to your Windows system directory (more installation instructions can be found in the file README.woe32 in the iconv distribution).

  Note: instead of copying libexpat.dll and iconv.dll to the Windows directory, you can add the directories `C:\Program Files\Expat-1.95.7\Libs` and `C:\Program Files\iconv-1.9.1\bin` to the `PATH` environment variable.
- While in the directory `ejabberd\src` run:

  ```
  configure.bat
  nmake -f Makefile.win32
  ```
- Edit the file `ejabberd\src\ejabberd.cfg` and run

  ```
  werl -s ejabberd -name ejabberd
  ```

2.4 Starting

Execute the following command to start `ejabberd`:

```
erl -pa /var/lib/ejabberd/ebin -name ejabberd -s ejabberd
```

or

```
erl -pa /var/lib/ejabberd/ebin -sname ejabberd -s ejabberd
```

In the latter case the Erlang node will be identified using only the first part of the host name, i. e. other Erlang nodes outside this domain cannot contact this node.

Note that when using the above command, `ejabberd` will search for the configuration file in the current directory and will use the current directory for storing its user database and for logging.

To specify the path to the configuration file, the log files and the Mnesia database directory, you may use the following command:

```
erl -pa /var/lib/ejabberd/ebin \
    -sname ejabberd \
    -s ejabberd \
    -ejabberd config \"/etc/ejabberd/ejabberd.cfg\" \
              log_path \"/var/log/ejabberd/ejabberd.log\" \
    -sasl sasl_error_logger \{file,\"/var/log/ejabberd/sasl.log\"\} \
    -mnesia dir \"/var/lib/ejabberd/spool\"
```

You can find other useful options in the Erlang manual page (`erl -man erl`).

To use more than 1024 connections, you should set the environment variable `ERL_MAX_PORTS`:

```
export ERL_MAX_PORTS=32000
```

Note that with this value, `ejabberd` will use more memory (approximately 6 MB more).

To reduce memory usage, you may set the environment variable `ERL_FULLSWEEP_AFTER`:

```
export ERL_FULLSWEEP_AFTER=0
```

But in this case `ejabberd` can start to work slower.

3 Basic Configuration

The configuration file will be loaded the first time you start `ejabberd`. The content from this file will be parsed and stored in a database. Subsequently the configuration will be loaded from the database and any commands in the configuration file are appended to the entries in the database. The configuration file contains a sequence of Erlang terms. Lines beginning with a `'%'` sign are ignored. Each term is a tuple of which the first element is the name of an option, and any further elements are that option's values. If the configuration file do not contain for instance the 'hosts' option, the old host name(s) stored in the database will be used.

You can override the old values stored in the database by adding next lines to the configuration file:

```
override_global.
override_local.
override_acls.
```

With these lines the old global options, local options and ACLs will be removed before new ones are added.

3.1 Host Names

The option `hosts` defines a list containing one or more domains that `ejabberd` will serve.

Examples:

- Serving one domain:

  ```
  {hosts, ["example.org"]}.
  ```

- Serving one domain, and backwards compatible with older `ejabberd` versions:

  ```
  {host, "example.org"}.
  ```

- Serving two domains:

  ```
  {hosts, ["example.net", "example.com"]}.
  ```

3.2 Virtual Hosting

Options can be defined separately for every virtual host using the `host_config` option. It has the following syntax:

```
{host_config, <hostname>, [<option>, <option>, ...]}.
```

Examples:

- Domain `example.net` is using the internal authentication method while domain `example.com` is using the LDAP server running on the domain `localhost` to perform authentication:

```
{host_config, "example.net", [{auth_method, internal}]}.

{host_config, "example.com", [{auth_method, ldap},
                              {ldap_servers, ["localhost"]},
                              {ldap_uidattr, "uid"},
                              {ldap_rootdn, "dc=localdomain"},
                              {ldap_rootdn, "dc=example,dc=com"},
                              {ldap_password, ""}]}.
```

- Domain `example.net` is using ODBC to perform authentication while domain `example.com` is using the LDAP servers running on the domains `localhost` and `otherhost`:

```
{host_config, "example.net", [{auth_method, odbc},
                              {odbc_server, "DSN=ejabberd;UID=ejabberd;PWD=ejabberd"}]}.

{host_config, "example.com", [{auth_method, ldap},
                              {ldap_servers, ["localhost", "otherhost"]},
                              {ldap_uidattr, "uid"},
                              {ldap_rootdn, "dc=localdomain"},
                              {ldap_rootdn, "dc=example,dc=com"},
                              {ldap_password, ""}]}.
```

3.3 Listened Sockets

The option `listen` defines for which addresses and ports `ejabberd` will listen and what services will be run on them. Each element of the list is a tuple with the following elements:

- Port number.
- Module that serves this port.
- Options to this module.

Currently next modules are implemented:

ejabberd_c2s	Description	Handles c2s connections.
	Options	access, certfile, inet6, ip, max_stanza_size, shaper, ssl, tls, starttls, starttls_required, zlib
ejabberd_s2s_in	Description	Handles incoming s2s connections.
	Options	inet6, ip, max_stanza_size
ejabberd_service	Description	Interacts with external components (*).
	Options	access, hosts, inet6, ip, shaper
ejabberd_http	Description	Handles incoming HTTP connections.
	Options	certfile, http_poll, inet6, ip, tls, web_admin

(*) The mechanism for external components[14] is defined in Jabber Component Protocol (JEP-0114[15]).

The following options are available:

`{access, <access rule>}` This option defines access to the port. The default value is `all`.

`{certfile, Path}` Path to a file containing the SSL certificate.

`{hosts, [Hostnames], [HostOptions]}` This option defines one or more hostnames of connected services and enables you to specify additional options including `{password, Secret}`.

`http_poll` This option enables HTTP Polling (JEP-0025[16]) support. HTTP Polling enables access via HTTP requests to `ejabberd` from behind firewalls which do not allow outgoing sockets on port 5222.

If HTTP Polling is enabled, it will be available at `http://server:port/http-poll/`. Be aware that support for HTTP Polling is also needed in the Jabber client. Remark also that HTTP Polling can be interesting to host a web-based Jabber client such as JWChat[17] (there is a tutorial to install JWChat[18] with instructions for `ejabberd`).

`inet6` Set up the socket for IPv6.

`{ip, IPAddress}` This option specifies which network interface to listen for. For example `{ip, {192, 168, 1, 1}}`.

`{max_stanza_size, Size}` This option specifies an approximate maximum size in bytes of XML stanzas. Approximate, because it is calculated with the precision of one block of readed data. For example `{max_stanza_size, 65536}`. The default value is `infinity`.

`{shaper, <access rule>}` This option defines a shaper for the port (see section 3.6). The default value is `none`.

[14] http://ejabberd.jabber.ru/tutorials-transports
[15] http://www.jabber.org/jeps/jep-0114.html
[16] http://www.jabber.org/jeps/jep-0025.html
[17] http://jwchat.sourceforge.net/
[18] http://ejabberd.jabber.ru/jwchat

`ssl` This option specifies that traffic on the port will be encrypted using SSL. You should also set the `certfile` option. It is recommended to use the `tls` option instead.

`starttls` This option specifies that STARTTLS encryption is available on connections to the port. You should also set the `certfile` option.

`starttls_required` This option specifies that STARTTLS encryption is required on connections to the port. No unencrypted connections will be allowed. You should also set the `certfile` option.

`tls` This option specifies that traffic on the port will be encrypted using SSL immediately after connecting. You should also set the `certfile` option.

`zlib` This option specifies that Zlib stream compression (as defined in JEP-0138[19]) is available on connections to the port. Client connections cannot use stream compression and stream encryption simultaneously. Hence, if you specify both `tls` (or `ssl`) and `zlib`, the latter option will not affect connections (there will be no stream compression).

`web_admin` This option enables the web interface for `ejabberd` administration which is available at `http://server:port/admin/`. Login and password are the username and password of one of the registered users who are granted access by the 'configure' access rule.

In addition, the following options are available for s2s connections:

`{s2s_use_starttls, true|false}` This option defines whether to use STARTTLS for s2s connections.

`{s2s_certfile, Path}` Path to a file containing a SSL certificate.

`{domain_certfile, Domain, Path}` Path to the file containing the SSL certificate for the specified domain.

For instance, the following configuration defines that:

- c2s connections are listened for on port 5222 and 5223 (SSL) and denied for the user called '`bad`'.
- s2s connections are listened for on port 5269 with STARTTLS for secured traffic enabled.
- Port 5280 is serving the web interface and the HTTP Polling service. Note that it is also possible to serve them on different ports. The second example in section 7.1 shows how exactly this can be done.
- All users except for the administrators have a traffic of limit 1,000 Bytes/second
- The AIM transport[20] `aim.example.org` is connected to port 5233 with password '`aimsecret`'.
- The ICQ transport JIT (`icq.example.org` and `sms.example.org`) is connected to port 5234 with password '`jitsecret`'.

[19] http://www.jabber.org/jeps/jep-0138.html
[20] http://ejabberd.jabber.ru/pyaimt

- The MSN transport[21] `msn.example.org` is connected to port 5235 with password '`msnsecret`'.
- The Yahoo! transport[22] `yahoo.example.org` is connected to port 5236 with password '`yahoosecret`'.
- The Gadu-Gadu transport[23] `gg.example.org` is connected to port 5237 with password '`ggsecret`'.
- The Jabber Mail Component[24] `jmc.example.org` is connected to port 5238 with password '`jmcsecret`'.

```
{acl, blocked, {user, "bad"}}.
{access, c2s, [{deny, blocked},
               {allow, all}]}.
{shaper, normal, {maxrate, 1000}}.
{access, c2s_shaper, [{none, admin},
                      {normal, all}]}.
{listen,
 [{5222, ejabberd_c2s,     [{access, c2s}, {shaper, c2s_shaper}]},
  {5223, ejabberd_c2s,     [{access, c2s},
                            ssl, {certfile, "/path/to/ssl.pem"}]},
  {5269, ejabberd_s2s_in,  []},
  {5280, ejabberd_http,    [http_poll, web_admin]},
  {5233, ejabberd_service, [{host, "aim.example.org",
                             [{password, "aimsecret"}]}]},
  {5234, ejabberd_service, [{hosts, ["icq.example.org", "sms.example.org"],
                             [{password, "jitsecret"}]}]},
  {5235, ejabberd_service, [{host, "msn.example.org",
                             [{password, "msnsecret"}]}]},
  {5236, ejabberd_service, [{host, "yahoo.example.org",
                             [{password, "yahoosecret"}]}]},
  {5237, ejabberd_service, [{host, "gg.example.org",
                             [{password, "ggsecret"}]}]},
  {5238, ejabberd_service, [{host, "jmc.example.org",
                             [{password, "jmcsecret"}]}]}
 ]
}.
{s2s_use_starttls, true}.
{s2s_certfile, "/path/to/ssl.pem"}.
```

