Table of Contents

Draft Sections:

Appendices (including Quickstart Guide):

***N** Note: Use Back Button to Return to Top*

In this single page guide, the links at the section ends actually link back to individual section pages. This is a bug due to the fact that this page is a concatenation of the indivudual section pages. Please use your browser "Back" button to return to the top of this document.

1. Introduction

This document comprises the installation and administrative guide for the Jabber 2 server (Jabberd), the latest release of the popular open source messaging system based on the Jabber Protocol. The goal of Jabber is to provide an XML protocol for synchronous and asynchronous communication for client to client, client to server, and server to server messaging, although the primary use for Jabber is instant messaging (IM).

The Jabberd server is the original open−source server implementation of the Jabber protocol, and it remains the most popular software for deploying Jabber either inside a company or as a public IM service. Jeremie Miller initiated the Jabber Project in 1998 as a free and open alternative to proprietary IM services. The Jabberd server continues to be the core of the Jabber Project, and Jabberd 2 is the successor to the widely used Jabberd 1.4 server. Jabberd 2 is based on a completely new code base with a new architecture, additional features and improved adherence to the Jabber protocol.

The creation of a common messaging protocol, now known as XMPP (Extensible Messaging and Presence Protocol), has allowed for the creation of numerous Jabber server implementations in addition to the Jabberd server. Among these are several open source projects, including WP Jabber and ejabberd, as well as several commercial offerings from companies such as i3connect, Jabcast, Tipic and Jabber, Inc.

1.1. Purpose and Scope

The authors of this document intend to provide a complete guide for jabberd 2 installation, administration and development:

- Architecture Overview
- System Preparation
- Server Installation
- Server Configuration

The intended audience are people who wish to install and/or maintain a jabberd 2 server on Unix, or one of its variants. As such, this document covers jabberd installation on Unix operating systems only.

1.2. Required Background

The authors have made every attempt to make this a step−by−step guide; however, some familiarity with a Unix or Linux operating system is assumed:

- Ability to use a Linux command−prompt console
- Familiarity with the file system on which your server is to be installed
- Familiarity with a text editor, such as vi, Nano or NEdit
- Basic ability to edit XML files

It is assumed that the reader has a basic familiarity with using a Jabber client. Additionally, it is assumed that the reader is familiar with hardware and software, such as a firewall, router or modem, that the jabber server will use to connect to the Internet if any such hardware is used. Configuration of these secondary programs and devices is beyond the scope of this guide.

1.3. How to Use This Document

This guide is organized into sections grouped according to intended use by the user:

- Jabberd Quickstart Guide (Appendix A)
- Jabber 2 Architecture (Section 2)
- Installing Jabberd 2 Server (Sections 3−5)
- Common Configuration and Administration Tasks (Sections 6−7)
- Detailed Configuration Guide (Sections 8−13)

The Quickstart Guide is designed to get experienced users up and running quickly. A detailed reading begins with Section 2, Jabber 2 Architecture, which provides background information about Jabber and the Jabberd server. This section can be skipped by those who wish to jump right into the detailed Jabberd installation instructions that begin in Section 3, Preparing to Install Jabberd 2. Sections 6 and 7 list common Configuration and Administration tasks, respectively, while the remaining sections provide detailed configuration information.

1.4. Conventions Used in this Document

This document is provided primarily as an installation guide, and as such, special conventions are used to make installation easier for the user. The conventions below appear throughout the installation sections of this guide:

Table 1.4. Document Conventions Used in this Guide

Convention	Name	Description
P	Parameter	Information about your specific setup that you will enter into a configuration file, etc.
C	Checkpoint	A point at which to stop and check your setup.
N	Note	An informational note.
I	Important	An important note or warning.
F	Required Files	Software or specific files needed to complete a step or series of steps.
O	Optional Step	A step that is not required for the most basic installation.
E	External System	A step that may require configuration on an external system, such as a router.

The most useful of these conventions are "Parameters." Section 3 begins with a list of information about your setup (Parameters) that you will need during the installation steps. You can gather all this information before you start installing Jabberd, and then you can refer back to your list for every step that displays a ***P***.

The installation guide is organized into numbered steps and sub−steps, etc. Users are encouraged to use the guide as a check list. When all of the sub−steps of a step are completed, then the parent step itself is also completed. Note that all children (sub−steps) of an "Optional Step" are optional also. Note that the "Optional Step" designation provides information about the conditions and/or requirements under which the optional step should be performed.

Steps labeled with "External System" provide the user with information about set up that may need to be performed on a system external to jabberd. These systems include routers, firewalls, DNS servers, etc. "External System" notes are intended to be informational rather than comprehensive. The remaining conventions are self−explanatory.

Note that each command listed in this guide refers to a command entered at a command prompt or at a command prompt shell, such as X−term or E−term. Note also that in this document, the term "Jabberd" refers to the Jabberd 2 server, except where noted. The term "Jabber" refers to Jabber−based system or systems, and "XMPP" refers to the protocol over which Jabber systems run.

This document was created using Structured Text. Structured Text uses standard text formatting conventions, such as underscores, to represent formatted text. Efforts have been made to keep the text and HTML representations close. One exception is the use of a bang character ("!") in the text version to escape unwanted formatting in the HTML version. "Bangs" appear at the beginning of some lines in the text version, and these can be ignored. They do not appear in the HTML version.

1.5. Further Reading

Visit the Jabber Software Foundation for the latest news about jabberd 2, jabber clients, and the Jabber Protocol. The Jabber Faq answers basic questions about Jabber. Readers are encouraged to visit the Jadmin Archive for questions about jabber administration, or subscribe to the Jadmin Mailing List for the most up to date jabberd administration information. Jabberd 2 development information can be found in the Jabberd Archive or by subscribing to the Jabberd List.

There are also several good books about Jabber. Note that books below detail *jabberd 1.4 only* as of this writing in 2003:

- Jabberd Administration Guide for version 1.4
- Programming Jabber by D.J. Adams
- Instant Messaging for Java by Iain Shigeoka
- Jabber Programming by Stephen Lee and Terence Smelser
- Jabber Developer's Handbook from Sams Publishing

1.6. Legalese

1.7. Sources

Robert Norris, who is the primary Jabberd 2 developer, and the posters on the Jabberd List provided technical information for this document.

View as PDF

Back | Up | Next

2. Jabberd 2 Architecture

***N** Note: This Section Is Work in Progress*

Note that this section is largely a work in process. Thus, information below should be regarded as draft documentation, and *not* as a finalized description of Jabberd 2 architecture. Essentially, I am adding to this section as I learn about Jabberd 2.

|| TODO: Add architecture introduction consisting of architecture feature list −or− just new feature list ||

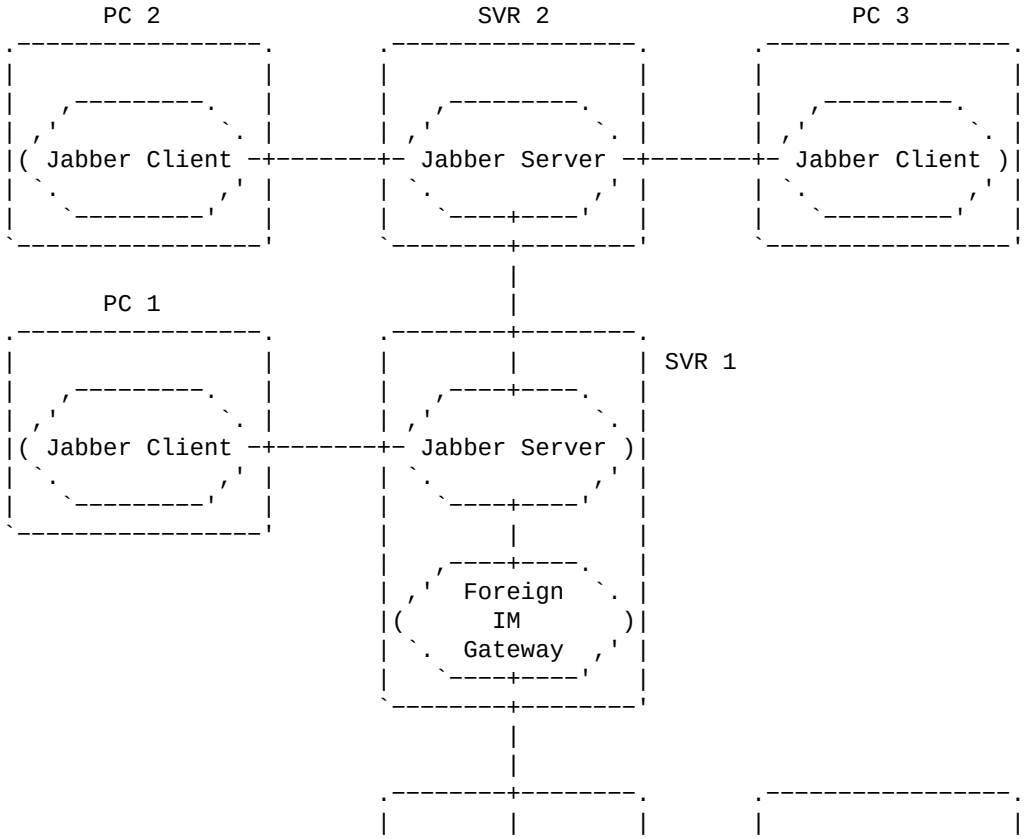
2.1. Jabber Network Architecture

Jabber, in the form of the XMPP protocol, provides a protocol for messaging, and as such it provides a standardized platform for Jabber server communication:

- Jabber Client to Jabber Server
- Jabber Server to Jabber Server
- Jabber Server to Foreign IM Gateway

Figures 2.1 and 2.2 below demonstrate how these three types of communication can occur.

***Figure 2.1.1. Jabber Deployment Diagram (High Level View)**:*



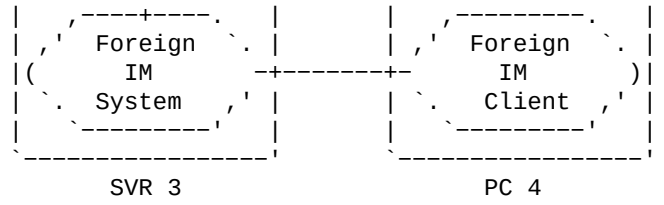
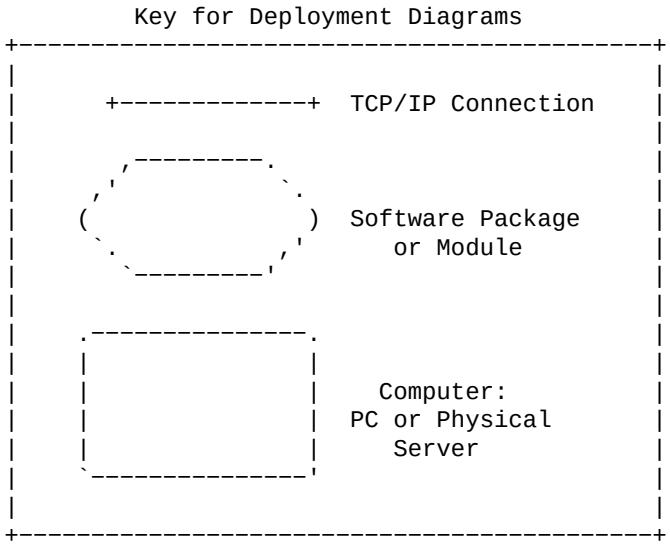


Figure 2.1. Adapted from XMPP Core Draft

Figure 2.1.2. Key to Deployment Diagrams:



2.1.1. Jabber Client to Jabber Server

In the diagram above, Jabber Clients on *PC 2* and *PC 3* are able to communicate via Jabber IM provided by *SVR 2*. Both clients have accounts on this server, and this part of the diagram, if taken by itself, would represent a *closed* or *private* Jabber system.

Note that a *Jabber Client* is not necessarily a user−based IM client. For example, the client running on *PC 3* might actually be part of a web server. In this manner, a user on *PC 1* might make updates to the web server by using a Jabber client.

2.2.2. Jabber Server to Jabber Server

Communication between *SVR 1* and *SVR 2* demonstrates how Jabber employs a distributed architecture. Clients on *PC 1* and *PC 2* are able to communicate with each other even though these clients have accounts on separate servers. These clients do not need to know anything about the remote servers. Instead, each client needs only to know the address of the client with which it wishes to communicate.

Jabber servers running on *SVR 1* and *SVR 2* rely on the Domain Name Service (DNS) for address lookup in order to communicate with each other. In this manner, the Jabber IM system resembles the email network architecture provided by POP and SMTP, the most widely used email protocols on the Internet. The ability to resolve names of other jabber servers and to route messaging makes jabber unlike proprietary IM systems. Jabber does not rely on a centralized server farm. Thus, Jabber is easily scalable, and it can be used as a closed or open system.

2.1.2. Jabber Server to Foreign IM Gateway

In this diagram, *SVR 1* communicates with a *Foreign IM Gateway* running on the same machine. This gateway is able to communicate with a *Foreign IM System*, such as AOL, MSN, Yahoo or IRC. Connection with this communication gateway allows clients on *PC 1* and *PC 4* to communicate despite the fact that the user on *PC 1* is running Jabber client software, while the user on *PC 4* is running software, such as AIM, that uses a completely different messaging protocol.

The *foreign IM* example demonstrates the flexibility that Jabber provides. The XMPP protocols dictate set of XML−based communication standards. Therefore, a gateway for other protocols can be created, provided that the details of such a non−XMPP protocol are known.

2.2. Jabberd 2 Component Architecture

N *Note: Subsequent Documentation is Jabberd 2 Specific*

> The preceding section provides an overview of how the Jabber architecture works. Other Jabber servers, in addition to Jabberd 2, provide the same services. However, the documentation in the subsequent sections deals with Jabberd 2 specifically.

The beauty of the Jabberd 2 architecture lies in the fact that its component architecture distributes services across five components, each of which communicates over TCP/IP:

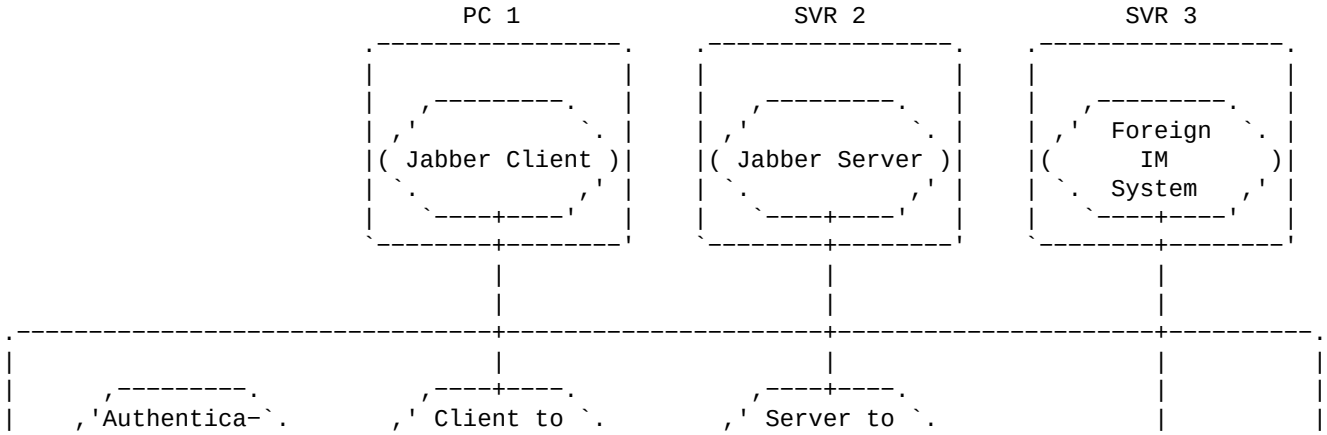
- Router
- Server to Server (S2S)
- Resolver
- Session Manager (SM)
- Client to Server

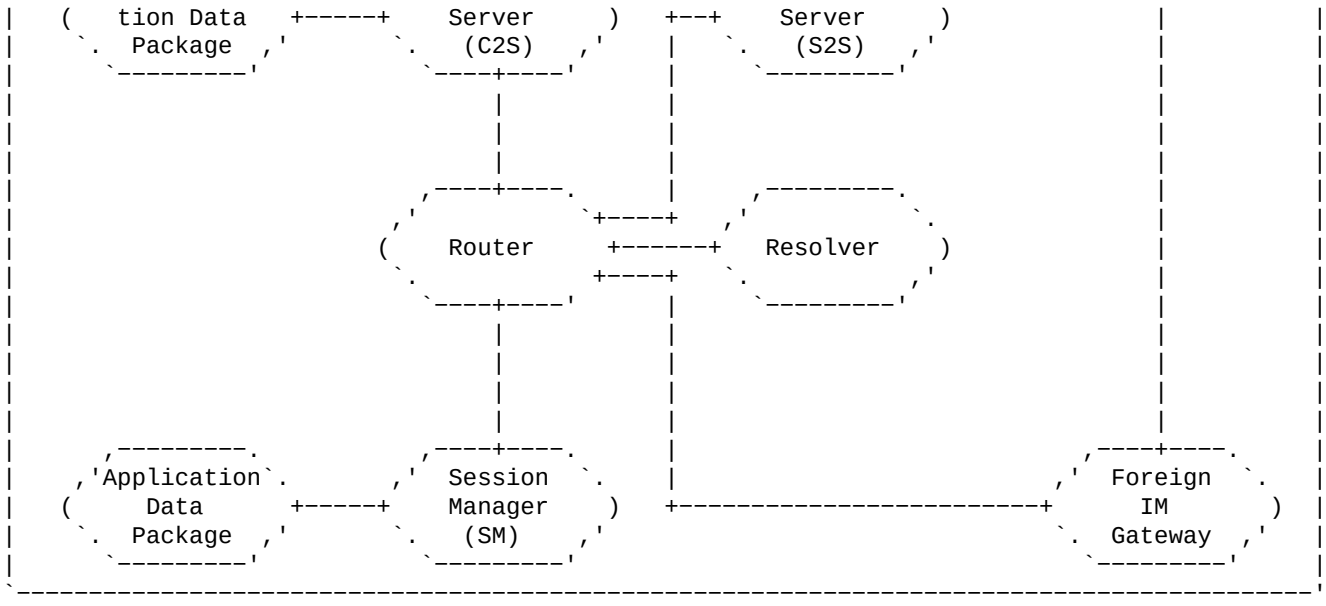
Jabberd 2 also relies on third−party components:

- Application Data Store
- Authentication Data Store
- Foreign IM Gateway(s)

Figure 2.2 shows how these components communicate and how they are generally located on a single physical server.

