Server documentation

Overview 2
1. Submitted server part 3
CD content 3
2. Installation guide 4
Prerequisites 4
Compilation 4
Configuration 5
3. mujMail imap extension 7
mujMail extension detection 7
xmujMail-compression command 7
xmujMail-convert command 8
xmujMail-synccli command 9
xmujMail-syncsrv command 10
xmujMail-url command 11
4. Implementation details 13

Overview

MujMail server part was created in order to extend mujMail with some new capabilities and to create platform where new functionality can be added later without increasing the mobile part of the application significantly. Server part consists of enhanced Cyrus mail server (cyrusimap.web.cmu.edu). Extending IMAP protocol with new commands seemed to be the most logical way as it offers such possibility by specification (RFC 3501, http://www.faqs.org/rfcs/rfc3501.html, Section 6.5.1.). As it was mentioned above, this way there is no need to add much new protocol related stuff into mobile part and make mujMail bigger.

Cyrus server provides to the developers working mail processing, parsing and other functionality as it is a highly scalable enterprise mail system designed for use in enterprise environments of various sizes using standard based technologies. Cyrus server is used quite often and thus provide good base for enhancement.

1. Submitted server part

Server solution requires Linux based x86 computer to run. There are no additional hardware requirements to the server part. Installation (building and configuring) of the server is briefly described later in the installation section.

Current sources of mujMail server can be downloaded from svn by executing the following command:

svn co https://javaphone.svn.sourceforge.net/svnroot/javaphone/mujMail/src-server

1.1. Contents of the CD

Server related resources are stored in server directory on the submitted CD. The list of files with the short description follows:

cyrus-imapd-2.2.12.tar.gz	Original Cyrus IMAP server
imapd.conf	Example configuration file was used on mujMail-dev server
jpgtn-2.06.tar.gz	Software used to downscale pictures
links-1.00pre12.tar.gz	Software used to retrieve web-pages
pdftohtml-0.39.tar.gz	Software used to convert PDF attachments into html pages
mujMail-patch.diff	Patch that can be applied onto Cyrus IMAP server (2.2.12) to add mujMail extension
src-server.tar	MujMail server solution (Cyrus server with mujMail extension)
srv-doc.pdf	This document

Note: After applying patch to standard server, you have to run autoconf to update configure script.

Note: Due to case-sensitivity issue (Cyrus installation contains two directories which differ only in cases of letters (sieve and SIEVE), it is not possible to completely checkout server sources on Windows systems).

2. Installation guide

Installation process consists of compilation of the modified version of Cyrus server and some mujMail specific configuration. The assumption is that the users are able to compile standard Cyrus server and configure it. This guide will help with standard Cyrus configuration, for more details see Cyrus help (src-server/doc/install-*.html).

2.1. Prerequisites

At first, the user needs to checkout mujMail server sources (from svn, server/server.tar.gz, download stable version of Cyrus server (version 2.2.12 from http://cyrusimap.web.cmu.edu/) and apply the patch provided on the CD). The user also needs to install the prerequisites of Cyrus server, mainly; BerkeleyDB and SASL libraries, using packaging system of Linux distribution (see Linux documentation).

For mujMail extension, user will need to have installed zlib library for connection compression, free console web browser created at MFF links (http://links.sourceforge.net/), pdftohtml (http://sourceforge.net/projects/pdftohtml/) for viewing PDF attachments and jpgtn (http://sourceforge.net/projects/freshmeat_jpgtn/) for viewing downscaled JPEG files. First two are quite usual and typically are prepared in package manager of Linux distribution. Last two are open source tools from http://www.sourceforge.net and their sources can be found in the server directory on the CD.

To begin with installation, try to compile original Cyrus server from sources. And potential problems have to be solved with Cyrus installation manual or contact us for help. Most typical problems are missing libraries. Version 2.2.12 of Cyrus server have one small bug that prevents successful compilation. Patch for this bug and can be found, e.g. on https://bugzilla.andrew.cmu.edu/show_bug.cgi?id=2692. Patch for this problem is included in the CD.

Hint: If the user does not want to solve Cyrus configuration, he can try to install Cyrus server from package manager of his Linux distribution. Standard configuration will come with it.

Hint: More information about IMAP and cyrus can be found in the book Managing IMAP (by D. Mullet & K.Mullet, O'Reily ISBN 0-596-00012-X).