Note, that for jabberd 1.4- or WPJabber-based services you have to make the transports log and do XDB by themselves:

```
<!--
   You have to add elogger and rlogger entries here when using ejabberd.
```

[21] http://ejabberd.jabber.ru/pymsnt
[22] http://ejabberd.jabber.ru/yahoo-transport-2
[23] http://ejabberd.jabber.ru/jabber-gg-transport
[24] http://ejabberd.jabber.ru/jmc

```
   In this case the transport will do the logging.
-->

<log id='logger'>
  <host/>
  <logtype/>
  <format>%d: [%t] (%h): %s</format>
  <file>/var/log/jabber/service.log</file>
</log>

<!--
   Some Jabber server implementations do not provide
   XDB services (for example, jabberd2 and ejabberd).
   xdb_file.so is loaded in to handle all XDB requests.
-->

<xdb id="xdb">
  <host/>
  <load>
    <!-- this is a lib of wpjabber or jabberd -->
    <xdb_file>/usr/lib/jabber/xdb_file.so</xdb_file>
    </load>
  <xdb_file xmlns="jabber:config:xdb_file">
    <spool><jabberd:cmdline flag='s'>/var/spool/jabber</jabberd:cmdline></spool>
  </xdb_file>
</xdb>
```

3.4 Authentication

The option `auth_method` defines the authentication method that is used for user authentication:

```
{auth_method, [<method>]}.
```

The following authentication methods are supported by `ejabberd`:

- internal (default) — See section 3.4.1.
- external — There are some example authentication scripts[25].
- ldap — See section 4.5.
- odbc — See section 4.1, 4.3, 4.2 and 4.4.
- anonymous — See section 3.4.2.

[25] http://ejabberd.jabber.ru/extauth

3.4.1 Internal

`ejabberd` uses its internal Mnesia database as the default authentication method.

- `auth_method`: The value `internal` will enable the internal authentication method.

Examples:

- To use internal authentication on `example.org` and LDAP authentication on `example.net`:

```
{host_config, "example.org", [{auth_method, [internal]}]}.
{host_config, "example.net", [{auth_method, [ldap]}]}.
```

- To use internal authentication on all virtual hosts:

```
{auth_method, internal}.
```

3.4.2 SASL Anonymous and Anonymous Login

The anonymous authentication method can be configured with the following options. Remember that you can use the `host_config` option to set virtual host specific options (see section 3.2). Note that there also is a detailed tutorial regarding SASL Anonymous and anonymous login configuration[26].

- `auth_method`: The value `anonymous` will enable the anonymous authentication method.
- `allow_multiple_connections`: This value for this option can be either `true` or `false` and is only used when the anonymous mode is enabled. Setting it to `true` means that the same username can be taken multiple times in anonymous login mode if different resource are used to connect. This option is only useful in very special occasions. The default value is `false`.
- `anonymous_protocol`: This option can take three values: `sasl_anon`, `login_anon` or `both`. `sasl_anon` means that the SASL Anonymous method will be used. `login_anon` means that the anonymous login method will be used. `both` means that SASL Anonymous and login anonymous are both enabled.

Those options are defined for each virtual host with the `host_config` parameter (see section 3.2).

Examples:

- To enable anonymous login on all virtual hosts:

```
{auth_method, [anonymous]}.
{anonymous_protocol, login_anon}.
```

[26] http://support.process-one.net/doc/display/MESSENGER/Anonymous+users+support

- Similar as previous example, but limited to `public.example.org`:

```
{host_config, "public.example.org", [{auth_method, [anonymous]},
                                     {anonymous_protocol, login_anon}]}.
```

- To enable anonymous login and internal authentication on a virtual host:

```
{host_config, "public.example.org", [{auth_method, [anonymous,internal]},
                                     {anonymous_protocol, login_anon}]}.
```

- To enable SASL Anonymous on a virtual host:

```
{host_config, "public.example.org", [{auth_method, [anonymous]},
                                     {anonymous_protocol, sasl_anon}]}.
```

- To enable SASL Anonymous and anonymous login on a virtual host:

```
{host_config, "public.example.org", [{auth_method, [anonymous]},
                                     {anonymous_protocol, both}]}.
```

- To enable SASL Anonymous, anonymous login, and internal authentication on a virtual host:

```
{host_config, "public.example.org", [{auth_method, [anonymous,internal]},
                                     {anonymous_protocol, both}]}.
```

3.5 Access Rules

Access control in `ejabberd` is performed via Access Control Lists (ACLs). The declarations of ACLs in the configuration file have the following syntax:

```
{acl, <aclname>, {<acltype>, ...}}.
```

`<acltype>` can be one of the following:

`all` Matches all JIDs. Example:

```
{acl, all, all}.
```

`{user, <username>}` Matches the user with the name `<username>` at the first virtual host. Example:

```
{acl, admin, {user, "yozhik"}}.
```

`{user, <username>, <server>}` Matches the user with the JID `<username>@<server>` and any resource. Example:

```
{acl, admin, {user, "yozhik", "example.org"}}.
```

`{server, <server>}` Matches any JID from server `<server>`. Example:

```
{acl, exampleorg, {server, "example.org"}}.
```

`{user_regexp, <regexp>}` Matches any local user with a name that matches `<regexp>` at the first virtual host. Example:

```
{acl, tests, {user, "^test[0-9]*$"}}.
```

`{user_regexp, <regexp>, <server>}` Matches any user with a name that matches `<regexp>` at server `<server>`. Example:

```
{acl, tests, {user, "^test", "example.org"}}.
```

`{server_regexp, <regexp>}` Matches any JID from the server that matches `<regexp>`. Example:

```
{acl, icq, {server, "^icq\\."}}.
```

`{node_regexp, <user_regexp>, <server_regexp>}` Matches any user with a name that matches `<user_regexp>` at any server that matches `<server_regexp>`. Example:

```
{acl, yohzik, {node_regexp, "^yohzik$", "^example.(com|org)$"}}.
```

`{user_glob, <glob>}`

`{user_glob, <glob>, <server>}`

`{server_glob, <glob>}`

`{node_glob, <user_glob>, <server_glob>}` This is the same as above. However, it uses shell glob patterns instead of regexp. These patterns can have the following special characters:

- `*` matches any string including the null string.
- `?` matches any single character.
- `[...]` matches any of the enclosed characters. Character ranges are specified by a pair of characters separated by a '`-`'. If the first character after '`[`' is a '`!`', any character not enclosed is matched.

The following ACLs are pre-defined:

`all` Matches any JID.

`none` Matches no JID.

An entry allowing or denying access to different services looks similar to this:

```
{access, <accessname>, [{allow, <aclname>},
                        {deny, <aclname>},
                        ...
                       ]}.
```

When a JID is checked to have access to `<accessname>`, the server sequentially checks if that JID mathes any of the ACLs that are named in the second elements of the tuples in the list. If it matches, the first element of the first matched tuple is returned, otherwise the value '`deny`' is returned.

Example:

```
{access, configure, [{allow, admin}]}.
{access, something, [{deny, badmans},
                     {allow, all}]}.
```

The following access rules are pre-defined:

`all` Always returns the value '`allow`'.

`none` Always returns the value '`deny`'.

3.6 Shapers

Shapers enable you to limit connection traffic. The syntax of shapers is like this:

```
{shaper, <shapername>, <kind>}.
```

Currently only one kind of shaper called `maxrate` is available. It has the following syntax:

```
{maxrate, <rate>}
```

where `<rate>` stands for the maximum allowed incomig rate in bytes per second.

Examples:

- To define a shaper named '`normal`' with traffic speed limited to 1,000 bytes/second:

  ```
  {shaper, normal, {maxrate, 1000}}.
  ```

- To define a shaper named '`fast`' with traffic speed limited to 50,000 bytes/second:

  ```
  {shaper, fast, {maxrate, 50000}}.
  ```

3.7 Limiting Opened Sessions

This option specifies the maximum number of sessions (authenticated connections) per user. If a user tries to open more sessions by using different resources, the first opened session will be disconnected. The error `session replaced` will be sent to the disconnected session. The value for this option can be either a number, or `infinity`. The default value is `10`.

Examples:

- To limit the number of sessions per user to 10 on all virtual hosts:

```
{max_user_sessions, 10}.
```

- This option can be defined per virtual host (see section 3.2). In next example the number of sessions per user on the first host is six, while there is no limit on the second host:

```
{host_config, "example.net", [{max_user_sessions, 6}]}.
{host_config, "example.com", [{max_user_sessions, infinity}]}.
```

3.8 Default Language

The option `language` defines the default language of server strings that can be seen by Jabber clients. If a Jabber client do not support `xml:lang`, the specified language is used. The default value is `en`. In order to take effect there must be a translation file `<language>.msg` in `ejabberd`'s `msgs` directory.

Examples:

- To set Russian as default language:

```
{language, "ru"}.
```

- To set Spanish as default language:

```
{language, "es"}.
```

4 Database Configuration

`ejabberd` uses its internal Mnesia database by default. However, it is possible to use a relational database or an LDAP server to store persistant, long-living data. `ejabberd` is very flexible: you can configure different authentication methods for different virtual hosts, you can configure different authentication mechanisms for the same virtual host (fallback), you can set different storage systems for modules, and so forth.

The following databases are supported by `ejabberd`:

- Microsoft SQL Server[27]
- Mnesia[28]
- MySQL[29]
- Any ODBC compatible database[30]
- PostgreSQL[31]

The following LDAP servers are tested with `ejabberd`:

- Active Directory[32] (see section 4.5.3)
- OpenLDAP[33]
- Normally any LDAP compatible server should work; inform us about your success with a not-listed server so that we can list it here.

4.1 MySQL

Although this section will describe `ejabberd`'s configuration when you want to use the native MySQL driver, it does not describe MySQL's installation and database creation. Check the MySQL documentation and the tutorial Using ejabberd with MySQL native driver[34] for information regarding these topics. Note that the tutorial contains information about `ejabberd`'s configuration which is duplicate to this section.

Moreover, the file mysql.sql in the directory src/odbc might be interesting for you. This file contains the ejabberd schema for MySQL. At the end of the file you can find information to update your database schema.

4.1.1 Driver Compilation

You can skip this step if you installed `ejabberd` using a binary installer or if the binary packages of `ejabberd` you are using include support for MySQL.