Figure 2.2. Jabberd Deployment Diagram (Component Level View):





N *Note: Jabberd 2 Relies on External Packages*

The diagram above shows all the components required for a functioning Jabberd 2 server in addition to one optional component, a *Foreign IM Gateway*. Foreign IM Gateways are not included with the Jabberd 2 distribution. Packages for authentication and application data are also not included with the Jabberd 2 Distribution. Instead, Jabberd 2 connects to third−party data and authentication packages that must be installed in order for Jabberd to function.

Figure 2.2.1 shows how Jabberd 2 distributes functions across five separate modules. Just as the data stores and gateway(s) may be located on separate machines, the Jabberd 2 components may be easily distributed across machines because each component consists of a configuration file and a binary executable that communicates via TCP/IP. This architecture allows the server to scale. When load on a Jabberd 2 server becomes high, component(s) can be moved to separate machines.

2.2.1. Router

The *Router* is the backbone of the Jabberd server. It accepts connections from Jabberd components and passes XML packets between components.

2.2.2. S2S

The *S2S* (Server to Server) component handles communications with external servers. *S2S* passes packets between other components and external servers, and it performs dialback to authenticate remote Jabber servers.

2.2.3. Resolver

The *Resolver* acts in support of the *S2S* component. The *Resolver* resolves hostnames for *S2S* as part of dialback authentication.

2.2.4. SM

The *SM* (Session Manager) component implements the bulk of the instant messaging features:

- Message passing
- Presence
- Rosters
- Subscriptions

The *SM* component connects to the "Application Data Package *(db) in order to provide persistant data storage. Additionally, the* SM *component handles the Jabber extensions of* disco *(discovery) and* privacy lists*.

2.2.5. C2S

The *C2S* (Client to Server) component handles communication with Jabber clients:

- Connects to Jabber clients
- Passes packets to the SM
- Authenticates clients
- Registers users
- Triggers activity with the SM

The *C2S* component connects to the *Authentication Data Package* (authreg) in order to register and authenticate users.

2.3. Jabberd 2 Module Decomposition

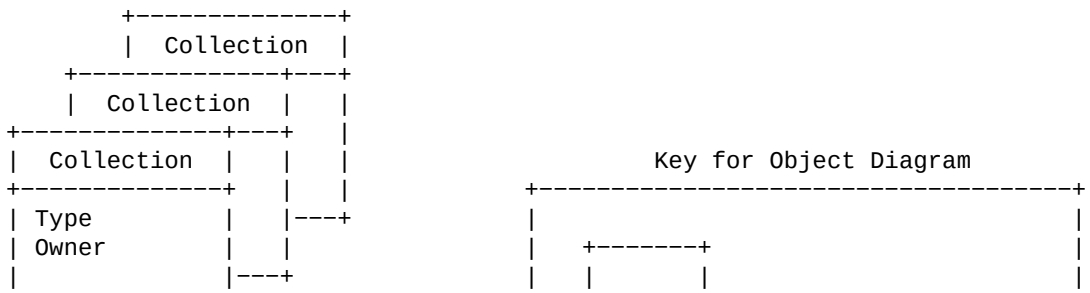
|| TODO: Section TBD ||

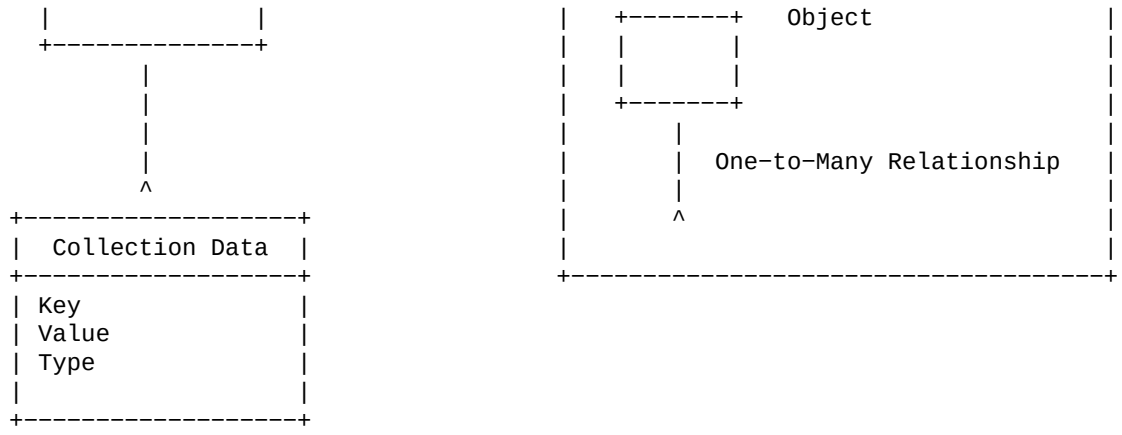
2.4. Jabberd 2 Data Handling

Jabberd employs data handling that allows for mapping to various data packages. The concept of a *collection* object lies at the core of Jabberd data handling. Each *collection* object has the attributes of *type* and *owner*. *Type* specifies what category of data is being handled, such as *queue*, *vcard* or *roster−items*. *Owner* specifies who owns the *collection*. For user−related data, *Owner* is a jabber ID (JID).

Each *collection* holds one or more data objects. Each of these data objects is a tuple consisting of a *key*, *value* and *type*. *Key* is a string that specifies the kind of data being held. *Value* is the data being held, and *type* is the is the data type, i.e. Boolean, integer, string, etc. The diagram below shows how these objects are related.

Figure 2.4. Jabberd Data Objects:





2.5. Jabberd 2 Data Structure (for MySQL)

Jabberd Data handling becomes clearer as we see it applied to the MySQL data package. In the relational world, each *collection type* becomes a table, and each *collection data key* becomes a table field. Thus, each row consists of a *collection owner* (JID) plus one or more data fields.

2.5.1 Table List (for MySQL)

The following is a list of MySQL tables for a deployment that uses Jabberd for both its authentication and its application data package (*authreg* and *db*, respectively):

active
: Stores date/time upon which each account first became active.

authreg
: Contains authentication information, including username, realm and password.

disco-items
: Stores persistent discovery information so that it is available for offline retrieval.

logout
: Stores JID and timestamp for most recent user log out.

motd-message
: Stores message of the day (MOTD) in XML format.

motd-times
: Records JID's and timestamps for receipt of MOTD.

privacy-default
: Stores the name of the current list in use for a user so it can be made active on startup.

privacy-items
: Stores user privacy lists (blacklists/whitelists).

private
: Provides private XML storage for uses such as user preferences or bookmarks.

queue
: Stores queued messages in XML format.

roster-groups
: Stores user roster items only for those roster items that have an assigned group.

roster-items
: Stores user roster items, including authorization status.

vacation-settings
: Handles vacation settings, including start, end and message.

vcard

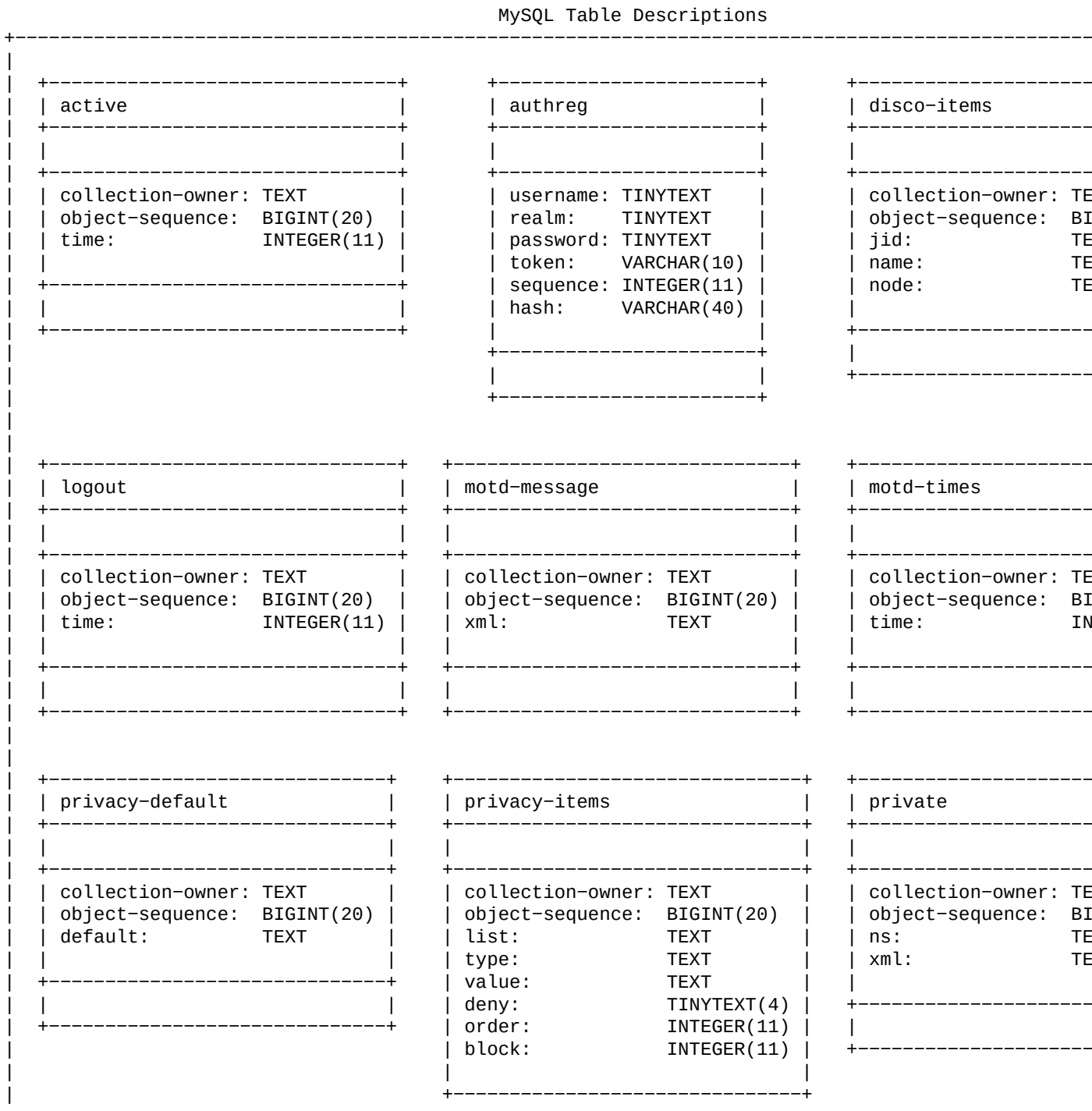
Stores vcard information.

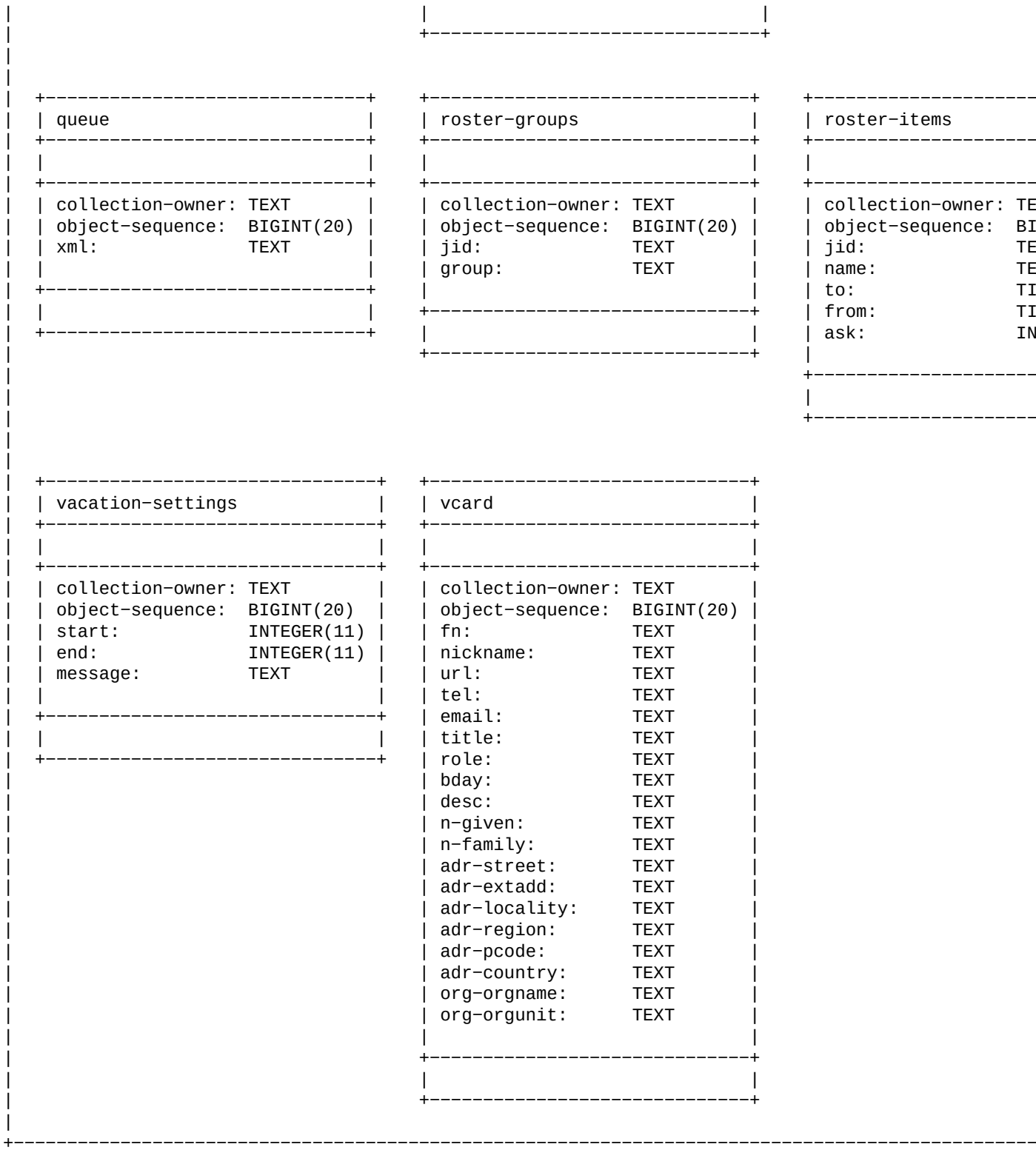
Each of the above tables represents a *collection type* in Jabberd. Note that the *authreg* table handles the authentication aspect for Jabberd when MySQL is used as the authentication data package.

2.5.2. Table Descriptions (for MySQL)

Descriptions for the Jabberd tables (as they exist in MySQL) appear in the diagram below:

Figure 2.5.2. Jabberd Table Descriptions:





View as PDF

Back | Up | Next

Comments

 Jabberd 2 Documentation Project

3. Preparation for Jabberd

This section will prepare you and your system to install Jabberd 2:

- Gather Required Information
- Create Jabber User and Group
- Create Directories for Logging and PID's
- Install Prerequisites
- Provision Application Data and Authentication Data Packages

The table in section 3.1. lists information required for Jabberd 2 installation. Collecting this information at this point is optional; however, completing this table now will make installation easier.

