Mediatex

for version 0.9, 7 October 2017

Nicolas Roche (nroche@narval.fr.eu.org)

Mediatex is an Electronic Record Management System (ERMS), focusing on the archival storage entity define by the OAIS draft, and on the NF Z 42-013 requirements.

Abstract

Perennial archives means we provide the resources to overcome the technological obsolescence of supports and drives we use.

Objectives

The perennial archival comes with 3 main objectives: conserve the document, make it accessible and preserve it's understanding.

- Supports reliability: On numeric ages, time life of hard drive and optical supports is 5 years. The numerical archives on hard drives or tapes are sensitive to the earth's magnetic field. Burning WORM supports and store them in good conditions provides a compromise (NF-Z-42-013 standard).
- Identify: Document understanding may be linked to other documents defining a context. Consequently, it is important to provide a way to identify references from documents to others. Grouping documents into collection makes indexes easier and gives the perimeter of data to preserve.
- Other datas: The same way supports need a compatible drive, application data need a software to read them. It is important to memorise which software and format versions are related to the data file, or best, to store software's source and format specifications. In order to do so, we need meta-data.

Geographical duplication

In order to mitigate natural and technological disasters, archives must be duplicated on several sites.

- Distribute meta-data as source code: Idea is servers share meta-data using a revision control system as programmers use it to share their source codes. This is not so obvious: meta-data files must be split into acceptable size so as to be merged in memory, must be human readable as automatic merge may fails and need a human arbitrage, and consume memory to be loaded. Nevertheless, revision control system does not only provides an elegant solution for geographical duplication, but also provide change's history on meta-data. Although using a centralised deposit, this solution offer the advantage to de-synchronise updates, that is a way to anticipate crash recoveries.
- Pull data into a cache: Servers by having up to date (or nevertheless converging) meta-data are able to populate their caches in order to backup unsafe files or to expose files needed from a support. They extract a file from containers (extraction meta-data) or retrieve it from an other server (connection meta-data). Indeed, if a cache is deleted, server will rebuild it from local and remote supports.
- Collaborate for data access: Each server build the same static HTML index (compiling the descriptive meta-data) so as any load-balancer is sufficient to prevent form crash disaster (connection meta-data are shared too). URL on archives do not change as they point to a CGI script that will deal with all servers and return an HTML redirection to content into a server's cache.

Reversibility

We shall consider that the ERMS becomes obsolete itself and we need to change it. A significant effort will be require to return archived content: compress, cut and send data using a

media (support or network) and export the related meta-data. Archiving on supports make sense here as we just have to bring them back. Parsers designed to load the meta-data after updates offer a native software library usable to export them into any format.

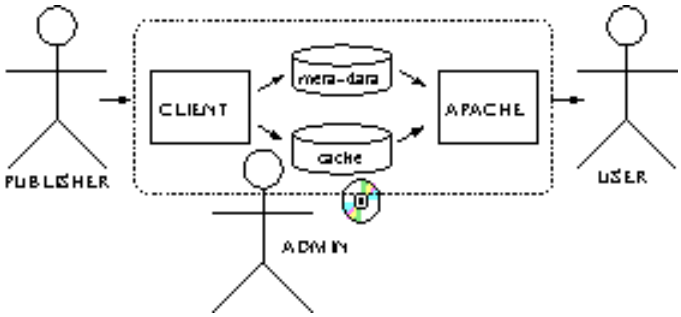
Table of Contents

1 Description.

The MEDIATEX project intends to make easier as possible archives management by spreading and accessing WORM supports (as compact disks) and dedicated meta-data. It is named MEDIATEX because it aims to provide perennial URL on electronic documents.

MEDIATEX is inspired by OAIS archive system description (`http://public.ccsds.org/publications/archive/650x0m2.pdf`) and aims to provide an OAIS's archival storage entity. Its goal is to provide tools that help to manage a NF Z 42-013 compliant supports collection.

1.1 Who



The MEDIATEX system interacts with 3 kind of users:

- ADMIN or `root` system user configures and start services on a given host. He also creates or joins collections.
- PUBLISHERS provide supports and meta-data for a collection: **addUser-delUser** (see Section 5.2.5 [addUser-delUser], page 79).
- USER log-in via an internet browser to browse collection and download archives: **Apache** (see Section 5.1.2 [Apache], page 60).
- DATE schedules the MEDIATEX system's maintenance operations.

Association with system users is detailed next: see Section 3.5 [System users], page 25.

1.2 What

MEDIATEX is mainly designed to provides an API to manage supports (to drive jukeboxes).

As a distributed system, it allows several servers to share data and meta-data related to collections of archives.

MEDIATEX handle the TCP connections between remote servers. It is designed to be installed directly on the archive producer's application servers.

- External supports, are spreads geographically over all servers. Supports was first designed to be optical disks, but may also be a file you manually mount from what ever device and file systems, or any locally hosted files. Because on GNU/Linux there is a bug with last block on CD-ROM, MEDIATEX use the ISO format image to retrieve the support size. So, only the data disks are manage, not the audio ones for instance.
  - Information on support's images (what they contain and which servers provide them) is shared by all servers around the related collections.
  - A single support may be used on several hosts, for instance if we send it or if servers are located in the same place. However, it is not recommended as MEDIATEX will badly deduce that the support is duplicated.
  - A support may be share locally around several collections.
- Servers are connected together via the internet public network and via NAT gateway to reach hosts located on private networks.
  - A server may participate to several collections.
  - A server manages a local set of private supports including copies.
  - Each MEDIATEX system instance is located on a given host. On a so call server, the PUBLISHER (see Section 1.1 [Who], page 3) decides which support are shared by which collection. This association remains private and is not accessible remotely.
- Collections groups archives contained on supports and make them available around the hosts that share the collection.
  - A collection share support's contents using there related hashes and sizes (deduced from the ISO image for CDROM). Collection also share extraction rules and content's description using a catalogue index.
  - A collection should be shared by several servers, at less 2 to prevent from a site crash. All collection's access parameters are shared around all the hosts belonging to the collection.