1. First, install the Erlang MySQL library[35]. Make sure the compiled files are in your Erlang path; you can put them for example in the same directory as your ejabberd .beam files.
2. Then, configure and install `ejabberd` with ODBC support enabled (this is also needed for native MySQL support!). This can be done, by using next commands:

```
./configure --enable-odbc && make install
```

[27] http://www.microsoft.com/sql/
[28] http://www.erlang.org/doc/doc-5.5.1/lib/mnesia-4.3.2/doc/
[29] http://mysql.com/
[30] http://en.wikipedia.org/wiki/Open_Database_Connectivity
[31] http://www.postgresql.org/
[32] http://www.microsoft.com/activedirectory/
[33] http://www.openldap.org/
[34] http://support.process-one.net/doc/display/MESSENGER/Using+ejabberd+with+MySQL+native+driver
[35] http://support.process-one.net/doc/display/CONTRIBS/Yxa

4.1.2 Authentication

The option value name may be misleading, as the `auth_method` name is used for access to a relational database through ODBC, as well as through the native MySQL interface. Anyway, the first configuration step is to define the odbc `auth_method`. For example:

```
{host_config, "public.example.org", [{auth_method, [odbc]}]}.
```

The actual database access is defined in the option `odbc_server`. Its value is used to define if we want to use ODBC, or one of the two native interface available, PostgreSQL or MySQL.

To use the native MySQL interface, you can pass a tuple of the following form as parameter:

```
{mysql, "Server", "Database", "Username", "Password"}
```

`mysql` is a keyword that should be kept as is. For example:

```
{odbc_server, {mysql, "localhost", "test", "root", "password"}}.
```

4.1.3 Storage

MySQL also can be used to store information into from several `ejabberd` modules. See section 5.1 to see which modules have a version with the '_odbc'. This suffix indicates that the module can be used with relational databases like MySQL. To enable storage to your database, just make sure that your database is running well (see previous sections), and replace the suffix-less or ldap module variant with the odbc module variant. Keep in mind that you cannot have several variants of the same module loaded!

4.2 Microsoft SQL Server

Although this section will describe `ejabberd`'s configuration when you want to use Microsoft SQL Server, it does not describe Microsoft SQL Server's installation and database creation. Check the MySQL documentation and the tutorial Using ejabberd with MySQL native driver[36] for information regarding these topics. Note that the tutorial contains information about `ejabberd`'s configuration which is duplicate to this section.

Moreover, the file mssql.sql in the directory src/odbc might be interesting for you. This file contains the ejabberd schema for Microsoft SQL Server. At the end of the file you can find information to update your database schema.

[36] http://support.process-one.net/doc/display/MESSENGER/Using+ejabberd+with+MySQL+native+driver

4.2.1 Driver Compilation

You can skip this step if you installed `ejabberd` using a binary installer or if the binary packages of `ejabberd` you are using include support for ODBC.

If you want to use Microsoft SQL Server with ODBC, you need to configure, compile and install `ejabberd` with support for ODBC and Microsoft SQL Server enabled. This can be done, by using next commands:

```
./configure --enable-odbc --enable-mssql && make install
```

4.2.2 Authentication

The configuration of Microsoft SQL Server is the same as the configuration of ODBC compatible serers (see section 4.4.2).

4.2.3 Storage

Microsoft SQL Server also can be used to store information into from several `ejabberd` modules. See section 5.1 to see which modules have a version with the '_odbc'. This suffix indicates that the module can be used with relational databases like Microsoft SQL Server. To enable storage to your database, just make sure that your database is running well (see previous sections), and replace the suffix-less or ldap module variant with the odbc module variant. Keep in mind that you cannot have several variants of the same module loaded!

4.3 PostgreSQL

Although this section will describe `ejabberd`'s configuration when you want to use the native PostgreSQL driver, it does not describe PostgreSQL's installation and database creation. Check the PostgreSQL documentation and the tutorial Using ejabberd with MySQL native driver[37] for information regarding these topics. Note that the tutorial contains information about `ejabberd`'s configuration which is duplicate to this section.

Also the file pg.sql in the directory src/odbc might be interesting for you. This file contains the ejabberd schema for PostgreSQL. At the end of the file you can find information to update your database schema.

4.3.1 Driver Compilation

You can skip this step if you installed `ejabberd` using a binary installer or if the binary packages of `ejabberd` you are using include support for PostgreSQL.

[37] http://support.process-one.net/doc/display/MESSENGER/Using+ejabberd+with+MySQL+native+driver

1. First, install the Erlang PgSQL library from Jungerl[38]. Make sure the compiled files are in your Erlang path; you can put them for example in the same directory as your ejabberd .beam files.
2. Then, configure, compile and install `ejabberd` with ODBC support enabled (this is also needed for native PostgreSQL support!). This can be done, by using next commands:

   ```
   ./configure --enable-odbc && make install
   ```

4.3.2 Authentication

The option value name may be misleading, as the `auth_method` name is used for access to a relational database through ODBC, as well as through the native PostgreSQL interface. Anyway, the first configuration step is to define the odbc `auth_method`. For example:

```
{host_config, "public.example.org", [{auth_method, [odbc]}]}.
```

The actual database access is defined in the option `odbc_server`. Its value is used to define if we want to use ODBC, or one of the two native interface available, PostgreSQL or MySQL.

To use the native PostgreSQL interface, you can pass a tuple of the following form as parameter:

```
{pgsql, "Server", "Database", "Username", "Password"}
```

`pgsql` is a keyword that should be kept as is. For example:

```
{odbc_server, {pgsql, "localhost", "database", "ejabberd", "password"}}.
```

4.3.3 Storage

PostgreSQL also can be used to store information into from several `ejabberd` modules. See section 5.1 to see which modules have a version with the '_odbc'. This suffix indicates that the module can be used with relational databases like PostgreSQL. To enable storage to your database, just make sure that your database is running well (see previous sections), and replace the suffix-less or ldap module variant with the odbc module variant. Keep in mind that you cannot have several variants of the same module loaded!

4.4 ODBC Compatible

Although this section will describe `ejabberd`'s configuration when you want to use the ODBC driver, it does not describe the installation and database creation of your database. Check the documentation of your database. The tutorial Using ejabberd with MySQL native driver[39] also can help you. Note that the tutorial contains information about `ejabberd`'s configuration which is duplicate to this section.

[38] http://jungerl.sourceforge.net/
[39] http://support.process-one.net/doc/display/MESSENGER/Using+ejabberd+with+MySQL+native+driver

4.4.1 Compilation

You can skip this step if you installed `ejabberd` using a binary installer or if the binary packages of `ejabberd` you are using include support for ODBC.

1. First, install the Erlang MySQL library[40]. Make sure the compiled files are in your Erlang path; you can put them for example in the same directory as your ejabberd .beam files.
2. Then, configure, compile and install `ejabberd` with ODBC support enabled. This can be done, by using next commands:

   ```
   ./configure --enable-odbc && make install
   ```

4.4.2 Authentication

The first configuration step is to define the odbc `auth_method`. For example:

```
{host_config, "public.example.org", [{auth_method, [odbc]}]}.
```

The actual database access is defined in the option `odbc_server`. Its value is used to defined if we want to use ODBC, or one of the two native interface available, PostgreSQL or MySQL.

To use a relational database through ODBC, you can pass the ODBC connection string as `odbc_server` parameter. For example:

```
{odbc_server, "DSN=database;UID=ejabberd;PWD=password"}.
```

4.4.3 Storage

An ODBC compatible database also can be used to store information into from several `ejabberd` modules. See section 5.1 to see which modules have a version with the '_odbc'. This suffix indicates that the module can be used with ODBC compatible relational databases. To enable storage to your database, just make sure that your database is running well (see previous sections), and replace the suffix-less or ldap module variant with the odbc module variant. Keep in mind that you cannot have several variants of the same module loaded!

4.5 LDAP

`ejabberd` has built-in LDAP support. You can authenticate users against LDAP server and use LDAP directory as vCard storage. Shared rosters are not supported yet.

[40] http://support.process-one.net/doc/display/CONTRIBS/Yxa

4.5.1 Connection

Parameters:

`ldap_server` IP address or dns name of your LDAP server. This option is required.

`ldap_port` Port to connect to your LDAP server. The default value is 389.

`ldap_rootdn` Bind DN. The default value is `""` which means 'anonymous connection'.

`ldap_password` Bind password. The default value is `""`.

Example:

```
{auth_method, ldap}.
{ldap_servers, ["ldap.example.org"]}.
{ldap_port, 389}.
{ldap_rootdn, "cn=Manager,dc=domain,dc=org"}.
{ldap_password, "secret"}.
```

Note that current LDAP implementation does not support SSL secured communication and SASL authentication.

4.5.2 Authentication

You can authenticate users against an LDAP directory. Available options are:

`ldap_base` LDAP base directory which stores users accounts. This option is required.

`ldap_uidattr` LDAP attribute which holds the user's part of a JID. The default value is `"uid"`.

`ldap_uidattr_format` Format of the `ldap_uidattr` variable. The format *must* contain one and only one pattern variable `"%u"` which will be replaced by the user's part of a JID. For example, `"%u@example.org"`. The default value is `"%u"`.

`ldap_filter` RFC 2254[41] LDAP filter. The default is `none`. Example: `"(&(objectClass=shadowAccount)(memberOf=Users))"`. Please, do not forget to close brackets and do not use superfluous whitespaces. Also you *must not* use `ldap_uidattr` attribute in filter because this attribute will be substituted in LDAP filter automatically.

[41] http://www.faqs.org/rfcs/rfc2254.html

4.5.3 Examples

Common example Let's say `ldap.example.org` is the name of our LDAP server. We have users with their passwords in `"ou=Users,dc=example,dc=org"` directory. Also we have addressbook, which contains users emails and their additional infos in `"ou=AddressBook,dc=example,dc=org"` directory. Corresponding authentication section should looks like this:

```
%% authentication method
{auth_method, ldap}.
%% DNS name of our LDAP server
{ldap_servers, ["ldap.example.org"]}.
%% Bind to LDAP server as "cn=Manager,dc=example,dc=org" with password "secret"
{ldap_rootdn, "cn=Manager,dc=example,dc=org"}.
{ldap_password, "secret"}.
%% define the user's base
{ldap_base, "ou=Users,dc=example,dc=org"}.
%% We want to authorize users from 'shadowAccount' object class only
{ldap_filter, "(objectClass=shadowAccount)"}.
```

Now we want to use users LDAP-info as their vCards. We have four attributes defined in our LDAP schema: `"mail"` — email address, `"givenName"` — first name, `"sn"` — second name, `"birthDay"` — birthday. Also we want users to search each other. Let's see how we can set it up:

```
{modules,
  ...
  {mod_vcard_ldap,
   [
    %% We use the same server and port, but want to bind anonymously because
    %% our LDAP server accepts anonymous requests to
    %% "ou=AddressBook,dc=example,dc=org" subtree.
    {ldap_rootdn, ""},
    {ldap_password, ""},
    %% define the addressbook's base
    {ldap_base, "ou=AddressBook,dc=example,dc=org"},
    %% user's part of JID is located in the "mail" attribute
    {ldap_uidattr, "mail"},
    %% common format for our emails
    {ldap_uidattr_format, "%u@mail.example.org"},
    %% We have to define empty filter here, because entries in addressbook does not
    %% belong to shadowAccount object class
    {ldap_filter, ""},
    %% Now we want to define vCard pattern
    {ldap_vcard_map,
     [{"NICKNAME", "%u", []}, % just use user's part of JID as his nickname
      {"GIVEN", "%s", ["givenName"]},
      {"FAMILY", "%s", ["sn"]},
```

```
      {"FN", "%s, %s", ["sn", "givenName"]}, % example: "Smith, John"
      {"EMAIL", "%s", ["mail"]},
      {"BDAY", "%s", ["birthDay"]}]},
    %% Search form
    {ldap_search_fields,
     [{"User", "%u"},
      {"Name", "givenName"},
      {"Family Name", "sn"},
      {"Email", "mail"},
      {"Birthday", "birthDay"}]},
    %% vCard fields to be reported
    %% Note that JID is always returned with search results
    {ldap_search_reported,
     [{"Full Name", "FN"},
      {"Nickname", "NICKNAME"},
      {"Birthday", "BDAY"}]}
  ]}
  ...
}.
```

Note that `mod_vcard_ldap` module checks for the existence of the user before searching in his information in LDAP.