3.1. Gather Required Information

The table below lists information that will be required during the installation process. Information is provided for each parameter below:

- Parameter
- Required
- Section
- Description
- Your Information

"Parameter" is the name of the piece of information. Throughout this guide, specific parameters are referenced with the ***P*** convention. "Required is either "Y" (yes), "N" (no), or *Option Name*. *Option Name* refers to the option for which the parameter is necessary. Some entries have suggested entries for "Your Information." These entries represent either limited choices or default values. Where default values are given, they are used in the examples in this guide.

For a minimal installation, complete all the required parameters. For more information about the conditions under which an optional parameter is required, see the referenced section. "Description" is a short description. Again, see the referenced section for more detailed information.

I *Important: Table Contains Passwords*

The table below contains passwords. If you write your passwords in the table below, store this document (or page) in a secure location.

Table 3.1. Required Information for Jabberd 2 Installation

Parameter	Required	Section(s)	Description	Your Information
Jabberd User and Group	Y	3.2	The Linux (or other OS) user and group that will be used to run Jabberd	*user:* jabber *group:* jabber
PID Directory	Y	3.3.1	Directory in which Jabberd stores PID Files	/usr/local/var/jabberd/pid
Log Directory	N	3.3.2	Directory for Jabberd logs. If not specified in configuration files, logging defaults to syslog.	/usr/local/var/jabberd/log
Application Data Package	Y	3.4.2	Third party package to be used for storage of Jabberd application data	MySQL, PostgreSQL or Berkeley DB
Authentication Data Package	Y	3.4.3	Third party package to be used for storage of Jabberd authentication data	MySQL, PostgreSQL, Berkeley DB, OpenLDAP or PAM
Data Directory	Berkeley DB	3.5.1	Directory for Berkeley DB data files	/usr/local/var/jabberd/db
MySQL User and Password	MySQL	3.5.2.2, 6.4.1, 6.5.1	MySQL user and password that Jabberd uses to connect to MySQL	*user:* jabberd2 *password:* secret
PostgreSQL User and Password	PostgreSQL	3.5.3.1, 6.4.2, 6.5.2	PostgreSQL user and password that Jabberd uses to connect to PostgreSQL	*user:* jabberd2 *password:* secret
Hostname	Y	5.3	Hostname on which your Jabberd server is to be installed. For Internet accessible servers, this would be something like `somedomain.com`	
SSL Key Location	N	6.3.1, 6.3.2	Location of OpenSSL pemfile. Required for SSL−encrypted communication.	/usr/local/etc/jabberd/server.pem
Router User and Password	N	6.4	User and password used for component connections with the Router.	

3.2. Create Jabber User and Group

You should create a `jabber` user and group to run the server:

***P** Parameter: Jabber User and Group*

Create a user and group that will be used to run Jabberd (as superuser):

```
su
```

```
groupadd jabber
useradd -g jabber jabber
```

***I** Important: Check Your User and Group Commands*

The above commands are intended as an example. The commands and parameters for adding a user and group may vary for your system. Consult your manuals if you have any doubt about these commands.

3.3. Create Directories for PID's and Logs

You should create a directory for Jabberd to store its PID and log files, and ownership of these directories should be set to the user created above.

***P** Parameter: PID Directory*

Create a directory for PID files (as superuser):

```
su
mkdir -p /usr/local/var/jabberd/pid/
chown -R jabber:jabber /usr/local/var/jabberd/pid/
```

The above directory is the default location for Jabberd PID files.

You may also choose to create a directory for Jabberd logs.

***O** Optional: Log Directory*

If you wish, create a separate directory for Jabberd logs, and set ownership to your jabber user:

```
mkdir -p /usr/local/var/jabberd/log/
chown -R jabber:jabber /usr/local/var/jabberd/log
```

***N** Note: Log Files Default to Syslog*

Note that Jabberd writes messages to `syslog` by default. In order to force Jabberd to write its logs to the directory above, the component XML files must be edited to specifiy the directory created above.

3.4. Install Prerequisites

Jabberd 2 has three prerequisites:

- OpenSSL
- Jabberd 2 Application Data Package
- Jabberd 2 Authentication Data Package

OpenSSL must be installed in order to provide secure communication connections. Jabberd 2 also requires data packages for application and authentication; however, a single package, such as MySQL, can be used to satisfy both application and authentication requirements for data storage.

3.4.1. OpenSSL

OpenSSL provides encrypted client to server and server to server communication for Jabber. The XMPP Protocol requires that Jabber servers support TLS (Transport Security Layer). TLS is the successor to SSL.

***N** Note: Minimum Version*

Jabber 2 relies on OpenSSL versions 0.9.6b or higher.

***I** Important: OpenSSL Upgrade Issues*

If you upgrade OpenSSL, you may need to recompile installed software that currently relies on an older version of OpenSSL. This warning is provided because many utilities rely on OpenSSL, and these may cease to function after OpenSSL is upgraded. Caution is recommended when upgrading OpenSSL, and detailed instructions for upgrading OpenSSL are beyond the scope of this manual.

See the OpenSSL site for more information. OpenSSL downloads can be found on the OpenSSL Source page. Instructions for installing OpenSSL for Jabberd 2 are included in the appendix to this guide. See Installing OpenSSL for Jabberd 2.

3.4.2. Application Data Package

Jabberd 2 has better database integration than was previously supported, and Jabberd 2 can use one of three free databases to provide application data storage:

- MySQL
- Berkeley DB (4.1.24 or higher)
- PostgreSQL

MySQL is the recommended and default data store. A file may be used for storing Jabberd 2 data; however, this is not recommended.

N *Note: MySQL and Unicode Support*

MySQL does *not* support Unicode character encoding. This will not be an issue for most installations; however, if your installation requires support for multiple alphabet encodings, you should consider using PostgreSQL instead of MySQL. See the Unicode web site for information about Unicode.

If you are using one of the above databases, you may configure it to work with Jabberd. Otherwise, you should choose one of these databases and install it prior to continuing with Jabberd installation. MySQL is the recommended database; however, Berkeley DB requires the least installation and administration effort. Thus, Berkeley DB may be ideal for an installation with a relatively small number of users.

I *Important: MySQL Requires Development Libraries and Headers*

Note that Jabberd requires more than a minimal MySQL installation. In addition to the basic MySQL installation, Jabberd requires that the development libraries and headers be installed. Either perform a *Max* installation as listed on the MySQL Downloads page, or install *Server*, *Client Programs*, *Libraries and header files*, and *Dynamic client libraries* separately. It may be necessary to uninstall your current MySQL installation in order to install the additional libraries.

Appendices to this document contain instructions for installing either MySQL or Berkeley DB for Jabberd 2. See Installing Berkeley DB for Jabberd 2 or Installing MySQL for Jabberd 2.

3.4.3. Authentication Data Package

Jabberd 2 can use one of five free third−party authentication data packages to manage authentication data:

- MySQL
- Berkeley DB (version 4.1.24 or higher)
- PostgreSQL
- OpenLDAP
- PAM

Note that the three supported application data packages can also be used to manage authentication information. Therefore, installing one of MySQL, Berkeley DB or PostgreSQL satisfies both Jabberd data package requirements. A file may be used to store authentication data; however, this is not recommended. The recommended and default package is MySQL.

If you have one of the above five authentication data packages installed, you may configure it for use with Jabberd 2. If not, you should install one of the above five packages.

Appendices to this document contain instructions for installing either MySQL or Berkeley DB for Jabberd 2. See Installing Berkeley DB for Jabberd 2 or Installing MySQL for Jabberd 2.

View as PDF		
Back	Up	Next

Comments

2004/01/07 16:48 PST
If you're installing from a clean debian install and from deb packages only, you have to install zlib lib or configure will complain about missing mysql


Jabberd 2 Documentation Project

4. Install Jabberd 2

This section describes how to build and install Jabberd 2.

4.1. Download Jabberd

***F** Required File*

Download the file `jabberd-2.0rcn.tar.gz` from the Jabberd 2 Releases page, where "n" is the latest version of Jabberd 2.

|| TODO: Update file name as Jabberd goes to full release ||

Download the file referenced above into a directory in `/home` for building the installation files. At the time of writing, Jabberd 2 Beta release 2 is the latest version and is used in the examples below.

4.2. Extract Jabberd Installation Files

Change to the directory where you downloaded the file above and then extract the Jabberd 2 files by running the command:

```
tar -zxvf jabberd-2.0rc2.tar.gz
```

4.3 Configure the Jabberd Build

Change to the directory created above:

```
cd jabberd-2.0rc2
```

O *Optional: View Configuration Options*

Prior to configuring Jabberd, you can view all configuration options by running the command:

```
./configure --help
```

This will provide a listing and syntax for setting configuration options. For example, you may choose to install Jabberd into a specific directory by using the `--prefix=PREFIX` option. By providing a `PREFIX` path, all Jabberd files will be installed under this directory. This may be useful if you are testing a new Jabberd installation. Another useful option is `--enable-debug`. This option allows Jabberd to provide detailed debugging information; however, it should be used carefully on production systems. If your OpenSSL package is in a non-standard location, you will need to use the `--with-openssl=PATH` option.

P *Parameters: Application Data Package and Authentication Data Package*

Unless you are using MySQL as the *Application* and *Authentication Data Package*, you will need to configure the build with the appropriate packages. An alternate application data package should be specified with the `--enable-storage=` option, and an alternate authentication package should be specified with the `--enable-authreg=` option. For example, the following command would be used to configure Jabberd to use Berkeley DB and to enable debugging:

```
./configure --enable-storage=db --enable-authreg=db --enable-debug
```

This command would be used to enable PostgreSQL as the *Application* and *Authorization Data Package* without debugging support:

```
./configure --enable-storage=pgsql --enable-authreg=pgsql
```

I *Important: Incorrect Parameters Are Ignored*

Note that PostgreSQL is abbreviated to `pgsql` in the example above. Jabberd ignores incorrect configuration parameters. Thus, an incorrectly entered configuration parameter might lead to a successful, however incorrect, Jabberd configuration. Run `./configure --help` if in doubt.

I *Important: Redhat 9 Configuration*

Building Jabberd 2 on Redhat 9 requires special configuration because Redhat 9 ships with its own version of Kerberos. For more details, see the FAQ about Redhat 9 (under the "System Specific" heading).

If you wish to use the default configuration, simply run the configuration command:

```
./configure
```

This will configure Jabberd to use MySQL and to install to `/usr/local`.

4.4. Build Jabberd

Build Jabberd by running the command:

```
make
```

4.5. Install Jabberd

Switch to the super−user:

```
su
```

Run make install:

```
make install
```

4.6. Default File Locations

Your Jabberd 2 installation is complete. Below is a listing of file locations for the default installation:

```
/usr/local/etc/jabberd     Jabberd Configuration Files
/usr/local/bin       Jabberd Binaries (jabberd, c2s, resolver, router, s2s, sm)
```

4.7. Set Ownership of Configuration Files

Jabberd configuration files contain passwords; therefore, you should set ownership and permissions on these files so that they are only accessible by your `jabber` user. Using the location of your configuration files and your `jabber` user, set ownership of these files:

```
chown jabber:jabber /usr/local/etc/jabberd/*
```

Then, set permissions on these files so that others can neither read from− or write to them:

```
chmod 660 /usr/local/etc/jabberd/*
```

Now, only your `jabber` user and super−user will be able to read and edit your configuration files.

O *Optional: Create Symlink for Configuration Files*

If you used the default file locations when installing Jabberd, you may wish to create a symlink (as superuser) in `/etc` for the configuration files. This will make it easier to find and edit them:

```
ln −s /usr/local/etc/jabberd/ /etc/jabberd
```

Jabberd 2 is now installed. Continue to the next section to being configuring your installation.

View as PDF		
Back	Up	Next

Comments

2003/12/29 09:38 PST

before you get to the jabber 2 configure step: Red Hat 9 decided to put the include file for kerberos in /usr/kerberos/include instead of /usr/include.

Solution:

−−−−−−−−

1. Make sure you have the krb5−devel and krb5−libs rpms installed.

2. Create Symlink for the following files as follows:

```
ln −s /usr/kerberos/include/com_err.h /usr/include/
ln −s /usr/kerberos/include/profile.h /usr/include/
ln −s /usr/kerberos/include/krb5.h /usr/include/
```

Will, 2003/12/29 14:48 PST
Regarding RH 9, I've added a note above that references the FAQ. It seems that there are a number of work arounds for the Kerberos configuration issue. −.Ed

2004/01/06 13:46 PST
An easier way to compile on RedHat? 9 is to do this:

env CFLAGS=−I/usr/kerberos/include ./configure ....

 Jabberd 2 Documentation Project

5. Basic Configuration

This section provides a quick road map for the most basic configuration and testing of your Jabberd 2 installation. Edit your configuration files as below to get up and running quickly.

Basic setup for Jabberd 2 consists of these steps:

1. Provision Data Packages for Jabberd
2. Provision Jabberd for Data Packages
3. Set host name (`sm.xml` and `c2s.xml`)

For default installations, configuration files can be found in `/usr/local/etc/jabberd/`, and they are accessible from `/etc/jabberd` if you created the symlink for this directory.

***N** Note: Redundant Information in XML Files*

Note that some of the Jabberd configuration information seems redundant across configuration files. The reason for this is that the Jabberd components (c2s, s2s, sm, resolver and router) are designed to

be stand alone units with a single configuration file for each component.

5.1. Provision Data Packages for Jabberd

System and/or package configuration is required for whichever data package(s) is to be used for your Jabberd 2 installation.

P *Parameters: Application Data and Authentication Data Packages*

Complete the relevant sub−sections below for your choice(s) of Jabberd application and authentication data package(s).

5.1.1. Provisioning Berkeley DB

Complete this section if you are using Berkeley DB for application and/or authorization storage. Provisioning your system to use Berkeley DB is quite simple. Berkeley DB only needs a directory in which to store data files, and this directory should be owned by the jabber user and group.

P *Parameter: Data Directory*

Create a directory and set permissions (using the user and group created above) for Berkeley DB. (as superuser):

```
mkdir −p /usr/local/var/jabberd/db
chown −R jabber:jabber /usr/local/var/jabberd
```

Berkeley DB is now ready to be used with Jabberd.

5.1.2. Provisioning MySQL

Complete this section if you are using MySQL for application and/or authorization storage. The Jabberd 2 source files contain a script that creates a MySQL database for Jabberd. In addition to running this script, a jabberd user must be created, and this user must be granted access to the database.

5.1.2.1. Run MySQL Script

The MySQL setup script is located in '[Jabberd Source Files]/tools'. Switch to the `tools` directory and start the MySQL console (the MySQL server should already be running). Then, run the `db−setup.mysql` script from the MySQL console:

```
mysql −u root −p
mysql>\. db−setup.mysql
```

5.1.2.2. Create Jabberd User for MySQL and Grant Permissions

Now that a database for Jabberd exists in the MySQL data directory, create a MySQL user that Jabberd can use to connect to the MySQL server.

P *Parameter: MySQL User and Password*

From the MySQL console, run the SQL statement below, replacing `secret` with the password you have chosen for your Jabberd MySQL user:

```
GRANT select,insert,delete,update ON jabberd2.*
to jabberd2@localhost IDENTIFIED by 'secret';
```

Note that the password `secret` is the default password used in the Jabberd configuration files for MySQL.

MySQL is now ready to be used with Jabberd.

5.1.3. Provisioning PostgreSQL

Complete this section if you are using PostgreSQL for application and/or authorization storage. The Jabberd 2 source files contain a script that creates a PostgreSQL database for Jabberd.

5.1.3.1. Create PostgreSQL Database

Create the database for Jabberd. (The PostgreSQL server should already be running):

```
createdb -U postgres jabberd2
```

The command above will create a database from which you will be able to run the script for setting up the Jabberd PostgreSQL database.

***N** Note: Unicode Support*

If you want to enable Unicode support for your PostgreSQL database, change the command above to the following:

```
createdb -U postgres -E UNICODE jabberd2
```

5.1.3.2. Create PostgreSQL User

Create a user for the PostgreSQL jabberd database.

***P** Parameter: PostgreSQL User and Password*

To create your Jabberd database user, enter the command below:

```
createuser -U postgres jabberd2
```

This command will initiate an interactive user creation script. When prompted, enter the password that Jabberd will use to connect to your PostgreSQL database:

```
Enter password for user "jabberd2":
Enter it again:
Shall the new user be allowed to create databases? (y/n) n
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
```

The `CREATE USER` statement indicates that the command was successful.

5.1.3.3. Run PostgreSQL Script

The PostgreSQL setup script is located in '[Jabberd Source Files]/tools'. Switch to the `tools` directory and start the PostgreSQL console as the jabberd2 user:

```
psql -U jabberd2 jabberd2
```

Then, run the `db-setup.pgsql` script from the PostgreSQL console:

```
jabberd2=>\i db-setup.pgsql
```

PostgreSQL is now ready to be used with Jabberd 2.