Note: jpgtn needs libjpeg library. Install it from your package manager.

2.2. Compilation

Compilation consists of standard triple (`configure && make && sudo make install`). MujMail extension does not have any additional requirements other than zlib library has to be available in your system. This requirement is tested in the end of configure script.

Before `make install` command create user `cyrus` and group `mail`, if not specified differently as configure script parameter. This step comes from standard Cyrus installation requirements.

There are no other requirements for the standard installation of mujMail server. Just execute `./configure` then `make` and, as last, `sudo make install` in src-server directory. If no error occurs, you will have the Cyrus server installation use in `/usr/cyrus/` directory. Cyrus is started by `/usr/cyrus/bin/master` program.

2.3. Configuration

In order to use the Cyrus server, you need to configure it properly (see /src-server/doc/install-configure.html).

To achieve this, the following steps have to be performed:

1) Modification of syslogd setting to log imap and sasl messages. Add the following lines:

```
local6.debug   /var/log/imapd.log
auth.debug     /vat/log/auth.log
```

2) Create configuration file /etc/imapd.conf. For the minimal configuration the following lines have to be added to imapd.conf file:

```
configdirectory: /var/imap
partition-default: /var/spool/imap
admins: alf admin anyone you want
sasl_pwcheck_method: saslauthd
```

   More information regarding configuration can found in man pages (or src-server/doc/man/imapd.conf.5.html). In server directory you can found configuration we use in our mujMail server. (server/imapd.conf)
   Note: This is most difficult step from configuration part.
   Hint: If Cyrus is using its own user database, admins are from Cyrus users database (not the system).
   Note: Admins should not use your boxes as ordinary users for retrieving emails.
3) Create "configdirectory" from configuration file. Typically create /var/imap/ directory and set owner to `cyrus`, group to `mail`, access rights to 750. (wrxr-x---).
   The same is have to be done for "default partition". Create specified directory and set same owner, group a rights.
4) Other directories can be created by Perl script (to be found at src-server/tools/mkimap)
5) Create configuration file for master process.
   Sample configurations file (src-server/mater/conf./master.conf) can be used to create the required configuration file. Store this configuration file as /etc/cyrus.conf.
   Probably you will not need to modify this file.
6) It is a good idea to configure your system to start Cyrus server (/usr/cyrus/bin/master) during system initialization.

These are the most important things. Probably you will want to configure Mail transfer agent to deliver email into Cyrus database. Consult these settings with documentation of Cyrus and your MTA (sendmail, postfix, …).

After standard configuration of Cyrus server you have to meet mujMail requirements:

1) In /var/imap/ directory create synchData subdirectory with `cyrus` user as owner, `mail` as a group and 750 access rights.
2) Same for convertData subdirectory: create and set owner, group and rights.
3) Build the prerequisites. Unpack jpgtn from archive, no special configuration is needed. Just run `./configure` and `make` in an unpacked directory. After the compilation try to run jpgtn program created in src directory. If it runs properly (usage help is printed on the console), copy created program into /var/imap/convertData directory. Change owner of copied program to `cyrus`, group to `mail`, and access rights to 750 (rwxr-x---).
4) Now unpack pdftohtml from archive and run `make` command in unpacked directory. After compilation, again, try if the program is created in src directory and is runnable and then copy them into /var/imap/convertData directory. Change owner of copied program to `cyrus`, group to `mail`, and access rights to 750 (rwxr-x---).
5) Ensure you have installed links browser by executing `links` command.

After successful configuration, start Cyrus server to check if it works correctly. (You are able to login, etc. …)

Hint: To manage mailboxes you can use cyradm utility.
Hint: If sasl library is using to authenticate a user, saslpasswd2 can be used to create users.
Note: You have to create both IMAP mailbox and authentication sasl account to permit user to download boxes.
Note: If you have troubles with authentication of users, you can put the following line into imapd.conf file:

sasl_pwcheck_method: alwaystrue.

With this setting of the server anyone will be able to log in and passwords are completely ignored.

Hint: To test your configuration you can use imtest utility.

3. MujMail IMAP extension

We extended IMAP protocol to support new commands in a consistent way. All the commands which are implemented for mujMail server are prefixed by X (X<atom> commands). It means that these commands are vendor specific extension. MujMail server introduces new commands: `xmujMail-url`, `xmujMail-compression`, `xmujMail-syncsrv`, `xmujMail-synccli` and `xmujMail-convert`.