Active Directory Active Directory is just an LDAP-server with predefined attributes. A sample configuration is showed below:

```
{auth_method, ldap}.
{ldap_servers, ["office.org"]}.    % List of LDAP servers
{ldap_base, "DC=office,DC=org"}. % Search base of LDAP directory
{ldap_rootdn, "CN=Administrator,CN=Users,DC=office,DC=org"}. % LDAP manager
{ldap_password, "*******"}. % Password to LDAP manager
{ldap_uidattr, "sAMAccountName"}.
{ldap_filter, "(memberOf=*)"}.

{mod_vcard_ldap,
 [{ldap_vcard_map,
   [{"NICKNAME", "%u", []},
    {"GIVEN", "%s", ["givenName"]},
    {"MIDDLE", "%s", ["initials"]},
    {"FAMILY", "%s", ["sn"]},
    {"FN", "%s", ["displayName"]},
    {"EMAIL", "%s", ["mail"]},
    {"ORGNAME", "%s", ["company"]},
    {"ORGUNIT", "%s", ["department"]},
    {"CTRY", "%s", ["c"]},
    {"LOCALITY", "%s", ["l"]},
    {"STREET", "%s", ["streetAddress"]},
```

```
    {"REGION", "%s", ["st"]},
    {"PCODE", "%s", ["postalCode"]},
    {"TITLE", "%s", ["title"]},
    {"URL", "%s", ["wWWHomePage"]},
    {"DESC", "%s", ["description"]},
    {"TEL", "%s", ["telephoneNumber"]}]},
  {ldap_search_fields,
   [{"User", "%u"},
    {"Name", "givenName"},
    {"Family Name", "sn"},
    {"Email", "mail"},
    {"Company", "company"},
    {"Department", "department"},
    {"Role", "title"},
    {"Description", "description"},
    {"Phone", "telephoneNumber"}]},
  {ldap_search_reported,
   [{"Full Name", "FN"},
    {"Nickname", "NICKNAME"},
    {"Email", "EMAIL"}]}
 ]
}.
```

5 Modules Configuration

The option `modules` defines the list of modules that will be loaded after `ejabberd`'s startup. Each entry in the list is a tuple in which the first element is the name of a module and the second is a list of options for that module.

Examples:

- In this example only the module `mod_echo` is loaded and no module options are specified between the square brackets:

```
{modules,
 [{mod_echo,      []}
 ]}.
```

- In the second example the modules `mod_echo`, `mod_time`, and `mod_version` are loaded without options. Remark that, besides the last entry, all entries end with a comma:

```
{modules,
 [{mod_echo,      []},
  {mod_time,      []},
  {mod_version,   []}
 ]}.
```

5.1 Overview

The following table lists all modules available in the official `ejabberd` distribution. You can find more contributed modules[42] on the `ejabberd` website. Please remember that these contributions might not work or that they can contain severe bugs and security leaks. Therefore, use them at your own risk!

You can see which database backend each module needs by looking at the suffix:

- '_ldap', this means that the module needs an LDAP server as backend.
- '_odbc', this means that the module needs a supported database (see 4) as backend.
- No suffix, this means that the modules uses Erlang's built-in database Mnesia as backend.

If you want to It is possible to use a relational database to store pieces of information. You can do this by changing the module name to a name with an `_odbc` suffix in `ejabberd` config file. You can use a relational database for the following data:

- Last connection date and time: Use `mod_last_odbc` instead of `mod_last`.
- Offline messages: Use `mod_offline_odbc` instead of `mod_offline`.
- Rosters: Use `mod_roster_odbc` instead of `mod_roster`.
- Users' VCARD: Use `mod_vcard_odbc` instead of `mod_vcard`.

[42] http://ejabberd.jabber.ru/contributions

Module	Feature	Dependencies	Needed for XMPP?
mod_adhoc	Ad-Hoc Commands (JEP-0050[43])		No
mod_announce	Manage announcements	mod_adhoc	No
mod_configure	Support for online configuration of ejabberd	mod_adhoc	No
mod_disco	Service Discovery (JEP-0030[44])		No
mod_echo	Echoes Jabber packets		No
mod_irc	IRC transport		No
mod_last	Last Activity (JEP-0012[45])		No
mod_last_odbc	Last Activity (JEP-0012[46])	supported database (*)	No
mod_muc	Multi-User Chat (JEP-0045[47])		No
mod_muc_log	Multi-User Chat room logging	mod_muc	No
mod_offline	Offline message storage		No
mod_offline_odbc	Offline message storage	supported database (*)	No
mod_privacy	Blocking Communication		Yes
mod_private	Private XML Storage (JEP-0049[48])		No
mod_pubsub	Publish-Subscribe (JEP-0060[49])		No
mod_register	In-Band Registration (JEP-0077[50])		No
mod_roster	Roster management		Yes (**)
mod_roster_odbc	Roster management	supported database (*)	Yes (**)
mod_service_log	Copy user messages to logger service		No
mod_shared_roster	Shared roster management	mod_roster or mod_roster_odbc	No
mod_stats	Statistics Gathering (JEP-0039[51])		No
mod_time	Entity Time (JEP-0090[52])		No
mod_vcard	vcard-temp (JEP-0054[53])		No
mod_vcard_ldap	vcard-temp (JEP-0054[54])	LDAP server	No
mod_vcard_odbc	vcard-temp (JEP-0054[55])	supported database (*)	No
mod_version	Software Version (JEP-0092[56])		No

- (*) For a list of supported databases, see section 4.
- (**) This module or a similar one with another database backend is needed for XMPP compliancy.

5.2 Common Options

The following options are used by many modules. Therefore, they are described in this separate section.

5.2.1 iqdisc

Many modules define handlers for processing IQ queries of different namespaces to this server or to a user (e. g. to example.org or to user@example.org). This option defines processing discipline for these queries. Possible values are:

`no_queue` All queries of a namespace with this processing discipline are processed immediately. This also means that no other packets can be processed until this one has been completely processed. Hence this discipline is not recommended if the processing of a query can take a relatively long time.

`one_queue` In this case a separate queue is created for the processing of IQ queries of a namespace with this discipline. In addition, the processing of this queue is done in parallel with that of other packets. This discipline is most recommended.

`parallel` For every packet with this discipline a separate Erlang process is spawned. Consequently, all these packets are processed in parallel. Although spawning of Erlang process has a relatively low cost, this can break the server's normal work, because the Erlang emulator has a limit on the number of processes (32000 by default).

Example:

```
{modules,
 [
  ...
  {mod_time, [{iqdisc, no_queue}]},
  ...
 ]}.
```

5.2.2 `hosts`

A module acting as a service can have one or more hostnames. These hostnames can be defined with the `hosts` option.

Examples:

- Serving the echo module on one domain:
  - ```
    {modules,
     [
      ...
      {mod_echo, [{hosts, ["echo.example.org"]}]},
      ...
     ]}.
    ```
  - Backwards compatibility with older `ejabberd` versions can be retained with:

    ```
    {modules,
     [
      ...
      {mod_echo, [{host, "echo.example.org"}]},
      ...
     ]}.
    ```
- Serving the echo module on two domains:
    ```

```
{modules,
 [
  ...
  {mod_echo, [{hosts, ["echo.example.net", "echo.example.com"]}]},
  ...
 ]}.
```

5.3 mod_announce

This module enables configured users to broadcast announcements and to set the message of the day (MOTD). Configured users can do these actions with their Jabber client by sending messages to specific JIDs. These JIDs are listed in next paragraph. The first JID in each entry will apply only to the virtual host `example.org`, while the JID between brackets will apply to all virtual hosts:

`example.org/announce/all (example.org/announce/all-hosts/all)` The message is sent to all registered users. If the user is online and connected to several resources, only the resource with the highest priority will receive the message. If the registered user is not connected, the message will be stored offline in assumption that offline storage (see section 5.10) is enabled.

`example.org/announce/online (example.org/announce/all-hosts/online)` The message is sent to all connected users. If the user is online and connected to several resources, all resources will receive the message.

`example.org/announce/motd (example.org/announce/all-hosts/motd)` The message is set as the message of the day (MOTD) and is sent to users when they login. In addition the message is sent to all connected users (similar to `announce/online`).

`example.org/announce/motd/update (example.org/announce/all-hosts/motd/update)` The message is set as message of the day (MOTD) and is sent to users when they login. The message is *not sent* to any currently connected user.

`example.org/announce/motd/delete (example.org/announce/all-hosts/motd/delete)` Any message sent to this JID removes the existing message of the day (MOTD).

Options:

`access` This option specifies who is allowed to send announcements and to set the message of the day (by default, nobody is able to send such messages).

Examples:

- Only administrators can send announcements:

```
{access, announce, [{allow, admins}]}.