5.1.4. Provisioning OpenLDAP

|| TODO: Can anyone provide information about setting up OpenLDAP for Jabberd? ||

5.1.5. Provisioning PAM

|| TODO: Can anyone provide information about setting up PAM for Jabberd? ||

5.2. Provision Jabberd for Data Packages

Just as your data package(s) must be configured for use by Jabberd, Jabberd must be configured to use your chosen data package(s). This configuration is performed in the Jabberd XML configuration files.

5.2.1 Configure Jabberd Application Data Package

Configuration for the Jabberd application data package is found in `sm.xml`, and the required configuration section corresponds to the `db=` option used during Jabberd build configuration (MySQL is the default). Jabberd database configuration in `sm.xml` is straight forward — you must set the database driver to use in addition to setting the user name and password for your application data package.

***N** Note: Berkeley DB Does Not Require Password*
: Note that if you are using Berkeley DB as your application data package, you should skip this section. Berkeley DB requires neither a user nor password for connections with the Jabberd server.

***N** Note: `secret` Is Default Password*
: Note that in the Jabberd XML files, the default password `secret` is used for all internal and external server connections. It is *not* recommended that this default password be used for connections to database servers because Jabberd has write permissions to these databases.

***P** Parameter: Application Data Package*
: In `sm.xml` under the section labeled `Storage database configuration`, edit the `driver` to match your installation. The default is `mysql`. For example, you would edit the `driver` as below if you are using PostgreSQL for your application data package:

```
<driver>pgsql</driver>
```

Note that "db" is used as the abbreviation for Berkeley DB.

After setting the database module, Complete the sub−section below as applicable to your installation.

5.2.1.1. Set MySQL User and Password (`sm.xml`)

***P** Parameter: MySQL User and Password*
: In `sm.xml`, edit the section shown below so that `secret` is replaced with your MySQL password. Change the user if you are not using the default user (`jabberd2`):

```
<!−− MySQL driver configuration −−>
<mysql>
  <!−− Database server host and port −−>
  <host>localhost</host>
  <port>3306</port>

  <!−− Database name −−>
```

```
  <dbname>jabberd2</dbname>

  <!-- Database username and password -->
  <user>jabberd2</user>
  <pass>secret</pass>
```

5.2.1.2. Set PostgreSQL User and Password (`sm.xml`)

P *Parameter: PostgreSQL User and Password*

In `sm.xml`, edit the section shown below so that `secret` is replaced with your PostgreSQL password. Change the user if you are not using the default user (`jabberd2`):

```
<!-- PostgreSQL driver configuration -->
<pgsql>
  <!-- Database server host and port -->
  <host>localhost</host>
  <port>5432</port>

  <!-- Database name -->
  <dbname>jabberd2</dbname>

  <!-- Database username and password -->
  <user>jabberd2</user>
  <pass>secret</pass>
```

5.2.2. Configure Jabberd Authentication Data Package

Configuration for the Jabberd authentication data package is found in `c2s.xml`, and the required configuration section corresponds to the `authreg=` option used during Jabberd build configuration (MySQL is the default). Jabberd authentication data package configuration is similar to the configuration in the previous section — you need to set the driver to use in addition to setting the user name and password for your authentication data package.

P *Parameter: Authentication Data Package*

In `c2s.xml` in the section labeled `Authentication/registration database configuration`, edit the `module` to match your installation. The default is `mysql`. For example, you would edit the "Backend module to use' as below if you are using Berkeley DB for your application data package:

```
<module>db</module>
```

After setting the database module, Complete the sub−section below as applicable to your installation.

5.2.2.1. Set MySQL User and Password (`c2s.xml`)

P *Parameter: MySQL User and Password*

In `c2s.xml`, edit the section shown below so that `secret` is replaced with your MySQL password. Change the user if you are not using the default user (`jabberd2`):

```
<!-- MySQL module configuration -->
<mysql>
  <!-- Database server host and port -->
  <host>localhost</host>
  <port>3306</port>

  <!-- Database name -->
```

```
  <dbname>jabberd2</dbname>

  <!-- Database username and password -->
  <user>jabberd2</user>
  <pass>secret</pass>
</mysql>
```

5.2.2.2. Set PostgreSQL User and Password (`c2s.xml`)

P *Parameter: PostgreSQL User and Password*

In `c2s.xml`, edit the section shown below so that `secret` is replaced with your PostgreSQL password. Change the user if you are not using the default user (`jabberd2`):

```
<!-- PostgreSQL module configuration -->
<pgsql>
  <!-- Database server host and port -->
  <host>localhost</host>
  <port>5432</port>

  <!-- Database name -->
  <dbname>jabberd2</dbname>

  <!-- Database username and password -->
  <user>jabberd2</user>
  <pass>secret</pass>
</pgsql>
```

|| TODO: XML setup for LDAP and PAM ||

Your Jabberd 2 server is now ready for testing.

C *Checkpoint: Start Your Server*

You should be able to start and test your Jabberd 2 server by using the Jabberd 2 startup script (as your jabber user):

```
su
su jabber
/usr/local/bin/jabberd
```

N *Note: Public Registration*

Public registration for new users is enabled in Jabberd2 by default. Thus, when testing your server, you can create a new user by logging on as a new user.

N *Note: Troubleshooting*

If Jabberd does not start, make sure that any previous instances have stopped. These instances include all the Jabberd runtime components (`jabberd`, `router`, `resolver`, `sm`, `s2s` and `c2s`). Also, check that your chosen data package servers are running (except Berkeley DB, which does not require starting). Check your `syslog` for error messages. If your server fails to start, you can start Jabberd 2 with the debug option (note that this requires building Jabberd 2 with the debug option — see section 4.3):

```
/usr/local/bin/jabberd -D
```

I *Important: Production Servers Should Not Be Run as Superuser*

Production servers should not be run as superuser.

C *Checkpoint: Connect from Client on Same Machine*

If your server starts, you should be able to use a Jabber client — running on the same machine as the server — to connect to your Jabberd 2 server. The Jabberd 2 server ID defaults to `localhost`, so

when creating an account, enter a Jabber ID such as `some_name@localhost`. Once you have created an account, you should be able to log on to the server. Note this may not be possible on servers that are only accessible remotely.

C *Checkpoint: Connect from a Machine on the Same Network*

If you are successful at connecting from the same machine on which Jabberd 2 is installed, you can try connecting from a different machine on the same network. The Jabber server should still be specified as `localhost` so that the Jabber ID should be something like `another_name@localhost`. In the client configuration, you will need to specify a host for this server (otherwise, the client would resolve localhost to itself). In the settings for your client, specify the host as the IP address on which your Jabberd 2 server is running, such as `host: 192.168.0.2, port: 5222`. Note that not all Jabber clients support a host address feature. Note this may not be possible on servers that are only accessible remotely.

5.3. Set Host Name in `sm.xml` and `c2s.xml`

The final step in basic configuration consists of setting the hostname in `sm.xml` and `c2s.xml`.

P *Parameter: Hostname*

Your server hostname (network ID) must be set in both `c2s.xml` and `sm.xml` so that the ID provides a network resolvable reference for your server. In `c2s.xml` this ID is found under the heading labeled `Local network configuration` (approx. line 59), and in `sm.xml` this ID is found under `Session manager configuration` (line 1). Edit `c2s.xml` and `sm.xml` so that this ID references your server:

```
<id>some_domain.com</id>
```

As the `c2s.xml` file notes, this is the hostname that will be appended to your user names to create Jabber ID's, and it must be resolvable via DNS for Jabberd to be accessible via the Internet.

C *Checkpoint: Connect from Client on Remote Network*

Restart your Jabberd server and use a Jabber client to connect to it from a remote network. This will test that `id` is set properly and that the machine name is resolvable via DNS.

Your Jabberd 2 server is now ready to use. Continue to the next section for detailed configuration options.

View as PDF

Back | Up | Next

Comments

2003/11/12 12:43 PST
Using Redhat 9 and the BerkeleyDB? RPMS (db4−4.1.25−13, db4−devel−4.1.25−13) you have to pass DB_PRIVATE to open() in authreg_db.c "Berkeley DB library configured to support only DB_PRIVATE environments"

2003/11/19 02:23 PST
Looks like the user jabberd2, created following section 5.1.3, has no rights to database jabberd2. One must grant permissions to this database to him.

 Jabberd 2 Documentation Project

6. Common Configuration Tasks

Before delving into detailed Jabberd configuration, this section attempts to provide a guide for the most common Jabberd configuration tasks:

- Creating an Administrative User
- Disabling Public Registration
- Configuring SSL
- Changing Router Password
- Using Jabberd 1.4 to Connect to Legacy Services
- Using JCR to Connect to Legacy Services
- Setting DNS SRV Records

Note that there are two options for connecting Jabberd 1.4 legacy services to your Jabberd 2 installation. Jabberd 2 can connect to services, such as conferencing and gateways, running within a Jabberd 1.4 process. Additionally, a component wrapper called JCR has been released, and this wrapper allows a Jabber 1.4 component (written in C) to be compiled and run as Jabberd 2 service. At the time of writing, JCR has been tested with MU Conferencing only.

6.1. Creating an Administrative User

Settings for administrative users are contained in the `aci` section of `sm.xml`. By default, the administrative user is `admin@localhost`. In order to enable an administrative user that can be accessed remotely, change the `jid` to a user of your own choosing as below:

```
<acl type='all'>
  <jid>admin@lsomedomain.com</jid>
</acl>
```

You will also need to create this user manually or from a Jabber client. When logged in, the administrative user will receive notices for user creation, and the administrative user will also be able to discover all online users, receive help requests, send MOTD's (messages of the day), etc.

Note that the user above is granted access to *all* administrative functions. You can assign specific administrative functions to users by specifying the `acl type`. See the examples in the `sm.xml` file.

Restart your Jabberd server for the change to take effect.

6.2. Disabling Public Registration

By default, Jabberd allows public registration for all users, which is to say that any user who can connect to your server can create their own Jabberd user on your server. In order to prevent public registration, edit the `c2s.xml` configuration file.

Under the `Authentication/registration database configuration` section, look for the `Registration configuration` subsection. Commenting the `enable` tag as below will disable public registration:

```
<!-- <enable/> -->
```

With public registration disabled, it may be useful to enable users to change their own passwords. This is disabled by default; however, it can be enabled in the same section as above:

```
<!-- Password change only. When registration is disabled, it may
     still be useful to allow clients to change their password. If
     you want this, uncomment this when you disable registration. -->
<!--
<password/>
-->
```

Uncommenting the `password` tag above will allow your users to change their own passwords.

Restart your Jabberd server for the change to take effect.

6.3. Configuring Jabberd for SSL Connections

Jabberd 2 is designed to provide for SSL connections not only between Jabber clients and the server, but also between the Jabberd server components (`sm`, `resolver`, `s2s` and `c2s`) and the Jabberd `router`. A single SSL certificate may be used for these two functions (Jabber client to Jabberd and Jabberd component to router), or two separate keys may be used. See the appendix, Generating a Self−Signed SSL Certificate for instructions about how to create your own self−signed certificate for use by Jabberd.

***N** Note: Self−Signed Certificates Not Trusted*

Note that self−signed certificates are not automatically trusted by Jabber clients because there is no chain of authority against which to verify authenticity. Nevertheless, creating a self−signed certificate not only will allow your Jabber users to communicate over a secure channel (possibly with warnings displayed by the client), but also such a certificate will provide for secure communication among the five Jabberd components (`router`, `sm`, `resolver`, `s2s` and `c2s`).

6.3.1. Assigning a Certificate for Use by Jabber Clients

The location of the SSL key for Jabber Clients is located in `c2s.xml`. Note that `c2s.xml` contains a location of the SSL key for Jabber Clients in addition to the location of the SSL key for `c2s` to `router` communications.

***P** Parameter: SSL Key Location*

Uncomment the `pemfile` (your SSL key) location under the section labeled `Local network configuration`, and edit it for the location of your SSL key. Note that if you use the default location of `/usr/local/etc/jabberd/server.pem`, you need only uncomment this section as below:

```
<!-- File containing a SSL certificate and private key for client
     connections. If this is commented out, clients will not be
     offered the STARTTLS stream extension -->

<pemfile>/usr/local/etc/jabberd/server.pem</pemfile>
```

You need only restart the C2S component for the SSL change to take effect.

N *Note: Disabling Non−SSL Communication*

To require SSL communication with clients, disable the non−SSL port in `c2s.xml`. In the `local` section, set the unencrypted port to '0':

```
<!-- Port to bind to, or 0 to disable unencrypted access to the
     server (default: 5222) -->
<port>0</port>
```

6.3.2. Assigning a Certificate for Use by Jabberd Components

Each of the five Jabberd components has its own configuration for encrypted component−to−router communications. Thus, these five configuration files must be edited to provide secure communication among Jabberd components:

```
router.xml
sm.xml
resolver.xml
s2s.xml
c2s.xml
```

P *Parameter: SSL Key Location*

In each of the files above, uncomment the `pemfile` tag for router communications. In the `router.xml` file, the `pemfile` is specified under the section labeled `Local network configuration`. In each of the the remaining four configuration files above, the `pemfile` location is specified under the section labeled `Router connection configuration`. Uncomment this section and edit it to point to the location of your SSL key. For example, you would edit `c2s.xml` as below if you are using the default location for your SSL key:

```
<!-- Router connection configuration -->
<router>
  <!-- IP/port the router is waiting for connections on -->
  <ip>127.0.0.1</ip>      <!-- default: 127.0.0.1 -->
  <port>5347</port>       <!-- default: 5347 -->

  <!-- Username/password to authenticate as -->
  <user>jabberd</user>    <!-- default: jabberd -->
  <pass>secret</pass>     <!-- default: secret -->

  <!-- File containing a SSL certificate and private key to use when
       setting up an encrypted channel with the router. If this is
       commented out, or the file can't be read, no attempt will be
       made to establish an encrypted channel with the router. -->

  <pemfile>/usr/local/etc/jabberd/server.pem</pemfile>

Restart your Jabberd server for the change to take effect.
```

6.4. Changing Router Password

The Jabberd configuration files contain passwords used to connect to the *router* component because communications between components and the router occur by XML over TCP/IP. These passwords help to ensure that only your installed components can communicate with the *router*. The configuration file, `router-users.xml`, contains ID's and password(s) for components that are allowed to connect to the router. By default, the ID is `jabberd` and the password is `secret`.

Additionally, each of the components (except the *router*) has a user ID and password specified in its configuration file. This is the ID and password combination that the respective component uses to connect with the router. Thus, for a component to be able to connect to the router, the component must have a user and password pair specified in its configuration file, and that pair must match an ID and password pair in `router-users.xml`.

To improve security for your Jabberd installation, you should change the password. This involves changing the password in `router-users.xml` and then in `sm.xml`, `resolver.xml`, `s2s.xml` and `c2s.xml`.

P *Parameter: Router User and Password*

In order to change the password used for authentication by the router, first change the password in `router-users.xml` as below (replacing `newpass` with your new password):

```
<users>
  <user>
    <name>jabberd</name>
    <secret>newpass</secret>
  </user>
</users>
```

Then, change the password in each of `sm.xml`, `resolver.xml`, `s2s.xml` and `c2s.xml`. In each of these files there are tags for the *router user* in the `router` section. Change the password as below (replacing `newpass` with your new password):

```
<!-- Router connection configuration -->
<router>
  <!-- IP/port the router is waiting for connections on -->
  <ip>127.0.0.1</ip>      <!-- default: 127.0.0.1 -->
  <port>5347</port>       <!-- default: 5347 -->

  <!-- Username/password to authenticate as -->
  <user>jabberd</user>    <!-- default: jabberd -->
  <pass>newpass</pass>     <!-- default: secret -->
```

Restart your Jabberd server for the change to take effect.

N *Note: Multiple Passwords Allowed*

Note that you can assign a user and ID for each of the components if you wish.

6.5. Using Jabberd 1.4 to Connect to Legacy Services

This section describes how to use an existing Jabberd 1.4 installation to connect legacy Jabberd 1.4 services, such as gateways and transports, to Jabberd 2. See the next section for using the JCR component to connect to legacy services. A familiarity with Jabberd 1.4. is assumed, as is a working Jabberd 1.4 installation. Jabberd 1.4 setup and configuration is beyond the scope of this document; however, many excellent resources exist,

including the Jadmin Archive.

Connecting Jabberd to a legacy service is similar to the means by which Jabberd 1.4 links to services; however there are several important differences when linking from Jabberd 2 to a Jabberd 1.4 service:

- The linked configuration file must have an XDB section
- The linked configuration file must have a log section
- The linked configuration file must specify the *router* IP address and port
- The linked configuration file and the *router* configuration file must share a password
- An alias for the service must be specified for the *router*

Additionally, a service namespace should exist in `sm.xml` if the component does not support discovery (`disco`). For example, MU Conference supports discovery, so no entry is required in `sm.xml` in order to make this service browsable. On the other hand, JUD (Jabber User Directory) does not support discovery; therefore, an entry would be required in `sm.xml` to make a JUD browsable by users. See `sm.xml` for examples.