3.1. MujMail extension detection

In order to be able to detect the presence of mujMail extension we add token MUJMAIL into result of IMAP's CAPABILITY command, which is a standard command, where all extensions are listed. To reduce the roundtrip and amount of transmitted data, we changed hello phrase that IMAP server sends to client automatically after establishing connection to contain keyword mujMail. Currently, this keyword is used to detect the presence of mujMail extension.

3.2. xmujMail-compression command

This command is used to setup the compression on the connection to client. Currently, we support no compression (default use) to be compatible with non-mujMail clients, RLE compression and GZIP compression. When you select RLE, communication is compressed in both directions and can be changed be subsequent `xmujmail-compression` command into different compression mode. When you enable GZIP compression, only outgoing data are compressed (most data are send in this direction), and this compression can not be due to buffering issues in compressions library changed into any different type.

Syntax:

```
{tag} xmujmail-compresision {version} {compression_type}
   {compression_specific_parameters}
```

Version: version of this command. Currently only 0.5 is supported. Newer version can add new compression or modify the ones used currently. Version parameter helps to distinguish these details.

Compression_type: currently one of these 3 keywords: NONE, RLE, GZIP.

Compression_specific_parameters: this entry is not currently used for any compression. Compression initialization parameters can be placed here.

Result:

OK – Everything good. Compression is set successfully. This OK line reply is the last line that is compressed by current compression settings. Next replies are sent with newly selected compression.

BAD – Command unknown or arguments are invalid. Bad syntax or unsupported compression type. No compression change happens.

Example:

C: 001 xmujMail-compression 0.5 RLE
S: 001 OK
This communication is compressed by selected compression

3.3. xmujMail-convert command

This command is used to retrieve converted attachments stored in used mailbox. User can retrieve PDF attachments of email converted into HTML format to be able to display it's content, and can change jpegs into smaller one, that are easier to handle and show in mobile phones.

Syntax:

```
{tag} xmujMail-convert {format_identification} {params}
   {UID} (binary[{bodypart_number}])
```

Format_identification: Specifies which conversion routine to use. Currently, only PDF and JPG are available to use.
Params: Format specific parameters used for conversion.
PDF – no parameters needed, skip this item
JPG – number that specifies longest side of converted picture in pixels
UID: UID of message in this mailbox which needs to be processed.
Bodypart_number: The number of attachment.

Result:

OK – Conversion was successful.
Before OK reply there is an untagged response (starts with * character as in ordinary IMAP commands) where UID of message is repeated then FETCH (binary {size of converted bodypart}, then, on the next line, sends the content of converted bodypard, this ends with new line and ending parenthesis
BAD – Command unknown or arguments invalid. Bad syntax or unknown bodypart or UID, …

Example:

C: 012 xmujMail-convert pdf 6 (BINARY[2]).
S * xmujmail-convert (UID 6 FLAGS () binary {4841}
S: <!DOC ….. more data here
S:)
S: 012 OK Completed

3.3. xmujMail-synccli command

This command is intended to retrieve the configuration previously stored by user. Cyrus server plays the role of textual storage. Stored information and their interpretation are a client task. From server point of view each stored entry is a block of textual information ended by doubled end line character. MujMail client stores here 3 types of information, information about accounts (servers, users …), settings of mujMail client and address book entries.

You need not to log into server to be able execute this command, and so user synchronization name and user synchronization password have to be sent as a command parameter. User names and passwords for synchronization are from different domain than standard Cyrus server user and passwords. This enables every user, even if they do not have account in our server to store their information here, and retrieve again, if needed. On the other hand, it can be perceived as a security risk to store your passwords on the server. mujMail client also enables to store the configuration in the local file if a mobile phone supports file system access. For the phones without file system support, there is the storage on mujMail server.

Syntax:

```
{tag} xmujMail-synccli {userID} {userPasswd}
```

UserID: user name that is used for synchronization purposes.
UserPasswd: password needed to get information about stored configuration

Result:

OK – in case that previous configuration was stored and userID exists and password match. Before OK reply tag, there are untagged responses for each stored entry. Each response starts with "* syncentry\n" and stored content. Note that one mail account is considered as one synchronization entry.
NO – no previous information was stored.
BAD – if bad syntax, parameter count, etc.