{modules,
 [
  ...
  {mod_announce, [{access, announce}]},
  ...
 ]}.
```

- Administrators as well as the direction can send announcements:

```
{acl, direction, {user, "big_boss", "example.org"}}.
{acl, direction, {user, "assistant", "example.org"}}.
{acl, admins, {user, "admin", "example.org"}}.
...
{access, announce, [{allow, admins},
                    {allow, direction}]}.
...
{modules,
 [
  ...
  {mod_announce, [{access, announce}]},
  ...
 ]}.
```

5.4 mod_disco

This module adds support for Service Discovery (JEP-0030[57]). With this module enabled, services on your server can be discovered by Jabber clients. Note that `ejabberd` has no modules with support for the superseded Jabber Browsing (JEP-0011[58]) and Agent Information (JEP-0094[59]). Accordingly, Jabber clients need to have support for the newer Service Discovery protocol if you want them be able to discover the services you offer.

Options:

`iqdisc` This specifies the processing discipline for Service Discovery (`http://jabber.org/protocol/disco#items` and `http://jabber.org/protocol/disco#info`) IQ queries (see section 5.2.1).

`extra_domains` With this option, extra domains can be added to the Service Discovery item list.

Examples:

- To serve a link to the Jabber User Directory on `jabber.org`:

[57] http://www.jabber.org/jeps/jep-0030.html
[58] http://www.jabber.org/jeps/jep-0011.html
[59] http://www.jabber.org/jeps/jep-0094.html

```
{modules,
 [
  ...
  {mod_disco, [{extra_domains, ["users.jabber.org"]}]},
  ...
 ]}.
```

- To serve a link to the transports on another server:

```
{modules,
 [
  ...
  {mod_disco, [{extra_domains, ["icq.example.com",
                                "msn.example.com"]}]},
  ...
 ]}.
```

- To serve a link to a few friendly servers:

```
{modules,
 [
  ...
  {mod_disco, [{extra_domains, ["example.org",
                                "example.com"]}]},
  ...
 ]}.
```

5.5 mod_echo

This module simply echoes any Jabber packet back to the sender. This mirror can be of interest for `ejabberd` and Jabber client debugging.

Options:

`hosts` This option defines the hostnames of the service (see section 5.2.2). If neither `hosts` nor the old `host` is present, the prefix '`echo.`' is added to all `ejabberd` hostnames.

Examples:

- Mirror, mirror, on the wall, who is the most beautiful of them all?

```
{modules,
 [
  ...
  {mod_echo, [{hosts, ["mirror.example.org"]}]},
  ...
 ]}.
```

- If you still do not understand the inner workings of `mod_echo`, you can find a few more examples in section 5.2.2.

5.6 mod_irc

This module is an IRC transport that can be used to join channels on IRC servers.

End user information:

- A Jabber client with 'groupchat 1.0' support or Multi-User Chat support (JEP-0045[60]) is necessary to join IRC channels.
- An IRC channel can be joined in nearly the same way as joining a Jabber Multi-User Chat room. The difference is that the room name will be 'channel%`irc.example.org`' in case `irc.example.org` is the IRC server hosting 'channel'. And of course the host should point to the IRC transport instead of the Multi-User Chat service.
- You can register your nickame by sending 'IDENTIFY password' to `nickserver!irc.example.org@irc.jabberserver.org`.
- Entering your password is possible by sending 'LOGIN nick password' to `nickserver!irc.example.org@irc.jabberserver.org`.
- When using a popular Jabber server, it can occur that no connection can be achieved with some IRC servers because they limit the number of conections from one IP.

Options:

`hosts` This option defines the hostnames of the service (see section 5.2.2). If neither `hosts` nor the old `host` is present, the prefix '`irc.`' is added to all `ejabberd` hostnames.

`access` This option can be used to specify who may use the IRC transport (default value: `all`).

Examples:

- In the first example, the IRC transport is available on (all) your virtual host(s) with the prefix '`irc.`'. Furthermore, anyone is able to use the transport.

```
{modules,
 [
  ...
  {mod_irc, [{access, all}]},
  ...
 ]}.
```

- In next example the IRC transport is available on the two virtual hosts `example.net` and `example.com` with different prefixes on each host. Moreover, the transport is only accessible by paying customers registered on our domains and on other servers.

[60] http://www.jabber.org/jeps/jep-0045.html

```
{acl, paying_customers, {user, "customer1", "example.net"}}.
{acl, paying_customers, {user, "customer2", "example.com"}}.
{acl, paying_customers, {user, "customer3", "example.org"}}.
...
{access, paying_customers, [{allow, paying_customers},
                            {deny, all}]}.
...
{modules,
 [
  ...
  {mod_irc, [{access, paying_customers},
             {hosts, ["irc.example.net", "irc-transport.example.com"]}]},
  ...
 ]}.
```

5.7 mod_last

This module adds support for Last Activity (JEP-0012[61]). It can be used to discover when a disconnected user last accessed the server, to know when a connected user was last active on the server, or to query the uptime of the `ejabberd` server.

Options:

`iqdisc` This specifies the processing discipline for Last activity (`jabber:iq:last`) IQ queries (see section 5.2.1).

5.8 mod_muc

With this module enabled, your server will support Multi-User Chat (JEP-0045[62]). End users will be able to join text conferences. Notice that this module is not (yet) clusterable.

Some of the features of Multi-User Chat:

- Sending private messages to room participants.
- Inviting users.
- Setting a conference topic.
- Creating password protected rooms.
- Kicking and banning participants.

Options:

[61] http://www.jabber.org/jeps/jep-0012.html
[62] http://www.jabber.org/jeps/jep-0045.html

`hosts` This option defines the hostnames of the service (see section 5.2.2). If neither `hosts` nor the old `host` is present, the prefix '`conference.`' is added to all `ejabberd` hostnames.

`access` You can specify who is allowed to use the Multi-User Chat service (by default, everyone is allowed to use it).

`access_create` To configure who is allowed to create new rooms at the Multi-User Chat service, this option can be used (by default, everybody is allowed to create rooms).

`access_admin` This option specifies who is allowed to administrate the Multi-User Chat service (the default value is `none`, which means that only the room creator can administer his room). By sending a message to the service JID, administrators can send service messages that will be displayed in every active room.

`history_size` A small history of the current discussion is sent to users when they enter the room. With this option you can define the number of history messages to keep and send to users joining the room. The value is an integer. Setting the value to `0` disables the history feature and, as a result, nothing is kept in memory. The default value is `20`. This value is global and thus affects all rooms on the server.

Examples:

- In the first example everyone is allowed to use the Multi-User Chat service. Everyone will also be able to create new rooms but only the user `admin@example.org` is allowed to administrate any room. In this example he is also a global administrator. When `admin@example.org` sends a message such as 'Tomorrow, the Jabber server will be moved to new hardware. This will involve service breakdowns around 23:00 UMT. We apologise for this inconvenience.' to `conference.example.org`, it will be displayed in all active rooms. In this example the history feature is disabled.

```
{acl, admins, {user, "admin", "example.org"}}.
...
{access, muc_admins, [{allow, admins}]}.
...
{modules,
 [
  ...
  {mod_muc, [{access, all},
             {access_create, all},
             {access_admin, muc_admins},
             {history_size, 0}]},
  ...
 ]}.
```

- In the second example the Multi-User Chat service is only accessible by paying customers registered on our domains and on other servers. Of course the administrator is also allowed to access rooms. In addition, he is the only authority able to create and administer rooms. When `admin@example.org` sends a message such as 'Tomorrow, the Jabber server will be moved to new hardware. This will involve service breakdowns around 23:00 UMT. We apologise for this inconvenience.' to `conference.example.org`, it will be displayed in all

active rooms. No `history_size` option is used, this means that the feature is enabled and the default value of 20 history messages will be send to the users.

```
{acl, paying_customers, {user, "customer1", "example.net"}}.
{acl, paying_customers, {user, "customer2", "example.com"}}.
{acl, paying_customers, {user, "customer3", "example.org"}}.
{acl, admins, {user, "admin", "example.org"}}.
...
{access, muc_admins, [{allow, admins},
                      {deny, all}]}.
{access, muc_access, [{allow, paying_customers},
                      {allow, admins},
                      {deny, all}]}.
...
{modules,
 [
  ...
  {mod_muc, [{access, muc_access},
             {access_create, muc_admins},
             {access_admin, muc_admins}]},
  ...
 ]}.
```

5.9 mod_muc_log

This module enables optional logging of Multi-User Chat (MUC) conversations to HTML. Once you enable this module, users can join a chatroom using a MUC capable Jabber client, and if they have enough privileges, they can request the configuration form in which they can set the option to enable chatroom logging.

Features:

- Chatroom details are added on top of each page: room title, JID, author, subject and configuration.
- Room title and JID are links to join the chatroom (using XMPP URIs[63]).
- Subject and chatroom configuration changes are tracked and displayed.
- Joins, leaves, nick changes, kicks, bans and '/me' are tracked and displayed, including the reason if available.
- Generated HTML files are XHTML 1.0 Transitional and CSS compliant.
- Timestamps are self-referencing links.
- Links on top for quicker navigation: Previous day, Next day, Up.
- CSS is used for style definition, and a custom CSS file can be used.

[63] http://www.ietf.org/rfc/rfc4622.txt

- URLs on messages and subjects are converted to hyperlinks.
- Timezone used on timestamps is shown on the log files.
- A custom link can be added on top of each page.

Options:

`access_log` This option restricts which users are allowed to enable or disable chatroom logging. The default value is `muc_admin`. Note for this default setting you need to have an access rule for `muc_admin` in order to take effect.

`cssfile` With this option you can set whether the HTML files should have a custom CSS file or if they need to use the embedded CSS file. Allowed values are `false` and an URL to a CSS file. With the first value, HTML files will include the embedded CSS code. With the latter, you can specify the URL of the custom CSS file (for example: 'http://example.com/my.css'). The default value is `false`.

`dirtype` The type of the created directories can be specified with this option. Allowed values are `subdirs` and `plain`. With the first value, subdirectories are created for each year and month. With the latter, the names of the log files contain the full date, and there are no subdirectories. The default value is `subdirs`.

`outdir` This option sets the full path to the directory in which the HTML files should be stored. Make sure the `ejabberd` daemon user has write access on that directory. The default value is `"www/muc"`.

`timezone` The time zone for the logs is configurable with this option. Allowed values are `local` and `universal`. With the first value, the local time, as reported to Erlang by the operating system, will be used. With the latter, GMT/UTC time will be used. The default value is `local`.

`top_link` With this option you can customize the link on the top right corner of each log file. The syntax of this option is {`"URL", "Text"`}. The default value is {`"/", "Home"`}.

Examples:

- In the first example any chatroom owner can enable logging, and a custom CSS file will be used (http://example.com/my.css). Further, the names of the log files will contain the full date, and there will be no subdirectories. The log files will be stored in /var/www/muclogs, and the time zone will be GMT/UTC. Finally, the top link will be `<a href="http://www.jabber.ru">Jabber.ru</a>`.

```
{access, muc, [{allow, all}]}.
...
{modules,
 [
  ...
  {mod_muc_log, [
            {access_log, muc},
```

```
                {cssfile, "http://example.com/my.css"},
                {dirtype, plain},
                {outdir, "/var/www/muclogs"},
                {timezone, universal},
                {top_link, {"http://www.jabber.ru", "Jabber.ru"}}
  ]},
  ...
]}.
```

- In the second example only `admin1@example.org` and `admin2@example.net` can enable logging, and the embedded CSS file will be used. Further, the names of the log files will only contain the day (number), and there will be subdirectories for each year and month. The log files will be stored in /var/www/muclogs, and the local time will be used. Finally, the top link will be the default `<a href="/">Home</a>`.