The sub−sections below describe how to configure a David Sutton's MU Conference for Jabberd 1.4, and the example concludes with a working MU Conference configuration file. See MU Conference for downloads and more information about this component. MU Conference provides a multi−user chat room service for Jabberd.

6.5.1. XDB Section

When running a gateway or transport with Jabberd 1.4, the main Jabberd process handles XDB and logging functions. This is not the case when running such a component with Jabberd 2. Instead, the component should run in its own process, and the component should handle its own XDB and logging functions. That is to say that legacy components should be run in a process that is self−contained. Thus, if you have a component that is linked to a Jabberd 1.4. server, you should add an *XDB* section to the service configuration file:

```
<xdb id="xdb">
  <host>conference.somedomain.com</host>

  <load>
    <xdb_file>/usr/local/jabber/xdb_file/xdb_file.so</xdb_file>
  </load>

  <xdb_file xmlns="jabber:config:xdb_file">
    <spool>/usr/local/var/spool/jabber</spool>
  </xdb_file>
</xdb>
```

The *XDB* section above specifies the hostname, the XDB module to load, and the location to write the spool file(s).

6.5.2. Log Section

As noted above, the component configuration file must also contain a logging section. You might add a section like this to the configuration file for your linked component:

```
<log id="muclog">
  <file>/usr/local/var/jabberd/log/muc.log</file>
  <host/>
  <logtype/>
  <format>%d: [%t] (%h): %s</format>
```

```
</log>
```

In the case of an MU Conference gateway, I have specified the log above as `muc.log`. Each service should have a separate log file.

6.5.3. Router IP Address and Port

The linked component should specify the IP address and port that the *router* is listening on. The default port for the *router* is `5347`, so the beginning of the `id` section of your linked file would look something like this:

```
<service id="muclinker">
  <uplink/>
  <connect>
    <ip>192.168.0.2</ip> <!-- IP Address of Router here -->
    <port>5347</port>
```

6.5.4. Shared Password

The Jabberd 2 server uses a password for component connections. This password is similar to the *shared secret* that is used in Jabberd 1.4. Continuing with the example above, you should create a password for your legacy component to use. (In Jabberd 2, all legacy components share the same password with the *router*) This password, or `secret`, must be specified in your the configuration file for your linked component:

```
<service id="muclinker">
  <uplink/>
  <connect>
    <ip>192.168.0.2</ip> <!-- IP Address of Router here -->
    <port>5347</port>
    <secret>ComponentPass</secret>
  </connect>
</service>
```

This `secret` must also be specified in your `router.xml` file. The `secret` is set in the section labeled 'local network configuration':

```
<!-- Local network configuration -->
<local>
  <!-- IP address to bind to (default: 0.0.0.0) -->
  <ip>0.0.0.0</ip>
  <!-- Port to bind to (default: 5347) -->
  <port>5347</port>
  <!-- File containing the user table. This is where the router gets
       its component and secret information from for component
       authentication.-->
  <users>/usr/local/etc/jabberd/router-users.xml</users>
  <!-- Shared secret used to identify legacy components (that is,
       "jabber:component:accept" components that authenticate using
       the "handshake" method). If this is commented out, support for
       legacy components will be disabled. -->
  <secret>ComponentPass</secret>
```

Note that the password is specified with the `secret` tag in `router.xml` (as above), and not in the `router-users.xml` file.

6.5.5. Router Name Alias

The `router.xml` file must also contain an alias for the linked component, and the `router.xml` file provides an example for uplinking an MSN transport. Add an alias section for the component to which you are linking. Continuing with the example above, this alias would be set as below in 'router.xml':

```
<!-- Name aliases.

     Packets destined for the domain specified in the "name" attribute
     will be routed to the component that has currently bound the name
     in the "target" attribute (assuming it is online).

     This is usually only required for some kinds of legacy
     components (particularly jabberd 1.4 "uplink" components) -->
<aliases>
  <!-- Example for a msn transport running from a jabberd 1.4 uplink -->
  <!--
  <alias name='msn.domain.com' target='msn-linker'/>
  -->
  <alias name='conference.somedomain.com' target='muclinker'/>
</aliases>
```

Note that the alias references the DNS-resolvable name for the component, and the `target` references the `service id` as specified in the linked component XML file (as shown in section 6.5.4).

6.5.6. Example: *muc.xml*

A working `muc.xml` file for MU Conferencing appears below:

```
<jabber>

  <service id="muclinker">
    <uplink/>
    <connect>
      <ip>192.168.0.2</ip> <!-- IP Address of Router here -->
      <port>5347</port>
      <secret>ComponentPass</secret>
    </connect>
  </service>

  <service id="conference.somedomain.com">

    <load>
       <conference>/usr/local/jabber/mu-conference/src/mu-conference.so</conference>
    </load>

    <conference xmlns="jabber:config:conference">
      <public/>

      <vCard>
  <FN>Public Chatrooms</FN>
  <DESC>This service is for public chatrooms.</DESC>
      </vCard>

      <history>50</history>

      <logdir>/usr/local/var/jabberd/log/muc/</logdir>

      <sadmin>
  <user>admin@somedomain.com</user>
```

```
    </sadmin>

    <notice>
  <join>has become available</join>
  <leave>has left</leave>
  <rename>is now known as</rename>
    </notice>

  </conference>

</service>

<log id="muclog">
  <file>/usr/local/var/jabberd/log/muc.log</file>
  <host/>
  <logtype/>
  <format>%d: [%t] (%h): %s</format>
</log>

<xdb id="xdb">
  <host>conference.somedomain.com</host>

  <load>
    <xdb_file>/usr/local/jabber/xdb_file/xdb_file.so</xdb_file>
  </load>

  <xdb_file xmlns="jabber:config:xdb_file">
    <spool>/usr/local/var/spool/jabber</spool>
  </xdb_file>
</xdb>

</jabber>
```

C *Checkpoint: Test Your Legacy Component*

Once your legacy component is set up, you should start up the component by running it in a Jabberd 1.4 process using a command similar to this:

```
/usr/local/jabber/jabberd/jabberd -c /etc/jabber/muc.xml
```

The `-c` parameter specifies a configuration file for Jabberd 1.4. After you restart your Jabberd 2 server, your new (legacy) service should be running. In the case of MU Conference, you should be able to set up multi-user chat rooms via a compatible client, or via the script included with the tarball.

6.6. Using JCR to Jabberd 2 Components

Given the newness of the Jabberd 2 server, there are few, if any, components designed for use specifically with the Jabberd 2 server. Nevertheless, Paul Curtis has written a component wrapper called the Jabber Runtime Component that is designed to run legacy components as Jabberd 2 services. The Jabber Runtime Component (JCR) allows C language components for Jabberd 1.4 to be run as standalone processes that connect to Jabberd 2. At the time of writing, only MU Conference has been tested with JCR.

6.7. Setting DNS SRV Records

Jabberd 2, in addition to other Jabber clients and servers, is able to use DNS SRV records for hostname resolution. DNS SRV records allow for delegation of services — by port — to other hosts. Thus, if you want your Jabber server to run on a host that is not the primary domain host, you would most likely want to set DNS SRV records to delegate Jabber client and server services to another host or hosts.

***N** Note: SRV Records Required Only for Non−Primary Host*

Note that DNS SRV records are required only if your Jabberd server is running on a host other than the primary domain host and if you do not wish to include the host (machine) name in your Jabber ID. For example, if a DNS query for `somedomain.com` resolves `host1.somedomain.com`, and your Jabberd server is running on `host1`, SRV records would not be required.

6.7.1. SRV Records for Jabberd

There are 3 SRV records that can be created for a Jabberd installation:

```
_jabber._tcp.<domain>   −>    <host>.<domain>:5269
_xmpp−server._tcp.<domain>    −>    <host>.<domain>:5269
_xmpp−client._tcp.<domain>    −>    <host>.<domain>:5222

|| TODO: Add SSL record ||
```

The first and second of these specify the host and the port for server−to−server (s2s) communications. There are two listings for this because the new XMPP protocol, regarding SRV records, is replacing the older Jabber standards. The third listing above specifies host and port for unencrypted client communications (c2s).

6.7.2. Creating SRV Records in Bind

The following are examples for creating a set of SRV records for the BIND server:

```
_jabber._tcp.some_domain.com. 86400 IN SRV 5 0 5269 host.some_domain.com.
_xmpp−server._tcp.some_domain.com. 86400 IN SRV 5 0 5269 host.some_domain.com.
_xmpp−client._tcp.some_domain.com. 86400 IN SRV 5 0 5222 host.some_domain.com.
```

Replace `some_domain.com` with your domain name and `host` with the name of the host, and do not omit the "." after the domain name.

6.7.3 Creating SRV Records in TinyDNS

TinyDNS does not have a format for SRV records; however, you can use Rob Mayoff's TinyDNS Record Maker to create TinyDNS SRV records. These TinyDNS SRV records were created for the host of `host.some_domain.com` using a priority of `10` and a weight of '0':

```
:_jabber._tcp.some_domain.com:33:\000\012\000\000\024\225\004host\013some_domain\003c
:_xmpp−client._tcp.some_domain.com:33:\000\012\000\000\024\146\004host\013some_domain
:_xmpp−server._tcp.some_domain.com:33:\000\012\000\000\024\225\004host\013some_domain
```

Use TinyDNS Record Maker to create a set of records to be added to the TinyDNS data file.

6.7.4. Testing SRV Records

Once the your DNS server is properly updated, you should test the listings using Dig. For example, to test the entry of `_jabber._tcp.some_domain.com`, using the DNS server `my.dns_server.com`, you would enter the command below:

```
dig @my.dns_server.com _jabber._tcp.some_domain.com any +short
```

This should provide you with the data from your DNS SRV record:

```
10 0 5269 host.some_domain.com.
```

Comments:

 Jabberd 2 Documentation Project

7. Common Administrative Tasks

This section attempts to provide a guide for the most common day−to−day Jabberd administrative tasks, including user management.

- Converting from Jabberd 1.4
- Adding Users
- Removing Users
- Sending MOTD's and Messages to All Online Users

Detailed configuration begins in the next section.

7.1. Converting from Jabberd 1.4

Robert Norris, the primary Jabberd developer, has created a Perl script that migrates Jabberd 1.4 data to either MySQL or PostgreSQL. Specifically, the script migrates only authentication information and rosters. The migrate.pl script is currently available from CVS on jabberstudio.org; however, it will be released with the next Jabberd 2 release.

Essentially, the script works as a service that connects to an existing Jabberd 1.4 installation, and it exports to an existing Jabberd 2 database. It has several dependencies (as listed in the script):

- XML::Stream 1.17 or higher (from JabberStudio)
- Net::Jabber 1.29 or higher (from JabberStudio)
- Digest::SHA1
- DBI
- DBD::Pg or DBD::mysql

The migrate.pl file contains complete instructions for use.

7.2. Adding Users

At the time of writing, there are no scripts for adding users; however, users can be added easily to the database. This section describes how to add a user to a MySQL database. Adding a user to other authentication databases, such as PostgreSQL or LDAP, would be similar. This section assumes that public

registration has been disabled (see Section 6.2).

7.2.1. Enable Auto−Create

The `user` section of `sm.xml` contains a tag named `auto−create`, and enabling this tag causes a user to be created in the *application data package* the first time a user logs on if that user does not yet exist in the database. With this tag enabled, users can be created in the *authentication data package* (`authreg`), and when that user logs in for the first time, Jabberd will create rosters, templates, etc. for that user. Specifically, the `auto−create` tag causes the `user−create` chain to be called when a new user logs in for the first time.

Auto−create is disabled by default. To enable it, uncomment the `auto−create` tag in the `user` section of the `sm.xml` file. This tag is near the end of the `sm.xml` file.

7.2.2. Create User from MySQL Console

Now that `auto−create` is enabled, you can create a user from the MySQL console.

Log onto the MySQL console as the Jabberd user:

```
mysql −u jabberd2 −p
```

From the MySQL console, Switch to the `jabberd2` database:

```
mysql> use jabberd2
```

From the MySQL console, Insert a row into the `authreg` table. The row should contain values for `username`, `realm` and 'password':

```
mysql> insert into authreg (username, realm, password)
    −> values ('myusername', 'somedomain.com', 'mypassword');
```

Change the value of `somedomain.com` to match your configuration. In most cases, the realm will be the same as the ID of the Jabber server. Change `myusername` and `mypassword` to the name and password for your new user.

Your new user will now be able to log on using a Jabber client.

7.3. Removing Users

The MySQL console can be used to delete users from a MySQL authentication database (authreg). Enter this command to delete a user:

```
mysql> delete from authreg where username='the_user_name';
```

Note that deleting the user from the `authreg` table will not delete additional user data, such as rosters; however, once the user has been deleted from the `authreg` table, the user will no longer be able to log on to the Jabberd server.

7.4. Sending MOTD's and Messages to All Online Users

Sending MOTD's (Messages of the Day) and messages to all online users must be done from an administrative account with `broadcast` privileges. See Section 6.1 for information about how to assign these privileges. Once an account has `broadcast` privileges, these messages can be sent from a Jabber client. Add the following contacts to the administrative account you have created:

```
somedomain.com/announce

somedomain.com/announce/online
```

Sending a message to `somedomain.com/announce` will send a Jabber MOTD to all users on the domain. Offline users will receive the message the next time they log on. Sending a message to `somedomain.com/announce/online` sends a message only to online users.

View as PDF		
Back	Up	Next

Comments


Jabberd 2 Documentation Project

8. Router.xml Configuration

The `router.xml` file provides configuration for the *router* component, which is the backbone of the Jabberd server. The router component accepts connections from other components, and it passes XML packets between them. The `router.xml` file contains these primary sections:

- ID, PID and Logging
- Network
- Input/Output Control
- Aliases
- Feature access controls

As readers are probably aware, the Jabberd XML files are self−documenting. This and subsequent configuration sections attempt merely to provide an overview of the structure of these files in addition to tips not found in the files themselves.

8.1. ID, PID and Logging

The first three subsections of `router.xml` handle basic housekeeping items. For nearly all installations, the `id` section should be kept as default (`router`). The `pid` section specifies where the *router* component

should write its process ID. If you do not need to be able to read the process ID from outside of Jabberd, you may comment this section out. For example, if you were to use D.J. Bernstein's DaemonTools to run Jabberd, you would comment the `pid` section.

Jabberd logging defaults to the `syslog`. If you prefer the `router` to write to its own log file, change the `log type` to `file`, and specify a location for the log.

8.2. Network

The `local` subsection handles network handles network and connection settings for the *router*. The `ip` and `port` tags specify the network settings.

***N** Note: `0.0.0.0` Specifies All Available IP's*

> Note that in all of the Jabberd XML files, an IP address specified as `0.0.0.0` specifies that the component should listen on all available IP addresses. `0.0.0.0` is the default setting for IP addresses, and keeping this default setting should be fine for most installations.

The `users` tag specifies the file containing user ID's and passwords for users that can access the *router*. See the next section for a description of this file.

The `secret` tags specify a shared secret for legacy services. Legacy services are authenticated via this shared secret, as opposed to being authenticated against the user list in the `router-users.xml` file. This *secret* is tantamount to the shared secret specified for Jabberd 1.4 linked configuration files. See Section 6.5 for information on setting up legacy services.

The `pemfile` location specifies the location of the certificate and private key for client communications. Note that in this instance, *client* refers to clients of the *router*, and not end-user Jabber clients. That is to say, the *pemfile* specified here is used to secure communications between other Jabberd components and the *router* itself. See Section 6.3 and Appendix: Generating A Self-Signed SSL Key for more information about setting up SSL on Jabberd. Commenting this section has the effect of disabling SSL communication between Jabberd components.

8.3. Input/Output Control

The `i/o` section controls the following input/output options:

- Connection Limiting
- Rate Limiting
- IP Access Control

Note that the default settings for these subsections should be fine for most installations.

8.3.1. Connection Limiting

Jabberd sets a limit on the number of connections via the `max_fds` (maximum file descriptors) setting. Jabberd uses a file descriptor for each connection. Thus, setting a maximum number of file descriptors for the router has the effect of limiting the number of concurrent connections for the Jabberd server. As the `router.xml` file notes, the *router* itself can use up to 4 connections (with other Jabberd components); therefore, the maximum number of file descriptors set here is actually slightly greater than the number of potential concurrent connections.

8.3.2. Rate Limiting

The `limits` subsection dictates throttling for individual connections. This section is tantamount to a simplified *karma* setup as found in Jabberd 1.4. The default for both `bytes` and `connects` is `0`. Thus, these limits are disabled by default. Administrators of servers under heavy load may wish to set limits here to prevent users from controlling excessive server resources. The `router.xml` file contains examples for setting rate limiting on connections.