Example:

C: 001 xmujMail-synccli test9@server-dev.mujmail.org mmtest9
S: * Syncentry EntryType: Account
S: AccountClassType: Primary
S: Active: 1
S: Email: your_email@gmail.com
...some lines skiped
S: CopyToSrvTrashFolderName: trash
S:
S:
S: * Syncentry EntryType: Account
S: AccountClassType: Primary
S: Active: 1
S: Email: your_email@gmail.com1
...some lines skiped
S: CopyToSrvTrash: 1

S: CopyToSrvTrashFolderName: trash
S:
S:
S: * Syncentry EntryType: Contact
S: Name: MrBeen
S: Email: MrBeen@gmail.comy
S: Notes: xxx
S:
S:
S: * Syncentry EntryType: Settings
S: SettingsVersion: 20081106
S: mujMailSrvAddr: server-dev.mujmail.org
...some lines skiped
S: mujMailPassword: none
S:
S:
S: 001 OK Completed

Note: synccli is acronym for synchronize client.

3.4. xmujMail-syncsrv command

This command is the opposite of xmujMail-syncsrv command. This command stores information on a server. Stored information should by textual, line oriented. Server does not interpret information. Each entry is ended by doubled end line character (\n\n).

Server synchronization session starts with xmujMail-syncsrv command. This command is replied with untagged ready answer. After this successful answer, client can many times use subcommand for storing one entry. After storing all entries client sends "{tag} Done" command. Session is ended by server reply if storing was done successfully. Subcommand for storing entry consists of a "{tag} syncentry\n" and lines with data to store which are ended by double end line.

Syntax:

```
{tag} xmujMail-syncsrc {userID} {userPassword}
```

Result:

OK – all entries were successfully stored
BAD – bad command syntax, bad session state, invalid subcommand
NO – user name currently exists

Example:

C: 001 Xmujmail-syncsrv test9@server-dev.mujmail.org mmtest9
S: * 001 Ready
C: 001 Syncentry EntryType: Account
C: AccountClassType: Primary
C: Active: 1

C: Email: your_email@gmail.com
...some lines skiped
C: CopyToSrvTrashFolderName: trash
C:
C:
C: A2 Syncentry EntryType: Account
C AccountClassType: Primary
C: Active: 1
C: Email: my_email@gmail.com1
...some lines skiped
C: CopyToSrvTrashFolderName: trash
C:
C: 001 syncentry EntryType: Contact
C: Name: MrBeen
C: Email: MrBeen@gmail.comy
C: Notes: xxx
C:
C: 001 Syncentry EntryType: Settings
C: SettingsVersion: 20081106
C: mujMailSrvAddr: server-dev.mujmail.org
...some lines skiped
C: mujMailPassword: none
C:
C: 001 Done
S: 001 OK Competed

Note: syncsrv is acronym for synchronize server.

3.5. xmujMail-url command

This command can be used for retrieving web pages, pictures and files using HTTP protocol. It is indented for retrieving HTML pages referenced (a href HTML tag) by emails.

Syntax:

```
{tag} xmujMail-url {processing_type} {modify_specifier} {url}
```

Processing type: Two possibilities: Original or Processed.

Original processing type returns web page as it is stored on the server without any post-processing.

Processed processing type is intended to process email into mujMail specific format. Currently not supported

Modify specifier: It is here to specify if user wants the server to process the page. Currently only available processing mode is noncompacted. This mode does not modify HTML page at all.

Url: web address client wants to retrieve.

Result:

OK – after successful download of page
NO – if web page does not exist, or if there is a problem with processing or retrieving
BAD – if syntax of command is invalid

After command have been issued the reply is started with untagged response * xmujMail-url {size} tag: {tag} sending data, where size is size in bytes of retrieved web page that is send on next line immediately after this line. After sending context of web page tagged result response follows.

Example:

C: 001 xmujMail-url original noncompacted "http://www.mujmail.org/"
S: * xMujmail-url {4990} tag:001 sending data
S: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
S: <html>
.. some lines skipped
S: 001 OK XmujMail-url

4. Implementation details

Most important changes take place in `imap` directory. For some new commands we created separated file prefixed with `mujMail_`. Suffix corresponds with a command which code is stored here. IMAP protocol parsing in Cyrus takes place in the imapd.c file, especially in cmd_loop routine and is implemented as a one big switch. For each letter there is a list of `if elseif` conditions where strcmp function is used to identify proper command. Typical `if` branch handles first a few parameter and calls cmd_CommandName function that makes full parameter processing and starts command execution. We extended mujMail server in the same manner.