```
{acl, admins, {user, "admin1", "example.org"}}.
{acl, admins, {user, "admin2", "example.net"}}.
...
{access, muc_log, [{allow, admins},
                   {deny, all}]}.
...
{modules,
 [
  ...
  {mod_muc_log, [
                {access_log, muc_log},
                {cssfile, false},
                {dirtype, subdirs},
                {outdir, "/var/www/muclogs"},
                {timezone, local}
  ]},
  ...
]}.
```

5.10 mod_offline

This module implements offline message storage. This means that all messages sent to an offline user will be stored on the server until that user comes online again. Thus it is very similar to how email works. Note that `ejabberdctl` has a command to delete expired messages (see section 7.2).

5.11 mod_privacy

This module implements Blocking Communication (also known as Privacy Rules) as defined in section 10 from XMPP IM. If end users have support for it in their Jabber client, they will be able to:

- Retrieving one's privacy lists.
- Adding, removing, and editing one's privacy lists.
- Setting, changing, or declining active lists.
- Setting, changing, or declining the default list (i.e., the list that is active by default).
- Allowing or blocking messages based on JID, group, or subscription type (or globally).
- Allowing or blocking inbound presence notifications based on JID, group, or subscription type (or globally).
- Allowing or blocking outbound presence notifications based on JID, group, or subscription type (or globally).
- Allowing or blocking IQ stanzas based on JID, group, or subscription type (or globally).
- Allowing or blocking all communications based on JID, group, or subscription type (or globally).

(from http://www.xmpp.org/specs/rfc3921.html#privacy)

Options:

`iqdisc` This specifies the processing discipline for Blocking Communication (`jabber:iq:privacy`) IQ queries (see section 5.2.1).

5.12 mod_private

This module adds support for Private XML Storage (JEP-0049[64]):

> Using this method, Jabber entities can store private data on the server and retrieve it whenever necessary. The data stored might be anything, as long as it is valid XML. One typical usage for this namespace is the server-side storage of client-specific preferences; another is Bookmark Storage (JEP-0048[65]).

Options:

`iqdisc` This specifies the processing discipline for Private XML Storage (`jabber:iq:private`) IQ queries (see section 5.2.1).

[64] http://www.jabber.org/jeps/jep-0049.html
[65] http://www.jabber.org/jeps/jep-0048.html

5.13 mod_pubsub

This module offers a Publish-Subscribe Service (JEP-0060[66]). Publish-Subscribe can be used to develop (examples are taken from the JEP):

- news feeds and content syndacation,
- avatar management,
- shared bookmarks,
- auction and trading systems,
- online catalogs,
- workflow systems,
- network management systems,
- NNTP gateways,
- vCard/profile management,
- and weblogs.

Another example is J-EAI[67]. This is an XMPP-based Enterprise Application Integration (EAI) platform (also known as ESB, the Enterprise Service Bus). The J-EAI project builts upon `ejabberd`'s codebase and has contributed several features to `mod_pubsub`.

Options:

`hosts` This option defines the hostnames of the service (see section 5.2.2). If neither `hosts` nor the old `host` is present, the prefix '`pubsub.`' is added to all `ejabberd` hostnames.

`served_hosts` To specify which hosts needs to be served, you can use this option. If absent, only the main `ejabberd` host is served.

`access_createnode` This option restricts which users are allowed to create pubsub nodes using ACL and ACCESS. The default value is `pubsub_createnode`.

Example:

```
{modules,
 [
  ...
  {mod_pubsub, [{served_hosts, ["example.com",
                                "example.org"]},
                {access_createnode, pubsub_createnode}]}
  ...
 ]}.
```

[66] http://www.jabber.org/jeps/jep-0060.html
[67] http://www.process-one.net/en/projects/j-eai/

5.14 mod_register

This module adds support for In-Band Registration (JEP-0077[68]). This protocol enables end users to use a Jabber client to:

- Register a new account on the server.
- Change the password from an existing account on the server.
- Delete an existing account on the server.

Options:

`access` This option can be configured to specify rules to restrict registration. If a rule returns 'deny' on the requested user name, registration for that user name is dennied. (there are no restrictions by default).

`iqdisc` This specifies the processing discipline for In-Band Registration (`jabber:iq:register`) IQ queries (see section 5.2.1).

Examples:

- Next example prohibits the registration of too short account names:

```
{acl, shortname, {user_glob, "?"}}.
{acl, shortname, {user_glob, "??"}}.
% The same using regexp:
%{acl, shortname, {user_regexp, "^..?$"}}.
...
{access, register, [{deny, shortname},
                    {allow, all}]}.
...
{modules,
 [
  ...
  {mod_register, [{access, register}]},
  ...
 ]}.
```

- The in-band registration of new accounts can be prohibited by changing the `access` option. If you really want to disable all In-Band Registration functionality, that is changing passwords in-band and deleting accounts in-band, you have to remove `mod_register` from the modules list. In this example all In-Band Registration functionality is disabled:

```
{access, register, [{deny, all}]}.

{modules,
```

[68] http://www.jabber.org/jeps/jep-0077.html

```
    [
     ...
%    {mod_register, [{access, register}]},
     ...
    ]}.
```

5.15 mod_roster

This module implements roster management as defined in RFC 3921: XMPP IM[69].

Options:

`iqdisc` This specifies the processing discipline for Roster Management (`jabber:iq:roster`) IQ queries (see section 5.2.1).

5.16 mod_service_log

This module adds support for logging end user packets via a Jabber message auditing service such as Bandersnatch[70]. All user packets are encapsulated in a `<route/>` element and sent to the specified service(s).

Options:

`loggers` With this option a (list of) service(s) that will receive the packets can be specified.

Examples:

- To log all end user packets to the Bandersnatch service running on `bandersnatch.example.com`:

```
{modules,
 [
  ...
  {mod_service_log, [{loggers, ["bandersnatch.example.com"]}]},
  ...
 ]}.
```

- To log all end user packets to the Bandersnatch service running on `bandersnatch.example.com` and the backup service on `bandersnatch.example.org`:

```
{modules,
 [
  ...
  {mod_service_log, [{loggers, ["bandersnatch.example.com",
                                "bandersnatch.example.org"]}]},
  ...
 ]}.
```

[69] http://www.xmpp.org/specs/rfc3921.html#roster
[70] http://www.funkypenguin.co.za/bandersnatch/

5.17 mod_shared_roster

This module enables you to create shared roster groups. This means that you can create groups of people that can see members from (other) groups in their rosters. The big advantages of this feature are that end users do not need to manually add all users to their rosters, and that they cannot permanently delete users from the shared roster groups.

Shared roster groups can be edited *only* via the web interface. Each group has a unique identification and the following parameters:

Name The name of the group, which will be displayed in the roster.

Description The description of the group. This parameter does not affect anything.

Members A list of full JIDs of group members, entered one per line in the web interface.

Displayed groups A list of groups that will be in the rosters of this group's members.

Examples:

- Take the case of a computer club that wants all its members seeing each other in their rosters. To achieve this, they need to create a shared roster group similar to next table:

Identification	Group 'club_members'
Name	Club Members
Description	Members from the computer club
Members	member1@example.org member2@example.org member3@example.org
Displayed groups	club_members

- In another case we have a company which has three divisions: Management, Marketing and Sales. All group members should see all other members in their rosters. Additonally, all managers should have all marketing and sales people in their roster. Simultaneously, all marketeers and the whole sales team should see all managers. This scenario can be achieved by creating shared roster groups as shown in the following table:

Identification	Group 'management'	Group 'marketing'	Group 'sales'
Name	Management	Marketing	Sales
Description			
Members	manager1@example.org manager2@example.org manager3@example.org manager4@example.org	marketeer1@example.org marketeer2@example.org marketeer3@example.org marketeer4@example.org	saleswoman1@example.org salesman1@example.org saleswoman2@example.org salesman2@example.org
Displayed groups	management marketing sales	management marketing	management sales

5.18 mod_stats

This module adds support for Statistics Gathering (JEP-0039[71]). This protocol allows you to retrieve next statistics from your `ejabberd` deployment:

- Total number of registered users on the current virtual host (users/total).
- Total number of registered users on all virtual hosts (users/all-hosts/total).
- Total number of online users on the current virtual host (users/online).
- Total number of online users on all virtual hosts (users/all-hosts/online).

Options:

`iqdisc` This specifies the processing discipline for Statistics Gathering (`http://jabber.org/protocol/stats`) IQ queries (see section 5.2.1).

As there are only a small amount of clients (for example Tkabber[72]) and software libraries with support for this JEP, a few examples are given of the XML you need to send in order to get the statistics. Here they are:

- You can request the number of online users on the current virtual host (`example.org`) by sending:

```
<iq to='example.org' type='get'>
  <query xmlns='http://jabber.org/protocol/stats'>
    <stat name='users/online'/>
  </query>
</iq>
```

- You can request the total number of registered users on all virtual hosts by sending:

```
<iq to='example.org' type='get'>
  <query xmlns='http://jabber.org/protocol/stats'>
    <stat name='users/all-hosts/total'/>
  </query>
</iq>
```

5.19 mod_time

This module features support for Entity Time (JEP-0090[73]). By using this JEP, you are able to discover the time at another entity's location.

Options:

`iqdisc` This specifies the processing discipline for Entity Time (`jabber:iq:time`) IQ queries (see section 5.2.1).

[71] http://www.jabber.org/jeps/jep-0039.html
[72] http://tkabber.jabber.ru/
[73] http://www.jabber.org/jeps/jep-0090.html

5.20 mod_vcard

This module allows end users to store and retrieve their vCard, and to retrieve other users vCards, as defined in vcard-temp (JEP-0054[74]). The module also implements an uncomplicated Jabber User Directory based on the vCards of these users. Moreover, it enables the server to send its vCard when queried.

Options:

`hosts` This option defines the hostnames of the service (see section 5.2.2). If neither `hosts` nor the old `host` is present, the prefix '`vjud.`' is added to all `ejabberd` hostnames.

`iqdisc` This specifies the processing discipline for `vcard-temp` IQ queries (see section 5.2.1).

`search` This option specifies whether the search functionality is enabled (value: `true`) or disabled (value: `false`). If disabled, the option `hosts` will be ignored and the Jabber User Directory service will not appear in the Service Discovery item list. The default value is `true`.

`matches` With this option, the number of reported search results can be limited. If the option's value is set to `infinity`, all search results are reported. The default value is `30`.

`allow_return_all` This option enables you to specify if search operations with empty input fields should return all users who added some information to their vCard. The default value is `false`.

`search_all_hosts` If this option is set to `true`, search operations will apply to all virtual hosts. Otherwise only the current host will be searched. The default value is `true`.

Examples:

- In this first situation, search results are limited to twenty items, every user who added information to their vCard will be listed when people do an empty search, and only users from the current host will be returned:

```
{modules,
 [
  ...
  {mod_vcard, [{search, true},
               {matches, 20},
               {allow_return_all, true},
               {search_all_hosts, false}]},
  ...
 ]}.
```

- The second situation differs in a way that search results are not limited, and that all virtual hosts will be searched instead of only the current one:

[74] http://www.jabber.org/jeps/jep-0054.html