8.3.3. IP Access Control

The `access` subsection specifies IP addresses that should be allowed or denied access to the *router*. IP addresses denied access to the *router* cannot have their packets handled and are thus denied access to Jabberd server functions.

|| QUESTION: Is the above statement actually correct? ||

The `order` subsection specifies the order of rule checking (checking of allow rules then deny rules versus checking of deny rules then allow rules). IP restrictions are set using either an `allow` or a `deny` tag below the `order` within the `access` subsection. Omission of both a deny and allow rule causes all connections to be accepted — as is the default setting.

8.4. Aliases

The `aliases` section provides support for linked legacy services, such as MU Conference and JUD. These legacy services connect directly to the *router* via linked configuration. Each `alias` specifies the resolvable `name` for the component and the `target` specifies the `service id` as specified in the linked component XML file. See Section 6.5 for more information about linking legacy services.

8.5. Feature Access Controls

The `aci` section specifies the type of access for each user specified in `router-users.xml`. By default, the `jabberd` user is granted all access because the other Jabberd components connect to the *router* as the `Jabberd` user. Of special note in this section is the example of a user with the 'acl type = 'log''. Such a user would be able to receive all packets that pass through the router, and thus, all packets handled by the Jabberd server.

Although Jabberd does not have a facility for logging all packets, such a facility could be easily provided by specifying a user with 'acl = 'log'' permissions and creating a component that would connect to the router as this user. Such a component would then be able to log all Jabberd packets.

View as PDF		
Back	Up	Next

Comments

9. Router−users.xml Configuration

The sole purpose of the `router−users.xml` file is to provide a list of user and password pair(s) for authentication by the *router*. The default user (`jabberd`) is the user against which the other Jabberd components (*sm*, *resolver*, *s2s* and *c2s*) authenticate.

The `router−users.xml` file probably does not deserve its own section, ***except to encourage strongly that all Jabberd administrators change the default password on production servers***. Using the default password (`secret`) leaves your Jabberd installation open to easy attack by malicious crackers.

Once the default password in `router−users.xml` is changed, the corresponding passwords must be changed in the following files:

```
sm.xml
resolver.xml
s2s.xml
c2s.xml
```

These passwords are found in the sections labeled `Router connection configuration` near the top of the above listed files.

View as PDF
Back | Up | Next

Comments

10. Sm.xml Configuration

The `sm.xml` file configures the *session manager* component of Jabberd 2. The *session manager* acts as a layer between the router and the externally available components — s2s and c2s. The `sm.xml` file configures the following functions:

- Jabberd Identification
- Communication with the Router
- Logging

- Database Connection and Configuration
- Access Control for Administrative Functions
- Modules that Are Called during Sessions
- Static Discovery Settings for Legacy Components
- User Options

Below is an overview of the settings in the `sm.xml` file.

10.1. Jabberd Identification

The first two sections of `sm.xml` specify the ID on the network and the PID file to write. The `id` section is important because it is this ID that becomes appended to user names to form a JID (Jabber Identification). For most installations, `id` will be the hostname — something like `somedomain.com`. This ID must be resolvable via DNS if the server is to communicate with other Jabber servers; therefore, it need not be resolvable for closed or local Jabberd systems. An IP address may be used as the 'id'; however, because an IP address is not resolvable via DNS, a system based on an IP address ID would not be able to communicate with other Jabber servers.

The `pid` section specifies the location of the PID file. This may be commented out if a PID file is not needed.

10.2. Communication with the Router

The `router` section controls communication with the *router* component. The default `ip` and `port` should be fine for most installations, although note that if the *session manager* is running on a separate server, an external IP address would be specified here.

The `user` and `pass` sub−sections specify the user name for connecting to the *router*. These must match against a pair specified in `router−users.xml` as explained in Section 9. Basic security procedures dictate that the default password should be changed for production systems.

The `pemfile` section specifies the certificate and private key to be used for communication with the *router*. See Section 6.3 and Appendix: Generating A Self−Signed SSL Key for more information about setting up SSL on Jabberd. Commenting this section has the effect of disabling SSL communication between the *session manager* and *router*.

The `retry` section specifies how the *session manager* should try to reconnect to the *router* if the connection cannot be established during startup or if the connection is lost during operation. The default settings prevent the *session manager* from indefinitely attempting to reconnect if this connection cannot be made. These default settings will essentially cause the *session manager* to die if the router dies or is killed.

10.3. Logging

Jabberd logging defaults to the `syslog`. If you prefer the *session manager* to write its own log file, change the `log type` to `file`, and specify a location for the log.

10.4. Database Connection and Configuration

The *session manager* handles the database connection referred to in this guide as the *application data package* (as opposed to the *authentication data package*). You need to edit only the sub−section applicable to your choice of *application data package* (MySQL, PostgreSQL or Berkeley DB). For database packages running on the same server as the *session manager* you should need to edit only the password used to connect to the

database (this is not necessary for Berkeley DB). Change the `host` for databases running on separate machines. Note that the default password (`secret`) should not be used on production machines.

10.5. Access Control for Administrative Functions

The `aci` section controls which users have access to administrative functions. Among these functions are broadcast, messages and discovery. Note that specification of an administrative user in this section does not cause that user to be created in the authentication (*authreg*) database. Thus, you must register a user separately. Section 7.4 describes how to use an administrative account to send messages to all users.

Administrative functions currently have only minimal support in Jabber clients; however, the JAJC client for Windows supports the administration functions discovery of online users and messages to online users.

10.6. Modules that Are Called during Sessions

The `modules` section controls module chains that are called by specific events. Events are defined in terms of receipt or delivery of a type of packet. These module chains should be edited only by experienced Jabberd administrators. Of note are the chains that are called during user creation (`user-create`) and user deletion (`user-delete`).

10.7. Static Discovery Settings for Legacy Components

The `discovery` section contains both standard service information for instant messaging (the `identity` sub-section) in addition to settings for static discovery settings for legacy components (the `items` sub-section). If your server is running legacy components, such as JUD, MU Conferencing, MSN-T or Aimtrans, you should complete an `item` sub-section if the component does not support *disco* (discovery). Follow the examples shown in this section to add an item for static discovery for your legacy components.

10.8. User Options

The `user` section specifies templates that should be added to a user's data store when the user is created, and this section also specifies whether non-existent user should be created when that user logs in. This option allows an administrator to add users to the authentication database (*authreg*) and then have those users be created the first time they log in.

View as PDF		
Back	Up	Next

Comments:


Jabberd 2 Documentation Project

11. Resolver.xml Configuration

The `resolver.xml` file configures the router component which handles hostname resolution for the *s2s* (server to server) component. The `resolver.xml` file configures the following functions:

- PID File
- Communication with the Router
- Logging

Configuration of the *resolver* is concerned primarily with how the *resolver* and *router* components communicate.

11.1 PID File

The `pid` section specifies the location of the PID file. This section may be commented if a PID file is not needed.

11.2. Communication with the Router

The `router` section controls communication with the *router* component. The default `ip` and `port` should be fine for most installations, although note that if the *resolver* is running on a separate server, an external IP address would be specified here.

The `user` and `pass` sub−sections specify the user name for connecting to the *router*. These must match against a pair specified in `router−users.xml` as explained in Section 9. Basic security procedures dictate that the default password should be changed for production systems.

The `pemfile` section specifies the certificate and private key to be used for communication with the *router*. See Section 6.3 and Appendix: Generating A Self−Signed SSL Key for more information about setting up SSL on Jabberd. Commenting this section has the effect of disabling SSL communication between the *resolver* and *router*.

The `retry` section specifies how the *resolver* should try to reconnect to the *router* if the connection cannot be established during startup or if the connection is lost during operation. The default settings prevent the *resolver* from indefinitely attempting to reconnect if this connection cannot be made. These default settings will essentially cause the *resolver* to die if the router dies or is killed.

11.3. Logging

Jabberd logging defaults to the `syslog`. If you prefer the *resolver* to write its own log file, change the `log type` to `file`, and specify a location for the log.

View as PDF		
Back	Up	Next

Comments:


Jabberd 2 Documentation Project

12. S2s.xml Configuration

The `s2s.xml` file handles configuration for the server−to−server Jabberd component. The `s2s.xml` file provides network settings for this component in addition to settings for communicating with the router component:

- PID File
- Communication with the Router
- Logging
- Network Configuration
- S2S Connection Checking

Below is an overview for the settings in the `s2s.xml` file.

12.1. PID File

The `pid` section specifies the location of the PID file. This section may be commented if a PID file is not needed.

12.2. Communication with the Router

The `router` section controls communication with the *router* component. The default `ip` and `port` should be fine for most installations, although note that if *s2s* is running on a separate server, an external IP address would be specified here.

The `user` and `pass` sub−sections specify the user name for connecting to the *router*. These must match against a pair specified in `router−users.xml` as explained in Section 9. Basic security procedures dictate that the default password should be changed for production systems.

The `pemfile` section specifies the certificate and private key to be used for communication with the *router*. See Section 6.3 and Appendix: Generating A Self−Signed SSL Key for more information about setting up SSL on Jabberd. Commenting this section has the effect of disabling SSL communication between *s2s* and *router*.

The `retry` section specifies how *s2s* should try to reconnect to the *router* if the connection cannot be established during startup or if the connection is lost during operation. The default settings prevent *s2s* from indefinitely attempting to reconnect if this connection cannot be made. These default settings will essentially cause *s2s* to die if the router dies or is killed.

12.3. Logging

Jabberd logging defaults to the `syslog`. If you prefer *s2s* to write its own log file, change the `log type` to `file`, and specify a location for the log.

12.4. Network Configuration

The `local` section specifies network configuration for *s2s* in addition to the secret used for dialback keys. The default IP address and port should be fine for most installations (the 0.0.0.0 setting allows *s2s* to listen on all available IP addresses). The `secret` setting specifies the passphrase that *s2s* uses to generate dialback keys for other Jabber servers. The default setting of `secret` should be changed on production servers.

12.5. S2S Connection Checking

The `check` section handles checking of connections with other Jabber servers. By default, these checks are disabled (`interval` is set to `0`). To enable checking, set an `interval` in seconds, and then set intervals for queue expiry, invalid route expiry and/or keep alives.

|| QUESTION: Under what conditions would connection checking be useful? ||

View as PDF		
Back	Up	Next

Comments:


Jabberd 2 Documentation Project

13. c2s.xml Configuration

The `c2s.xml` file configures the client−to−server Jabberd component. The *c2s* component handles communications with Jabber clients, and the settings in `c2s.xml` are primarily concerned with client communication:

- PID File
- Communication with the Router
- Logging
- Network Configuration
- Input/Output Control
- Client Authentication and Registration
- Authentication Encryption

Below is an overview of the settings in the `c2s.xml` file.

13.1. PID File

The `pid` section specifies the location of the PID file. This section may be commented if a PID file is not needed.

13.2. Communication with the Router

The `router` section controls communication with the *router* component. The default `ip` and `port` should be fine for most installations, although note that if *c2s* is running on a separate server, an external IP address would be specified here.

The `user` and `pass` sub−sections specify the user name for connecting to the *router*. These must match against a pair specified in `router−users.xml` as explained in Section 9. Basic security procedures dictate that the default password should be changed for production systems.

The `pemfile` section specifies the certificate and private key to be used for communication with the *router*. See Section 6.3 and Appendix: Generating A Self−Signed SSL Key for more information about setting up SSL on Jabberd. Commenting this section has the effect of disabling SSL communication between *c2s* and *router*.

The `retry` section specifies how *c2s* should try to reconnect to the *router* if the connection cannot be established during startup or if the connection is lost during operation. The default settings prevent *c2s* from indefinitely attempting to reconnect if this connection cannot be made. These default settings will essentially cause *c2s* to die if the router dies or is killed.

13.3. Logging

Jabberd logging defaults to the `syslog`. If you prefer *c2s* to write its own log file, change the `log type` to `file`, and specify a location for the log.

13.4. Network Configuration

The `local` section specifies network configuration for *c2s*. For most installations, the `id` should be the same as the ID specified in `sm.xml`. The `realm` attribute can be used in cases where multiple jabber servers are relying on the same authentication database. This might be the case for a large company with several Jabber servers that authenticate via LDAP. The default IP address and port should be fine for most installations (the 0.0.0.0 setting allows *c2s* to listen on all available IP addresses).

The `pemfile` sub−section of the `local` section specifies the certificate and private key to be used for *client* communications. See Section 6.3 and Appendix: Generating A Self−Signed SSL Key for more information about setting up SSL on Jabberd. Commenting this section has the effect of disabling SSL communication between Jabberd and clients. Note that this key pair does not need to be the same key pair used for internal Jabberd communications.

13.5. Input/Output Control

The `i/o` section controls the following input/output options:

- Connection Limiting
- Rate Limiting
- IP Access Control

Note that the default settings for these subsections should be fine for most installations.

13.5.1. Connection Limiting

Jabberd sets a limit on the number of connections via the `max_fds` (maximum file descriptors) setting. Jabberd uses a file descriptor for each connection. Thus, setting a maximum number of file descriptors for *c2s* has the effect of limiting the number of concurrent *client* connections for the Jabberd server. As the `c2s.xml` file notes, *c2s* itself can use up to 5 connections (with other Jabberd components); therefore, the maximum number of file descriptors set here is actually slightly greater than the number of potential concurrent connections.

13.5.2. Rate Limiting

The `limits` subsection dictates throttling for individual connections. This section is tantamount to a simplified *karma* setup as found in Jabberd 1.4. The default for both `bytes` and `connects` is `0`. Thus, these limits are disabled by default. Administrators of servers under heavy load may wish to set limits here to prevent users from controlling excessive server resources. The `c2s.xml` file contains examples for setting rate limiting on connections.

13.5.3. IP Access Control

The `access` subsection specifies IP addresses that should be allowed or denied access to *c2s*. IP addresses denied access to *c2s* cannot have their packets handled and are thus denied access to Jabberd server functions.

The `order` subsection specifies the order of rule checking (checking of allow rules then deny rules versus checking of deny rules then allow rules). IP restrictions are set using either an `allow` or a `deny` tag below the `order` within the `access` subsection. Omission of both a deny and allow rule causes all connections to be accepted — as is the default setting.

13.5.4. Client Connection Checking

The `check` section handles checking of connections with other Jabber clients. By default, these checks are disabled (`interval` is set to `0`). To enable checking, set an `interval` in seconds, and then set intervals for queue expiry, invalid route expiry and/or keep alives.

13.6. Client Authentication and Registration

The `authreg` section controls aspects of client authentication and registration:

- Authentication Package
- Public Registration
- Password Change
- Authentication Mechanisms
- Authentication Package Configuration

These sub−sections are described below.

13.6.1. Authentication Package

The `module` subsection specifies the authentication data package to use. Jabberd sets this `module` during build according to your configuration choice, and so you should not need to edit this unless you change the authentication data package after installing Jabberd.

13.6.2. Public Registration

The `enable` subsection of `register` controls whether registration is publicly open for new users. Public registration is enabled by default. Comment the `enable` tag to disable public registration.

13.6.3. Password Change

The `password` subsection of `register` specifies whether users can change their own passwords. Password change is disabled by default. As `c2s.xml` notes, it may be useful to enable password change if public user registration is disabled.

13.6.4. Authentication Mechanisms

The `mechanisms` sub−section specifies which authentication encryption methods will be offered to clients. As `c2s.xml` notes, you should comment out any mechanisms that you do not want to offer.

13.6.5. Authentication Package Configuration

The last sub−section of `authreg` controls connectivity with your authentication data package. The default settings should be fine for most installations unless your authentication data package is running on a separate server.

View as PDF

Back | Up

Comments:

2003/12/16 01:34 PST
If I use MySQL as the data and auth package, and at the same time I comment the enable tag to disable public registration, then user's registeration behavior will crash jabber server (V 2).

Will, 2003/12/16 09:14 PST
Please post Jabberd 2 issues to the Jabberd List. *−Ed.*


Jabberd 2 Documentation Project

A.1. Jabberd 2 Quick Start Guide

This guide is intended to get Jabberd 2 installed and running as quickly as possible:

1. Install OpenSSL

2. Install Berkeley DB
3. Create Jabber User and Group
4. Create Directories for Data and Logging
5. Install Jabberd 2
6. Configure Server
7. Test Server

Jabberd 2 can use MySQL, PostgreSQL or Berkeley DB to store its data. MySQL is the recommended database for Jabberd 2; however, this guide suggests using Berkeley DB because it requires the least amount of configuration for Jabberd 2 data storage. See the Jabberd 2 Installation and Administrative Guide for a detailed guide to installing and configuring Jabberd 2.

A.1.1. Install OpenSSL

Jabberd 2 requires that OpenSSL (version 0.9.6b or higher) be installed prior to installing Jabberd 2. If OpenSSL version 0.9.6b is not installed on your system, see the OpenSSL site or Installing OpenSSL for Jabberd.