Newly created function for retrieving URL takes place in cmd_mujMail_url. This command processes the parameters, and forks. In one child, it executes `links` program, in the other one it waits for the result, processes them and sends the result to the client. In file mujMail_cvrt_format format there is the function that serves as the generic interface for converting different formats.

Synchronization stores information in textual files. Standard access rights setting protect ordinary users to see sensitive data stored here. Files are named after userID from synchronization command. For synchronization purposes there are supporting functions in mujMail_synch.c file. These library functions work as the wrappers for standard files and makes illusion that file consists of syncentries blocks. Most important functions save and read syncentry block. Load uses buffer for storing loaded data. Save function uses buffer with input data. For filling buffer there is support function getsyncdata that fills buffer from connection.

Conversion command uses external programs to process bodyparts. Bodyparts are retrieved from user mailbox by execution of a fetch command. Fetch result is then preprocessed, meaning checking of reply string. Fetch result is stored into temporary file and the appropriate conversion program is executed to convert the bodypart. Converted result is read from input file into memory buffer. Reply header is created and send to client.

Compressing connections involved changes in two places. We have to create new IMAP command to manage used compression and we also have to modify connection used by Cyrus server. New command addition is done in natural way, same as all other added commands as xmujMail-compression. In cmd_loop cycle, we only parse first two mandatory parameters and call cmd_mujMail_compression where business logic of processing parameters and setting compression takes place. In file mujMail_compression.h there is a COMPRESSION_TYPES enumeration, which is used as a parameter for setting connection. When changing compression we flush output buffers, to ensure that no previous data will be affected by change.

Conception of compression is influenced by way how Cyrus server processes incoming connections. For each connection, server runs new instance of imapd server and redirects network communication into its standard input and output. This provides security benefits, because only this redirecting process needs to have root rights. Imapd runs under special unprivileged user `cyrus`. In imapd process there is a prot_stream functionality. This quite independent part of Cyrus replaces standard C library for processing input and output. This proc_stream utility takes

care about buffering, applying SSL and SASL encoding, adds ability needed for interactive protocols (flush on read). This input processing library is stored in src-server/lib/prot.c file. We modify these functions to add the desired functionality. Compression has to be done before encryption when sending data to client, after encryption when reading data from client. This is essential due mobile part implementation.

Cyrus prot_stream implementation uses 2 buffers. One, called buf is a small buffer where the data is hold before they are processed by encryption or in other way. Second one is bigbuf, which is memory mapped file that is used as outgoing buffer (to be able to send data without blocking). This buffer contains encrypted data and is send without any processing into outgoing connection.

We add function prot_mm_setcompression to change compression mode used for connection. We add new entries intended for compression into connection structure. Main changes take place in prof_flush_encode function and in prot_fill_mm_buff. We modify nature how prot_stream works with buffers and add new mm_buf.

When reading data from client we modify behavior of buffers. Newly read data are stored into mm_buf. Then they are decrypted and stay in this mm_buffer. Standard buffer buf still holds uncompressed data. If there are no data in standard buffer, prot_fill function is called. This function, firstly, checks, if any pending data remain in mujMail buffer and try to use it. If not, try to fill fill mm_buff with data. Function prot_mm_fill is responsible for decompressing data from mm_buff into standard buffer. This function works in line oriented manner. One call decompresses only one line. This is needed to be able to change the compression. Each line end can represent end of xmujMail-compression command, if we decompress and return more data, we can not easily ensure that this incorrect data will not be processed by anyone else and we are able to take them back.

Sending data is done in a straightforward way. Before changing compression, we flush data, so buffer is empty and no data are available to standard buffer buf. Compression takes place in the start of prot_flush_encode function. It takes data from standard buffer, and compresses them into mm_buff. RLE compression process byte one by one. GZIP compression is done by library with stream interface, so we write buffer into one socket and read from compressed socket. Mm_buff is then processed by encryption libraries and sent to connection of bigbuffer.