```
{modules,
 [
  ...
  {mod_vcard, [{search, true},
               {matches, infinity},
               {allow_return_all, true}]},
  ...
 ]}.
```

5.21 mod_vcard_ldap

ejabberd can map LDAP attributes to vCard fields. This behaviour is implemented in the mod_vcard_ldap module. This module does not depend on the authentication method (see 4.5.2). The mod_vcard_ldap module has its own optional parameters. The first group of parameters has the same meaning as the top-level LDAP parameters to set the authentication method: ldap_servers, ldap_port, ldap_rootdn, ldap_password, ldap_base, ldap_uidattr, ldap_uidattr_format and ldap_filter. See section 4.5.2 for detailed information about these options. If one of these options is not set, ejabberd will look for the top-level option with the same name. The second group of parameters consists of the following mod_vcard_ldap-specific options:

hosts This option defines the hostnames of the service (see section 5.2.2). If neither hosts nor the old host is present, the prefix 'vjud.' is added to all ejabberd hostnames.

iqdisc This specifies the processing discipline for vcard-temp IQ queries (see section 5.2.1).

search This option specifies whether the search functionality is enabled (value: true) or disabled (value: false). If disabled, the option hosts will be ignored and the Jabber User Directory service will not appear in the Service Discovery item list. The default value is true.

ldap_vcard_map With this option you can set the table that maps LDAP attributes to vCard fields. The format is: [Name_of_vCard_field, Pattern, List_of_LDAP_attributes, ...]. Name_of_vcard_field is the type name of the vCard as defined in RFC 2426[75]. Pattern is a string which contains pattern variables "%u", "%d" or "%s". List_of_LDAP_attributes is the list containing LDAP attributes. The pattern variables "%s" will be sequentially replaced with the values of LDAP attributes from List_of_LDAP_attributes, "%u" will be replaced with the user part of a JID, and "%d" will be replaced with the domain part of a JID. The default is:

```
[{"NICKNAME", "%u", []},
 {"FN", "%s", ["displayName"]},
 {"FAMILY", "%s", ["sn"]},
 {"GIVEN", "%s", ["givenName"]},
 {"MIDDLE", "%s", ["initials"]},
 {"ORGNAME", "%s", ["o"]},
 {"ORGUNIT", "%s", ["ou"]},
 {"CTRY", "%s", ["c"]},
 {"LOCALITY", "%s", ["l"]},
```

[75] http://www.ietf.org/rfc/rfc2426.txt

```
 {"STREET", "%s", ["street"]},
 {"REGION", "%s", ["st"]},
 {"PCODE", "%s", ["postalCode"]},
 {"TITLE", "%s", ["title"]},
 {"URL", "%s", ["labeleduri"]},
 {"DESC", "%s", ["description"]},
 {"TEL", "%s", ["telephoneNumber"]},
 {"EMAIL", "%s", ["mail"]},
 {"BDAY", "%s", ["birthDay"]},
 {"ROLE", "%s", ["employeeType"]},
 {"PHOTO", "%s", ["jpegPhoto"]}]
```

`ldap_search_fields` This option defines the search form and the LDAP attributes to search within. The format is: `[Name, Attribute, ...]`. `Name` is the name of a search form field which will be automatically translated by using the translation files (see `msgs/*.msg` for available words). `Attribute` is the LDAP attribute or the pattern `"%u"`. The default is:

```
[{"User", "%u"},
 {"Full Name", "displayName"},
 {"Given Name", "givenName"},
 {"Middle Name", "initials"},
 {"Family Name", "sn"},
 {"Nickname", "%u"},
 {"Birthday", "birthDay"},
 {"Country", "c"},
 {"City", "l"},
 {"Email", "mail"},
 {"Organization Name", "o"},
 {"Organization Unit", "ou"}]
```

`ldap_search_reported` This option defines which search fields should be reported. The format is: `[Name, vCard_Name, ...]`. `Name` is the name of a search form field which will be automatically translated by using the translation files (see `msgs/*.msg` for available words). `vCard_Name` is the vCard field name defined in the `ldap_vcard_map` option. The default is:

```
[{"Full Name", "FN"},
 {"Given Name", "GIVEN"},
 {"Middle Name", "MIDDLE"},
 {"Family Name", "FAMILY"},
 {"Nickname", "NICKNAME"},
 {"Birthday", "BDAY"},
 {"Country", "CTRY"},
 {"City", "LOCALITY"},
 {"Email", "EMAIL"},
 {"Organization Name", "ORGNAME"},
 {"Organization Unit", "ORGUNIT"}]
```

Examples:

- Let's say `ldap.example.org` is the name of our LDAP server. We have users with their passwords in `"ou=Users,dc=example,dc=org"` directory. Also we have addressbook, which contains users emails and their additional infos in `"ou=AddressBook,dc=example,dc=org"` directory. Corresponding authentication section should looks like this:

```
%% authentication method
{auth_method, ldap}.
%% DNS name of our LDAP server
{ldap_servers, ["ldap.example.org"]}.
%% We want to authorize users from 'shadowAccount' object class only
{ldap_filter, "(objectClass=shadowAccount)"}.
```

  Now we want to use users LDAP-info as their vCards. We have four attributes defined in our LDAP schema: `"mail"` — email address, `"givenName"` — first name, `"sn"` — second name, `"birthDay"` — birthday. Also we want users to search each other. Let's see how we can set it up:

```
{modules,
  ...
  {mod_vcard_ldap,
   [
    %% We use the same server and port, but want to bind anonymously because
    %% our LDAP server accepts anonymous requests to
    %% "ou=AddressBook,dc=example,dc=org" subtree.
    {ldap_rootdn, ""},
    {ldap_password, ""},
    %% define the addressbook's base
    {ldap_base, "ou=AddressBook,dc=example,dc=org"},
    %% user's part of JID is located in the "mail" attribute
    {ldap_uidattr, "mail"},
    %% common format for our emails
    {ldap_uidattr_format, "%u@mail.example.org"},
    %% We have to define empty filter here, because entries in addressbook does not
    %% belong to shadowAccount object class
    {ldap_filter, ""},
    %% Now we want to define vCard pattern
    {ldap_vcard_map,
     [{"NICKNAME", "%u", []}, % just use user's part of JID as his nickname
      {"GIVEN", "%s", ["givenName"]},
      {"FAMILY", "%s", ["sn"]},
      {"FN", "%s, %s", ["sn", "givenName"]}, % example: "Smith, John"
      {"EMAIL", "%s", ["mail"]},
      {"BDAY", "%s", ["birthDay"]}]},
    %% Search form
    {ldap_search_fields,
     [{"User", "%u"},
      {"Name", "givenName"},
      {"Family Name", "sn"},
      {"Email", "mail"},
```

```
      {"Birthday", "birthDay"}]},
    %% vCard fields to be reported
    %% Note that JID is always returned with search results
    {ldap_search_reported,
     [{"Full Name", "FN"},
      {"Nickname", "NICKNAME"},
      {"Birthday", "BDAY"}]}
  ]}
  ...
}.
```

Note that `mod_vcard_ldap` module checks an existence of the user before searching his info in LDAP.

- `ldap_vcard_map` example:

```
{ldap_vcard_map,
 [{"NICKNAME", "%u", []},
  {"FN", "%s", ["displayName"]},
  {"CTRY", "Russia", []},
  {"EMAIL", "%u@%d", []},
  {"DESC", "%s\n%s", ["title", "description"]}
 ]},
```

- `ldap_search_fields` example:

```
{ldap_search_fields,
 [{"User", "uid"},
  {"Full Name", "displayName"},
  {"Email", "mail"}
 ]},
```

- `ldap_search_reported` example:

```
{ldap_search_reported,
 [{"Full Name", "FN"},
  {"Email", "EMAIL"},
  {"Birthday", "BDAY"},
  {"Nickname", "NICKNAME"}
 ]},
```

5.22 mod_version

This module implements Software Version (JEP-0092[76]). Consequently, it answers `ejabberd`'s version when queried.

Options:

[76] http://www.jabber.org/jeps/jep-0092.html

`iqdisc` This specifies the processing discipline for Software Version (`jabber:iq:version`) IQ queries (see section 5.2.1).

6 Creating an Initial Administrator

Before the web interface can be entered to perform administration tasks, an account with administrator rights is needed on your `ejabberd` deployment.

Instructions to create an initial administrator account:

1. Register an account on your `ejabberd` deployment. An account can be created in two ways:
   (a) Using the tool `ejabberdctl` (see section 7.2):

   ```
   % ejabberdctl node@host register admin example.org password
   ```

   (b) Using In-Band Registration (see section 5.14): you can use a Jabber client to register an account.
2. Edit the configuration file to promote the account created in the previous step to an account with administrator rights. Note that if you want to add more administrators, a seperate acl entry is needed for each administrator.

   ```
   {acl, admins, {user, "admin", "example.org"}}.
   {access, configure, [{allow, admins}]}.
   ```

3. Restart `ejabberd` to load the new configuration.
4. Open the web interface (`http://server:port/admin/`) in your favourite browser. Make sure to enter the *full* JID as username (in this example: `admin@example.org`. The reason that you also need to enter the suffix, is because `ejabberd`'s virtual hosting support.

7 Online Configuration and Monitoring

7.1 Web Interface

To perform online configuration of `ejabberd` you need to enable the `ejabberd_http` listener with the option `web_admin` (see section 3.3). Then you can open `http://server:port/admin/` in your favourite web browser. You will be asked to enter the username (the *full* Jabber ID) and password of an `ejabberd` user with administrator rights. After authentication you will see a page similar to figure 1.

Here you can edit access restrictions, manage users, create backups, manage the database, enable/disable ports listened for, view server statistics,...

Examples:

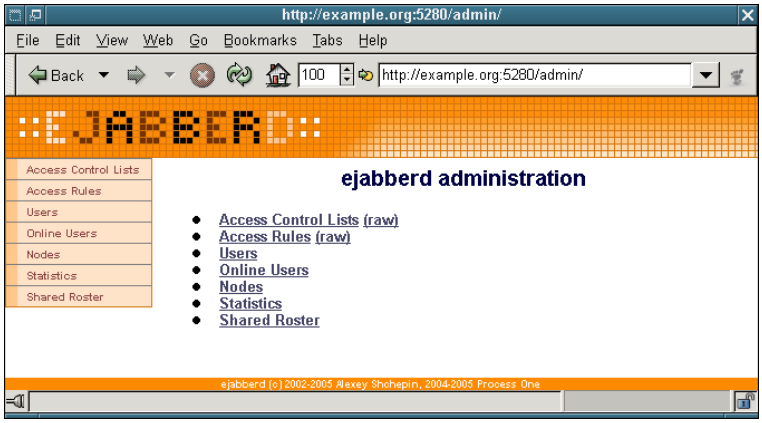

Figure 1: Top page from the web interface

- You can serve the web interface on the same port as the HTTP Polling interface. In this example you should point your web browser to `http://example.org:5280/admin/` to administer all virtual hosts or to `http://example.org:5280/admin/server/example.com/` to administer only the virtual host `example.com`. Before you get access to the web interface you need to enter as username, the JID and password from a registered user that is allowed to configure `ejabberd`. In this example you can enter as username '`admin@example.net`' to administer all virtual hosts (first URL). If you log in with '`admin@example.com`' on `http://example.org:5280/admin/server/example.com/` you can only administer the virtual host `example.com`.