A.1.2. Install Berkeley DB

Jabberd 2 requires Berkeley DB version 4.1.24 or higher; if this is not installed on your system, see the Berkeley DB site or Installing Berkeley DB for Jabberd 2.

Consideration should be exercised when choosing a data store for a Jabberd 2 production server because converting from one database to another may be difficult.

A.1.3. Create Jabber User and Group

You should create a specific `jabber` user and group to run the server:

```
su
groupadd jabber
useradd -g jabber jabber
```

Note that the above commands are intended as an example. The commands and parameters for adding a user and group may vary for your system. Consult your manuals if you have any doubt about these commands.

A.1.4. Create Directories Data, Logging and PID's

You should create directories for data, logging and PID's, and your jabber user will need read and write permissions on these directories.

A.1.4.1. Create Data Directory

Create a directory to store the Jabberd database files (as superuser):

```
mkdir -p /usr/local/var/jabberd/db
```

A.1.4.2. Create Log Directory

Create a directory to store Jabberd log files (as superuser):

```
mkdir -p /usr/local/var/jabberd/log
```

A.1.4.3 Create PID Directory

Create a directory to store Jabberd PID files (as superuser):

```
mkdir -p /usr/local/var/jabberd/pid
```

A.1.4.4. Set Ownership for Data and Log Directories

Change the ownership of the directories created above (as superuser). If you used the locations specified above, enter the command:

```
chown -R jabber:jabber /usr/local/var/jabberd
```

A.1.5. Install Jabberd 2

This section describes how to download, configure, build and install Jabberd 2 on your system.

A.1.5.1. Download Jabberd

Download the file `jabberd-2.n.tar.gz` from the Jabberd 2 Releases page, where "n" is the latest version of Jabberd 2.

Download the file referenced above into a convenient directory for building the installation files. At the time of writing, Jabberd 2 RC2 is the latest version and is used in the examples below.

A.1.5.2. Extract Jabberd Installation Files

Change to the directory where you downloaded the file above and then extract the Jabberd 2 files by running the command:

```
tar -zxvf jabberd-2.0rc2.tar.gz
```

A.1.5.3 Configure the Jabberd Build

Change to the directory created above:

```
cd jabberd-2.0rc2
./configure --enable-authreg=db --enable-storage=db
```

The command above will configure your build to use Berkeley DB as the data store. You may also wish to enable debugging. To do this, add `--enable-debug` to the configuration options above. For help with configuration options, enter `./configure --help`.

A.1.5.4. Build Jabberd

Build Jabberd buy running the command:

```
make
```

A.1.5.5. Install Jabberd

Switch to the super−user:

```
su
```

Run make install:

```
make install
```

A.1.5.6. Default File Locations

Your Jabberd 2 installation is complete. Below is a listing of file locations for the default installation:

```
/usr/local/etc    Jabberd Configuration Files
/usr/local/bin    Jabberd Binaries (jabberd, c2s, resolver, router, s2s, sm)
```

A.1.6. Configure Server

The most basic Jabberd configuration (when using Berkeley DB) requires a total of 4 configuration edits:

1. Set *hostname ID* in `c2s.xml`
2. Set *authreg module* to use in `c2s.xml`
3. Set *hostname ID* in `sm.xml`
4. Set *storage module* to use in `sm.xml`

The configuration files are all found in `/usr/local/etc/jabberd`.

A.1.6.1. Set hostname ID in `c2s.xml`

In `c2s.xml` edit the `id` tag under the section labelled `local network configuration` so that the `id` references the fully qualified domain name (FQDN) of your jabber server. For example, using the FQDN of `jabber.somedomain.com`, your `c2s.xml` configuration would appear as below:

```
<local>
  <!-- Who we identify ourselves as. This should correspond to the
       ID (host) that the session manager thinks it is. You can
       specify more than one to support virtual hosts, as long as you
       have additional session manager instances on the network to
       handle those hosts. The realm attribute specifies the auth/reg
       or SASL authentication realm for the host. If the attribute is
       not specified, the realm will be selected by the SASL
       mechanism, or will be the same as the ID itself. Be aware that
       users are assigned to a realm, not a host, so two hosts in the
       same realm will have the same users.
       If no realm is specified, it will be set to be the same as the
       ID. -->
  <id>jabber.somedomain.com</id>
```

Note that this `id` must be resolvable by the clients that will be connecting to your Jabberd server.

A.1.6.2. Set authreg module to use in `c2s.xml`

Further down in `c2s.xml` is a section labeled `Authentication/registration database configuration`. This is where the *authreg* data module is specified. Edit the `module` tag so that Berkeley

DB is specified as the `Backend module to use`. Throughout Jabberd configuration, Berkely DB is abbreviated to `db`. Therefore, you should edit the `module` tag as below:

```
<!-- Authentication/registration database configuration -->
<authreg>
  <!-- Backend module to use -->
  <module>db</module>
```

A.1.6.3. Set hostname ID in `sm.xml`

At the top of `sm.xml` is the `id` setting for hostname. Edit the `id` tag with the same hostname specified in section A.1.6.1 above:

```
<sm>
  <!-- Our ID on the network. Users will have this as the domain part of
       their JID. If you want your server to be accessible from other
       Jabber servers, this ID must be resolvable by DNS.s
       (default: localhost) -->
  <id>jabber.somedomain.com</id>
```

A.1.6.4. Set storage module to use in `sm.xml`

Further down in `sm.xml` is a section labeled `Storage database configuration`. This is where the *storage* data module is specified. Edit this section as below so that Berkely DB is specified as the *storage* data driver to use:

```
  <!-- Storage database configuration -->
  <storage>
    <!-- By default, we use the MySQL driver for all storage -->
    <driver>db</driver>
```

A.1.7. Test Server

If you have created the logging and data directories specified above, you should be able to start your Jabberd 2 server and connect to it.

A.1.7.1. Starting Jabberd 2

Start your Jabberd 2 server by using the start up script:

```
su
su jabber
/usr/local/bin/jabberd
```

If your server fails to start, you can start Jabberd 2 with the debug option (note that this requires building Jabberd 2 with the debug option — see section A.1.5.3 above):

```
/usr/local/bin/jabberd -D
```

A.1.7.2. Connecting to your Jabberd Server

You should now be able to connect to your Jabberd server, and create a new account. If you are unable to connect, make certain that the hostname used (sections A.1.6.1 and A.1.6.3) is resolvable from the machine on which the client is running.

A.1.8. Further Configuration

Once your server is running, you may wish to configure it further. See especially section 6, Common Configuration Tasks, for further information about Jabberd configuration.

View as PDF
Up

Comments

2003/12/03 10:31 PST
Section 5.6 With rc1 you must also edit c2s.xml and set the db and sm.xml db before continuing.

2003/12/03 10:32 PST
Previous comment should have stated the c2s.xml module section and the sm.xml driver section set to db.

2003/12/05 12:07 PST
Looks like this hasn't been updated since alpha releases. Any significant changes in RC1? Thanks!

Will, 2003/12/05 12:18 PST
True, the quick configuration guide hasn't been updated in some time. I'll get on it this weekend. The biggest change is that Berkeley DB is no longer the default database. To use Berkely DB, users need to set it as the db module in both c2s.xml and sm.xml −Ed.

Will, 2003/12/20 08:16 PST
Quick Start guide is now up to date.

 Jabberd 2 Documentation Project

A.2. Installing OpenSSL for Jabberd 2

Jabberd 2 requires that OpenSSL (version 0.9.6b or higher) be installed prior to installing Jabberd 2. This appendix describes how to download the OpenSSL source in order to build and install it for Jabberd 2.

A.2.1. Download OpenSSL Installation File

Download the file `openssl-0.9.nz.tar.gz` from the OpenSSL site, where "nz" is the latest stable version of OpenSSL. At the time of writing, OpenSSL 0.9.7b' is the most current OpenSSL version and is used in the examples below.

Note that the file above contains encryption software. Follow your local laws if you download it.

A.2.2. Extract OpenSSL Installation Files

Change to the directory where you downloaded the file and then extract the OpenSSL files by running the command:

```
tar −zxvf openssl−0.9.7b.tar.gz
```

A.2.3. Build OpenSSL

Change to the OpenSSL directory just created:

```
cd openssl−0.9.7b
```

Configure OpenSSL for your system:

```
./config −−prefix=/usr shared
```

By default, OpenSSL installs to `/usr/local/openssl`. Using the `−−prefix=/usr` option causes the OpenSSL libraries to be installed to the `/usr` directory. Using this option makes OpenSSL more accessible not only to Jabber, but also to other applications. The `shared` option causes the build to create shared libraries. Read the `INSTALL` file if you are in doubt about options for your system.

Build OpenSSL on your system:

```
make
```

You can test your OpenSSL build if you wish:

```
make test
```

There should be no errors in the output.

A.2.4. Install OpenSSL

Switch to the super−user:

```
su
```

Install OpenSSL on your system:

```
   make install
OpenSSL installation is now complete &#151; unless the 'make install' output shows errors.  I
```

View as PDF
Up

A.3. Installing Berkeley DB for Jabberd 2

Berkeley DB (version 4.1.24 or higher) is one of the three database systems that can be configured to provide data storage for Jabberd 2. This appendix describes how to download the Berkeley DB source in order to build and install if for Jabberd 2. The advantage of using Berkeley DB for Jabberd data storage is that Berkeley DB requires no configuration, maintenance or administration after it is installed. A.3.1. Download Berkeley DB Installation Files

Download the file `db-4.1.nn.tar.gz` from the Berkeley DB Downloads page, where "nn" is the latest version of Berkeley DB. At the time of writing, Berkeley DB 4.1.25 is the most current version, and is used in the examples below.

Note that if you choose a version with strong encryption, you should follow your local laws if you download it.

A.3.2. Extract Berkeley DB Files

Change to the directory where you downloaded the file above and then extract the Berkeley DB files by running the command:

```
tar -zxvf db-4.1.25.tar.gz
```

At the time of writing, there is a patch for Berkeley DB version 4.1.25. You can apply this patch by first downloading the patch to the directory where you extracted the Berkeley DB files, for example `db-4.1.25`. Then run `patch -p0 < patchfile` to apply the patch, for example:

```
patch -p0 < patch.4.1.25.1
```

A.3.3. Build Berkeley DB

Change to the `build_unix` directory of the Berkeley DB directory just created:

```
cd db-4.1.25/build_unix
```

Configure Berkeley DB for your system. Berkeley DB uses its directory structure to identify the operating system type, so the "configure" command is different from most:

```
../dist/configure
```

Build Berkeley DB on your system:

```
make
```

A.3.4. Install Berkeley DB

Switch to the super-user:

```
su
```

Install Berkeley DB on your system:

```
make install
Berkeley DB installation is now complete &#151; unless the 'make install' output shows errors
```

View as PDF
Up


Jabberd 2 Documentation Project

A.4. Installing MySQL for Jabberd 2

This appendix describes how to install MySQL for Jabberd 2. MySQL is one of the three database systems that can be configured to provide data storage for Jabberd 2. The advantage of using MySQL is that it is fast and robust. Additionally, there are good documentation and support for MySQL.

A.4.1. Install from Source or Binaries

Pre−compiled RPM's are available, and the MySQL maintainers recommend that RPM's be used for installing MySQL. See MySQL 4.0 Downloads for the RPM's. Read the note below, if you install from RPM's. Otherwise, skip to A.4.2. to begin installing from the source code.

***I** Important: Minimum MySQL Version*
Jabberd 2 requires MySQL version 4.0 or higher.

***I** Important: Required MySQL RPM's*
Jabberd 2 requires that the following 4 RPM's be installed: Server; Client Programs; Libraries and Header files; and Dynamic Client Libraries.

A.4.2. Create MySQL User and Group

You will want to create a `mysql` user and group to run the MySQL server:

```
su
groupadd mysql
useradd −g mysql mysql
```

***I** Important: Check Your User and Group Commands*
The above commands are intended as an example. The commands and parameters for adding a user and group may vary for your system. Consult your manuals if you have any doubt about these commands.

A.4.3. Download MySQL Installation Files

Download the file `mysql-4.0.15a.tar.gz` from the MySQL Downloads page, where "nn.nn" is the latest version of MySQL. This download file is listed as the source tarball downloads near the bottom of the page. At the time of writing, MySQL 4.0.15 is the most current version, and is used in the examples below.

A.4.4. Extract MySQL Files

Change to the directory where you downloaded the file above and then extract the MySQL files by running the command:

```
tar -zxvf mysql-4.0.15.tar.gz
```

A.4.5. Build MySQL

Change to the `build_unix` directory of the MySQL directory just created:

```
cd mysql-4.0.15
```

Configure MySQL for your system:

```
./configure
```

Build MySQL on your system:

```
make
```

A.4.6. Install MySQL

Switch to the super-user:

```
su
```

Install MySQL on your system:

```
make install
```

If you receive errors, refer to the MySQL Documentation site.

A.4.7. Set Root Password for MySQL

MySQL uses its own passwords, which are completely separate from the operating system passwords. When MySQL is installed, the root MySQL password is blank, so you should create a root MySQL password as soon as you install MySQL. Use the following command (with your own password) to create the password (as superuser):

```
/usr/local/bin/mysqladmin -u root password 'new-password'
```

A.4.8. Copy Preference File to /etc

The MySQL install process does not automatically install a MySQL preference file to `/etc/`. Instead, sample configuration files are contained in the `support-files` directory of the source code. Choose one

that fits your requirements (large, medium or small installation), and copy it to `/etc` so that the preference file has the name `my.cnf` (as superuser):

```
cd support files
cp my-small.cnf /etc/my.cnf
```

A.4.9. Set Ownership for Data Directories

Set the ownership for the MySQL data directories using the user and group created above (as superuser):

```
chown -R mysql:mysql /usr/local/var/mysql/
```

A.4.10. Create Symlinks for Shared Libraries

Create symlinks for the MySQL shared libraries so that Jabberd 2 can locate them (as superuser):

```
ln -s /usr/local/include/mysql/ /usr/include/mysql
ln -s /usr/local/lib/mysql/ /usr/lib/mysql
```

A.4.11. Set MySQL to Start at Boot

Set the the MySQL daemon to start at boot as the mysql user created above. Boot scripts vary by distribution and architecture and so are beyond the scope of this guide. Consult your user documentation to set `/usr/local/libexec/mysqld` to start at boot time.

A.4.12. Test MySQL

Your MySQL server is ready to be tested. Start the server (as superuser):

```
/usr/local/bin/mysqld_safe --user=mysql &
```

You should now be able to connect to the server as the MySQL root users:

```
/usr/local/bin/mysql
```

This should provide a `mysql>` prompt. You can test your installation with a show databases command:

```
mysql> SHOW DATABASES;
```

You should see output like this:

```
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)
```

MySQL is now successfully installed on your system.

View as PDF
Up

Comments

2003/11/26 23:19 PST
who can tell me the db's doc, every field 's means and the table's means 2003/11/26 23:56 PST

There's a start to this at http://www.jabberdoc.org/section02.html .. it describes every table but not necessarily every field.

2003/12/15 05:23 PST
The document should emphasize that the version of MySQL must be 4.0 or higher, otherwise the jabberd 2 can not be compiled.

2003/12/22 10:21 PST
Updates on "Install from Source"

Following "2.3.1 Quick Source Installation Overview" of MySQL Documentation worked for me (mysql−4.0.17.tar.gz):

```
<−−snipp−−>
shell> groupadd mysql
shell> useradd −g mysql mysql
shell> gunzip < mysql−VERSION.tar.gz | tar −xvf −
shell> cd mysql−VERSION
shell> ./configure −−prefix=/usr/local/mysql
shell> make
shell> make install
shell> scripts/mysql_install_db
shell> chown −R root  /usr/local/mysql
shell> chown −R mysql /usr/local/mysql/var
shell> chgrp −R mysql /usr/local/mysql
shell> cp support−files/my−medium.cnf /etc/my.cnf
shell> /usr/local/mysql/bin/mysqld_safe −−user=mysql &
<−−snipp−−>
```

InnoDB is enabled by default.

Will, 2003/12/23 07:51 PST
I've changed the configuration above so that InnoDB *is enabled. −Ed.*

2003/12/24 03:51 PST
The tar−command under A.4.4. should look like: tar −zxvf mysql−4.0.15.tar.gz

Will, 2003/12/24 05:26 PST
Thanks for catching a typo! −Ed.


Jabberd 2 Documentation Project

A.5. Generating a Self−Signed SSL Certificate

This appendix describes how to generate a self−signed OpenSSL certificate for use with Jabberd. See Section 6 for a warning about using self−signed keys.

A.5.1. Generate Key Pair

From a working directory, enter the command below to begin an interactive key generation process:

```
openssl req −new −x509 −newkey rsa:1024 −days 3650 −keyout privkey.pem −out server.pem
```

You will be prompted for a passphrase for the private key. After entering and confirming your passphrase, you will be prompted for public information about your key.

N *Note: Common Name*
Note that you should enter your FQDN as the `Common Name` for your certificate.

N *Note: Key Lifetime*
Note that the command above creates a key with a 3650 day (10 year lifetime). To change the key lifetime, use a different number of days for the `−days` parameter.

A.5.2. Remove Passphrase

Enter this command to remove the passphrase from your private key:

```
openssl rsa −in privkey.pem −out privkey.pem
```

A.5.3. Combine the Private and Public Key

Enter this command to combine the private and public keys into a single file:

```
cat privkey.pem >> server.pem
```

A.5.4. Delete Private Key

You should now delete your private key:

```
rm privkey.pem
```

A.5.5. Move Key and Set Permissions

You can now move your key to its permanent location. For example, to move the key to the default Jabberd `pemfile` location, you would enter this command (as superuser):

```
mv server.pem /usr/local/etc/jabberd/server.pem
```

Then, you should set permissions on this file so that it is owned by superuser and is readonly (as superuser):

```
chown root:jabber /usr/local/etc/jabberd/server.pem
chmod 640 /usr/local/etc/jabberd/server.pem
```

Your certificate is now ready for use by Jabberd. You should make a backup (such as to a floppy) of your certificate.

View as PDF
Up

Comments

 Jabberd 2 Documentation Project

A.6. Jabberd for Corporate Use

This appendix provides some tips for installing Jabberd for use by a corporation or other private organization. These tips were originally posted by Ken Wermann on the Jadmin Mailing List :

- Install from Source
- Install to Unix
- Locate Server in DMZ or on LAN
- Enable Only SSL Communication
- Apply OpenSSL Patches
- Disable User Registration
- Use Strong Server Passwords
- Avoid Transports and Additional Services
- Log Conversations
- Use JAJC or Exodus Jabber Client

These tips are not meant as an exhaustive guide to installing Jabberd for corporate use.

A.6.1. Install from Source

Install the latest stable Jabberd server from source. Although there are many packages (RPM's, etc) available for Jabberd, users report various problems with some of these. At the time of writing, Jabberd 2.0s1 is the latest stable release, and it is available from Jabber Studio.

A.6.2. Install to Unix

Jabberd is native to Unix. Although libraries exist for installing Jabberd to Windows, you should make your corporate installation only to a flavor of Unix.

A.6.3. Locate Server in DMZ or on LAN

Administrators should give careful consideration to whether users may need to connect from outside the corporate firewall. If users will *never* need to connect from outside the firewall, then locate the Jabberd server on the corporate LAN. For the Jabberd host name, you can use any name that clients can resolve. Your

Jabberd server will not be able to communicate with other Jabber servers.

If you have users that need to connect from outside the firewall — even if only occasionally — you should locate your Jabberd server in a DMZ. Open port 5223 on both the DMZ and firewall to permit SSL encrypted Jabber communication. For your Jabberd host name, you can use a host on your domain, such as `jabber.mycompany.com`.

A.6.4. Enable Only SSL Communication

Even if your server is located on the corporate LAN, you should enable only SSL communications. Instant messaging traffic is easy to sniff on a network. See Section 6.3 for information about how to configure Jabberd 2 for SSL.

A.6.5. Apply OpenSSL Patches

Keep your OpenSSL installation up to date and apply the latest patches as they become available. You may wish to subscribe to the OpenSSL Mailing Lists.

A.6.6. Disable User Registration

Public user registration should be disabled. See Section 13.6.2 for information about how to disable public registration for Jabberd 2. Enabling user password change (Section 13.6.3) would be a good idea.

With public registration disabled, an administrator(s) will need to create user accounts. See the JabberStudio: Script Repository for user creation scripts that can be used with Jabberd 1.4. See Section 7.2 for information about how to create accounts with Jabberd 2.

A.6.7. Use Strong Server Passwords

Make certain that you change all default server passwords and secrets, and you should use strong passwords for these. These passwords include the password for your database connection and the password for your router connections.

A.6.8. Avoid Transports

Although Jabber transports (for foreign IM systems) can provide desirable features, administrators should avoid providing these services because they may create additional security vulnerabilities in addition to HR risks. On the other hand, JUD and Conferencing are recommended services for corporate installations. Note that at the time of writing JUD does not function properly with Jabberd 2.

A.6.9. Log Conversations

Check with you legal department to determine the requirements for message logging. Laws about message logging vary by country, state, province, etc., and these laws are often prescribe how long message logs must be preserved. This is especially true for certain industries, such as financial and healthcare.

Log IM messages if this is permitted or required. Inform your users that their conversations are being logged because this is good practice and because this will discourage inappropriate use of IM. See JabberStudio for utilities for monitoring Jabber traffic. Currently, there are no available utilities for logging messages on Jabberd 2.

A.6.10. Use JAJC or Exodus Jabber Client

JAJC is a good choice of a client for deployments running Jabberd 1.4 on Windows 2000 or higher because it is a mature, full−featured Jabber client that supports SSL.

Exodus is a good choice of a client for deployments running Jabberd 2. Like JAJC, Exodus is a stable client with many Jabber features. Additionally, Exodus supports SASL for authentication and TLS for channel encryption. Exodus is also a good choice for a clients running on older Windows systems.

View as PDF
Up

Comments

2004/01/05 10:55 PST
Easy Administration of Jabber2 Users by the Administrator

Hi, I am using phpMyEdit for creation of the user Ids on the mysql database. it is a php script to do straighforward addition, deletion &changing of the authreg table of the jabberd2 database. The only drawback is that there is no in−built security but by enabling digest based authentication on Apache we are atleast authenticating the person maintaining the user−ID's.

regards − Durga Prasad

www.datasoftindia.net


Jabberd 2 Documentation Project

A.7. Automatic Startup and Shutdown Using an RC Script

SysV initialization scripts and process controls provide one method of automatically starting and stopping Jabberd 2 when the machine comes up and goes down. This appendix describes how to implement SysV control for Jabberd on the Fedora 1.0 platform. To modify this process for your own distro, please consult your Linux or Unix documentation.

A.7.1. Download RC Script

An RC script for Jabberd 2 currently exists in the tools directory of the Jabberd 2 CVS Repository. Download the file `jabberd.rc` from CVS and copy it to the `/etc/rc.d/init.d` directory.

A.7.2. Rename RC Script

Rename the downloaded file to 'jabberd2':

```
mv −f /etc/rc.d/init.d/jabberd.rc /etc/rc.d/init.d/jabberd2
```

A.7.3. Assign Ownership and Permissions

The RC script file should be executable and owned by root:

```
chown −f root:root /etc/rc.d/init.d/jabberd2
chmod −f 0755 /etc/rc.d/init.d/jabberd2
```

A.7.4. Create Symbolic Links

You should create start and kill symbolic links in all appropriate run level directories:

```
ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc0.d/K15jabberd2

ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc1.d/K15jabberd2

ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc2.d/S85jabberd2
ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc2.d/K15jabberd2

ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc3.d/S85jabberd2
ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc3.d/K15jabberd2

ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc4.d/S85jabberd2
ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc4.d/K15jabberd2

ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc5.d/S85jabberd2
ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc5.d/K15jabberd2

ln −s /etc/rc.d/init.d/jabberd2 /etc/rc.d/rc6.d/K15jabberd2
```

A.7.5. Restart Jabberd 2

You should not be able to test your RC script:

```
/etc/rc.d/init.d/jabberd2 restart
/etc/rc.d/init.d/jabberd2 restart
```

The second invocation of restart should all result in [OK]. Note that if you execute redhat−config−services from the Fedora gui or execute ntsysv from a Fedora terminal or console session, you will now find jabberd2 in the list of services.

View as PDF
Up

A.8. Automatic Startup and Shutdown Using Daemontools

D.J. Bernstein's Daemontools provide another method of controlling Jabberd 2 through machine startup and shutdown. This appendix assumes that Daemontools are installed on your system. See How to Install Daemontools for installation instructions. See the Daemontools homepage for a description of Daemontools.

Daemontools configuration requires a run script in a dedicated directory. This directory is then symbolically linked to the `/service` directory for monitoring by the Svscan daemon of Daemontools. This how−to is a bit long because each of the Jabberd 2 binaries must have its own linked directory and run script. Additionally, the PID file writing should be disabled (in Jabberd XML configuration) for each of the binaries.

A.8.1. Create Master Directory

Create a master directory for the 5 Daemontools directories to be created below. The most convenient place for this directory is in `/usr/local/etc/jabberd/`. Create this new directory (as root):

```
mkdir /usr/local/etc/jabberd/daemontools
```

A.8.2. Create Run Directories for Binaries

Create a sub−directory for each of the Jabberd binaries:

```
mkdir /usr/local/etc/jabberd/daemontools/router

mkdir /usr/local/etc/jabberd/daemontools/resolver

mkdir /usr/local/etc/jabberd/daemontools/sm

mkdir /usr/local/etc/jabberd/daemontools/c2s

mkdir /usr/local/etc/jabberd/daemontools/s2s
```

You should now have 5 sub−directories under `/usr/local/etc/jabberd/daemontools/`.

A.8.3. Create Run Scripts

Each of the sub−directories now needs a run script for its respective Jabberd binary. Change to the `router` sub−directory to create your first run script:

```
cd /usr/local/etc/jabberd/daemontools/router
```

Using an editor, create a file called `run` in this directory. Edit the file as below so that it contains a run script for the router component:

```
#!/bin/sh
exec setuidgid jabber /usr/local/bin/router /usr/local/etc/jabberd/router.xml
```

Repeat this process for each of the remaining 4 sub−directories. You can use the above text as a template, and replace the 2 instances of `router` with the respective Jabberd binary (resolver, sm, c2s and s2s). For example, the `run` file in the `resolver` sub−directory should be created as below:

```
#!/bin/sh
exec setuidgid jabber /usr/local/bin/resolver /usr/local/etc/jabberd/resolver.xml
```

A.8.4. Make Run Scripts Executable

Each of the run scripts above needs to be executable. Run the command below to make each of the run scripts executable:

```
chmod 0755 /usr/local/etc/jabberd/daemontools/*/run
```

A.8.5. Disable PID's for each of the Jabberd Binaries

Svscan does not rely on PID files; therefore, PID file writing should be disabled in each of these configuration files:

```
router.xml
resolver.xml
sm.xml
c2s.xml
s2s.xml
```

Near the top of each of these files is a tag set for the PID file location. Comment these tags in each file above to disable PID file writing. For example, the top of `router.xml` should appear as below:

```
<!−− Router configuration −−>
<router>
  <!−− ID of the router on the network (default: router) −−>
  <id>router</id>

  <!−− The process ID file. comment this out if you don't need to know
       to know the process ID from outside the process (eg for control
       scripts) −−>
  <!−− <pidfile>/usr/local/var/jabberd/pid/router.pid</pidfile> −−>
```

A.8.6. Create Symbolic Links

Each of the sub−directories now needs to be linked to the `/service/` directory. Once these symlinks are created, svscan will start and monitor the respective process.

***I** Important: Stop Jabberd Server Before Proceeding*

In order to prevent conflicts, you should make certain that your Jabberd server is completely stopped before proceeding with the final step of Daemontools configuration.

Create 5 symlinks as below:

```
ln −s /usr/local/etc/jabberd/daemontools/router /service

ln −s /usr/local/etc/jabberd/daemontools/resolver /service

ln −s /usr/local/etc/jabberd/daemontools/sm /service
```

```
ln -s /usr/local/etc/jabberd/daemontools/c2s /service

ln -s /usr/local/etc/jabberd/daemontools/s2s /service
```

A.8.7. Check that Services are Running

Make sure that the `svscan` daemon is running on your machine. On my distro (Gentoo Linux), the `svscan` daemon is started from `/etc/init.d`. You should now be able to check the status of your Jabberd services by using the `svstat` command. For example, to check the status of the router component, you would switch to the `/service` directory and then enter the `svstat` command as below:

```
svstat router
```

You should see output similar to this:

```
router: up (pid 20011) 248 seconds
```

Svscan will now monitor your Jabberd processes, and if a process goes down, Svscan will restart it.

***N** Note: Svscan Should Be Set to Start at Boot*

If you just installed Daemontools, you may need to add the svscan daemon to an appropriate run level so that it starts during boot. Consult your distro's documentation for the best way to do this.

View as PDF
Up


Jabberd 2 Documentation Project

Jabberd 2 FAQ

General

Q. I've read the guide and the FAQ; however, I still have a question. Where can I search for answers?

A. There are several good sources for answers. The Jabber FAQ's cover general Jabber questions:

- ◊ Jabber General FAQ
- ◊ Jabber End User FAQ
- ◊ Jabber Development FAQ

You can also try searching the Jadmin Archive for questions about jabber administration, or subscribe to the Jadmin Mailing List to post your own question.

Q. Where is the Jabberd 2 homepage?

A. Jabberd 2 Homepage

Q. There's a feature that I'd like to see included with Jabberd 2. Where should I post my suggestion?

A. You can post a feature request to Jabberd 2 Feature Requests

Q. I think I've found a bug in Jabberd 2. Where should I post this information?

A. If you are confident that you've found a bug, you can post it to the Jabberd 2 Bug List, or you can

post a question to the Jabberd List.

Q. How does Jabberd 2 handle multiple domains?

A. You can add multiple ID's to the Client to Server (C2S) configuration; however, you have to setup individual instances of the session manager (SM) for each domain.

Q: When will components be available for Jabberd 2?

A: Existing components can all be made to work with Jabberd 2. Jabberd 2 does not have an internal component API, so shared object components are impossible.

Q: How do I get existing components to work with Jabberd 2?

A: "External" components (ie components that connect to a Jabber server over TCP using the "jabber:component:accept" protocol will work as−is. Simply configure the component to connect to the router using the port and secret specified in router.xml.

"Internal" or "library load" components that load at runtime into a jabberd 1.4 instance can be used by placing the 1.4 instance into "uplink" mode. An appropriate "linker" alias needs to be added to router.xml to support this.

Jabberd 2 does not provide XDB and logging facilities. For "internal" components, these can be provided by the enclosing jabberd 1.4 instance. For "external" components, they will need to connect to a 1.4 instance that provides these services, which in turn uplinks to the Jabberd 2 router.

Client−Side Issues

Q. Why does Jabberd 2 sometimes bounce messages sent to offline users?

A. Jabberd 2 handles messages to offline users slightly differently than was the case with Jabberd 1.4. Specifically, Jabberd 2 will bounce a message to an offline user if a resource is included in the address. To send an offline message to somebody you have to send it to its jabber ID without a resource.

Q. I just created a user in the database. Why can't I login as the user I just created?

A. It is not enough to add users to the `authreg` table because this only introduces users to the c2s component, but not to the sm component. Correct entries are required in the `active` table as well. It is best to use a Jabber client to register users. If this is not acceptable, corresponding entries can be manually created in the `active` table, or the session manager can be configured to create new users automatically the first time they log in. Uncomment the `auto−create` tag in the `User options` section of `sm.xml` to enable auto−creation of new users. (Note that enabling `auto−create` does not enable inband registration, as only users that pass the c2s component are created.)

Building and Installation

Q. Why does configuration fail to find my MySQL installation?

A. The most common reason why Jabberd cannot find the MySQL libraries is that not all of the required libraries are installed. In addition to the basic MySQL installation, Jabberd requires that the development libraries and headers be installed. Either perform a *Max* installation as listed on the MySQL Downloads page, or install *Server*, *Client Programs*, *Libraries and header files*, and *Dynamic client libraries* separately.

Q. Where can I find packages (RPM's, Debian, etc.) for Jabberd 2?

A. Debian packages for Jabberd 2 are to be released soon. RPM's are not available at this time.

System Specific

Q. On Redhat, why does ./configure fail?

A. The most likely reason for `./configure` to fail on Redhat is that Redhat ships with a MySQL installation that does not include the development libraries required by Jabberd 2. See the MySQL question under *Building and Installation* above.

Q. On Redhat 9, why does the build crash after a successful configure?

A. The most likely reason for the build to crash on Redhat 9 is that RH 9 ships with its own version of Kerberos. The OpenSSL libraries are found when `./configure` is run. However, Kerberos references cause the Jabberd 2 build to crash.

There is an easy work around for this problem. Execute the command below before running `./configure`:

```
export CFLAGS="-I/usr/kerberos/include"
```

Then run `./configure`, and build using the `-e` option:

```
make -e
```

View as PDF
Up

2004/01/08 06:23 PST
Rather than dumping kerberos or having to work around it in RH9 just do the following:

```
export CFLAGS="-I/usr/kerberos/include"
```

Then compile using make -e

Presuming that kerberos is installed this effectively just adds it to the include path but avoids having to edit the configure file

Will, 2004/01/11 09:34 PST
Thanks for the easy RH fix. I've changed the FAQ above. -Ed.