```
...
{acl, admins, {user, "admin", "example.net"}}.
{host_config, "example.com", [{acl, admins, {user, "admin", "example.com"}}]}.
{access, configure, [{allow, admins}]}.
...
{hosts, ["example.org"]}.
...
{listen,
 [...
  {5280, ejabberd_http, [http_poll, web_admin]},
  ...
 ]
}.
```

- For security reasons, you can serve the web interface on a secured connection, on a port differing from the HTTP Polling interface, and bind it to the internal LAN IP. The web interface will be accessible by pointing your web browser to `https://192.168.1.1:5280/admin/`:

```
...
{hosts, ["example.org"]}.
...
{listen,
 [...
```

```
        {5270, ejabberd_http,    [http_poll]},
        {5280, ejabberd_http,    [web_admin, {ip, {192, 168, 1, 1}},
                                  tls, {certfile, "/usr/local/etc/server.pem"}]},
        ...
       ]
      }.
```

7.2 ejabberdctl

It is possible to do some administration operations using the command line tool `ejabberdctl`. You can list all available options by running `ejabberdctl` without arguments:

```
% ejabberdctl
Usage: ejabberdctl node command

Available commands:
  status                        get ejabberd status
  stop                          stop ejabberd
  restart                       restart ejabberd
  reopen-log                    reopen log file
  register user server password register a user
  unregister user server        unregister a user
  backup file                   store a database backup to file
  restore file                  restore a database backup from file
  install-fallback file         install a database fallback from file
  dump file                     dump a database to a text file
  load file                     restore a database from a text file
  import-file file              import user data from jabberd 1.4 spool file
  import-dir dir                import user data from jabberd 1.4 spool directory
  registered-users              list all registered users
  delete-expired-messages       delete expired offline messages from database

Example:
  ejabberdctl ejabberd@host restart
```

Additional information:

`reopen-log` If you use a tool to rotate logs, you have to configure it so that this command is executed after each rotation.

`backup, restore, install-fallback, dump, load` You can use these commands to create and restore backups.

`import-file, import-dir` These options can be used to migrate from other Jabber/XMPP servers. There exist tutorials to migrate from other software to ejabberd[77].

`delete-expired-messages` This option can be used to delete old messages in offline storage. This might be useful when the number of offline messages is very high.

[77] http://ejabberd.jabber.ru/migrate-to-ejabberd

8 Firewall Settings

You need to take the following TCP ports in mind when configuring your firewall:

Port	Description
5222	SASL and unencrypted c2s connections.
5223	Obsolete SSL c2s connections.
5269	s2s connections.
4369	Only for clustering (see 10).
port range	Only for clustring (see 10). This range is configurable (see 2.4).

9 SRV Records

- General information: SRV record[78]
- Practical information: Setting DNS SRV Records[79]

10 Clustering

10.1 How it Works

A Jabber domain is served by one or more `ejabberd` nodes. These nodes can be run on different machines that are connected via a network. They all must have the ability to connect to port 4369 of all another nodes, and must have the same magic cookie (see Erlang/OTP documentation, in other words the file `~ejabberd/.erlang.cookie` must be the same on all nodes). This is needed because all nodes exchange information about connected users, s2s connections, registered services, etc...

Each `ejabberd` node has the following modules:

- router,
- local router,
- session manager,
- s2s manager.

[78] http://en.wikipedia.org/wiki/SRV_record
[79] http://jabberd.jabberstudio.org/2/docs/section05.html#5_7

10.1.1 Router

This module is the main router of Jabber packets on each node. It routes them based on their destination's domains. It uses a global routing table. The domain of the packet's destination is searched in the routing table, and if it is found, the packet is routed to the appropriate process. If not, it is sent to the s2s manager.

10.1.2 Local Router

This module routes packets which have a destination domain equal to one of this server's host names. If the destination JID has a non-empty user part, it is routed to the session manager, otherwise it is processed depending on its content.

10.1.3 Session Manager

This module routes packets to local users. It looks up to which user resource a packet must be sent via a presence table. Then the packet is either routed to the appropriate c2s process, or stored in offline storage, or bounced back.

10.1.4 s2s Manager

This module routes packets to other Jabber servers. First, it checks if an opened s2s connection from the domain of the packet's source to the domain of the packet's destination exists. If that is the case, the s2s manager routes the packet to the process serving this connection, otherwise a new connection is opened.

10.2 Clustering Setup

Suppose you already configured `ejabberd` on one machine named (`first`), and you need to setup another one to make an `ejabberd` cluster. Then do following steps:

1. Copy `~ejabberd/.erlang.cookie` file from `first` to `second`.

   (alt) You can also add '`-cookie content_of_.erlang.cookie`' option to all '`erl`' commands below.

2. On `second` run the following command as the `ejabberd` daemon user, in the working directory of `ejabberd`:

```
erl -sname ejabberd \
    -mnesia extra_db_nodes "['ejabberd@first']" \
    -s mnesia
```

This will start Mnesia serving the same database as `ejabberd@first`. You can check this by running the command '`mnesia:info().`'. You should see a lot of remote tables and a line like the following:

```
running db nodes   = [ejabberd@first, ejabberd@second]
```

3. Now run the following in the same '`erl`' session:

   ```
   mnesia:change_table_copy_type(schema, node(), disc_copies).
   ```

   This will create local disc storage for the database.

   (alt) Change storage type of the `scheme` table to 'RAM and disc copy' on the second node via the web interface.

4. Now you can add replicas of various tables to this node with '`mnesia:add_table_copy`' or '`mnesia:change_table_copy_type`' as above (just replace '`schema`' with another table name and '`disc_copies`' can be replaced with '`ram_copies`' or '`disc_only_copies`').

   Which tables to replicate is very dependant on your needs, you can get some hints from the command '`mnesia:info().`', by looking at the size of tables and the default storage type for each table on 'first'.

   Replicating a table makes lookups in this table faster on this node. Writing, on the other hand, will be slower. And of course if machine with one of the replicas is down, other replicas will be used.

   Also section 5.3 (Table Fragmentation) of Mnesia User's Guide[80] can be helpful.

   (alt) Same as in previous item, but for other tables.

5. Run '`init:stop().`' or just '`q().`' to exit from the Erlang shell. This probably can take some time if Mnesia has not yet transfered and processed all data it needed from `first`.

6. Now run `ejabberd` on `second` with almost the same config as on `first` (you probably do not need to duplicate '`acl`' and '`access`' options — they will be taken from `first`, and `mod_muc` and `mod_irc` should be enabled only on one machine in the cluster).

You can repeat these steps for other machines supposed to serve this domain.

A Internationalization and Localization

All built-in modules support the `xml:lang` attribute inside IQ queries. Figure 2, for example, shows the reply to the following query:

```
<iq id='5'
    to='example.org'
    type='get'
    xml:lang='ru'>
  <query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

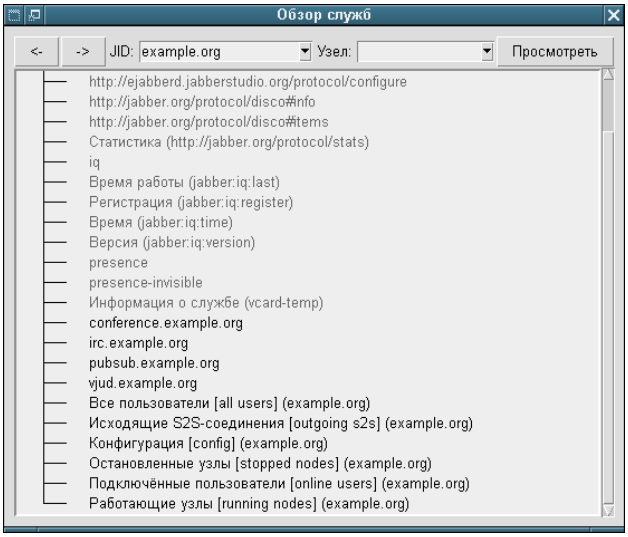

Figure 2: Service Discovery when `xml:lang='ru'`

The web interface also supports the `Accept-Language` HTTP header (compare figure 3 with figure 1)

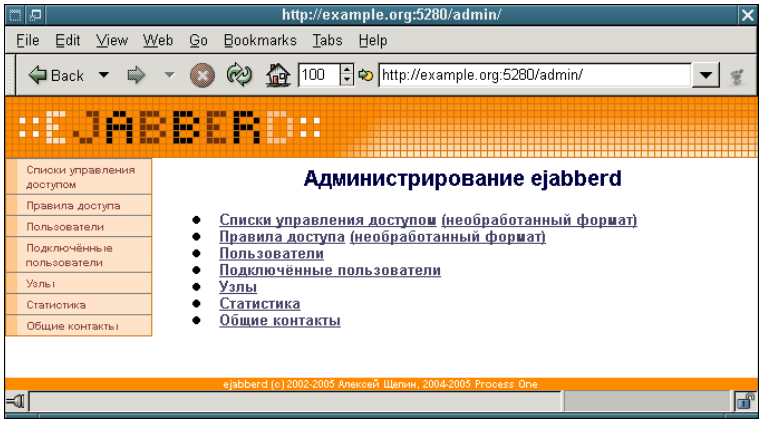

Figure 3: Top page from the web interface with HTTP header 'Accept-Language: ru'

[80] http://www.erlang.se/doc/doc-5.4.9/lib/mnesia-4.2.2/doc/html/Mnesia_chap5.html#5.3

B Release Notes

Release notes are available from ejabberd Home Page[81]

C Acknowledgements

Thanks to all people who contributed to this guide:

- Alexey Shchepin (xmpp:aleksey@jabber.ru)
- Badlop (xmpp:badlop@jabberes.org)
- Evgeniy Khramtsov (xmpp:xram@jabber.ru)
- Florian Zumbiehl (xmpp:florz@florz.de)
- Michael Grigutsch (xmpp:migri@jabber.i-pobox.net)
- Mickael Remond (xmpp:mremond@erlang-projects.org)
- Sander Devrieze (xmpp:sander@devrieze.dyndns.org)
- Sergei Golovan (xmpp:sgolovan@nes.ru)
- Vsevolod Pelipas (xmpp:vsevoload@jabber.ru)

D Copyright Information

[81] http://www.process-one.net/en/ejabberd/release_notes/

Index