Moreover, the collection concept is used to group document as MEDIATEX does not handle dedicated life cycle on archives.

What it is not.

MEDIATEX focus first on redundancy, done by software. To do so:

- it isn't using SGBD to handle meta-data but text files.
- it doesn't do mirroring replication (rsync), but share caches locally and access them remotely (read only).
- is not a PHP application but it build a static WEB site and provides a CGI application based on a C library. (see Section 6.6 [Portail], page 126)
- is not based on a replicated SGBD but on a GIT repository.
- it doesn't provide dynamic nor plain-text search. To do it you first have to export the meta-data into a database using the C library. (see Section 6.7 [Linking], page 128)
- it doesn't handle removal data from supports, which implies to re-burn CDs. It also does not provide a top API to remove meta-data.

- it isn't SEDA or any XML exchange protocol compliant (`http://www.archivesdefrance.culture.gouv.fr/seda/index.html`). It can ever be interfaced with an upper layer server that will manage XML transactions and translations of meta-data. (see Section 6.7 [Linking], page 128)

- it doesn't manage time-stamps. (`https://www.openssl.org/docs/manmaster/apps/ts.html`). Time-stamps or whatever additional meta-data files may still be provided and stored as data files.

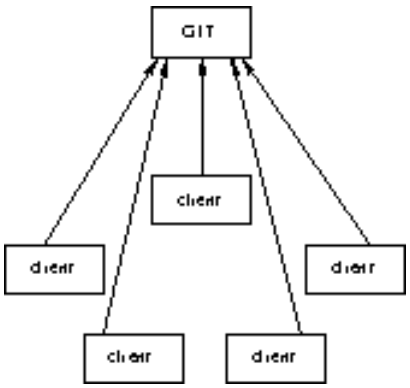
- it doesn't manage claim-based authentication or any "single sign on" mechanism (`https://en.wikipedia.org/wiki/OpenID`). However, APACHE2's `.htaccess` file directives may still be added to the related meta-data file to do so (example provided as sub project).

Time-stamps and claim-based authentication servers which need a static certificate and a static domain name are monolithic servers and so, resides out of this scope. MEDIATEX system is designed to be crashed anywhere but still alive elsewhere.

1.3 Where

- Information about supports (support name, status, number of copies) is locally hold by each server keeping it private.

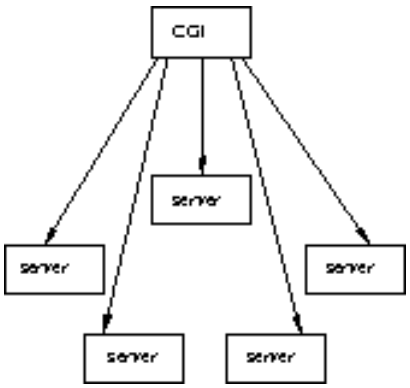
- Collection's meta-data files are duplicated on all servers, but revisions are managed by a uniq GIT "master" server.



- Archive's data files are stored in a cache for each collection by each server.

- A USER (see Section 1.1 [Who], page 3) query is distributed to all servers. If none of

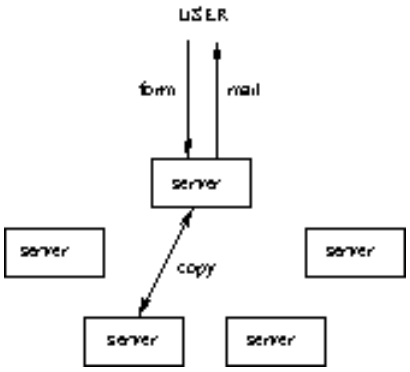
them can fulfil the query, the server receiving the initial query asks for the USER's mail address and registers it. The USER mail address is never sent to the other servers.



- As the catalog provide static HTML content, a DNS's round robbin is usable to switch between server's WEB interface in order to recover from a site crash.

1.4 When

- When PUBLISHER (see Section 1.1 [Who], page 3) modifies the meta-data on a given server, the other servers update there own copies later using the centralised GIT server.
- When USER (see Section 1.1 [Who], page 3) queries for a file that no server provides in their own cache, the first requested server provides an HTML form in order to recall the USER latter, when the file becomes available.



- DATE (see Section 1.1 [Who], page 3) is implement by the **Cron** (see Section 5.1.6 [Cron], page 68) daemon.
- By reading the "message of the day", the PUBLISHER is asked to do some maintenance operations such as:
  - to provide external supports needed for extraction or needed to perform a periodic check.
  - to backup files available from the cache on new supports and to share them with the related collection.

1.5 How

MEDIATEX system is built around other GNU TOOLS (see Section 5.1 [Tools], page 56):

- Trace-ability is done by SYSLOG for programs and by GIT for changes in the meta-data.
- Access is granted by APACHE, but when archives need to be retrieved, the USER (see Section 1.1 [Who], page 3) will be recall later using the default system mailer, that should probably use the SENDMAIL interface.
- Catalogue index and extraction rules have to be provided or edited by hand by the PUBLISHER (see Section 1.1 [Who], page 3) using a text editor.
- Servers tasks are driven by CRON.
- PUBLISHER is asked to build and provide local supports via the host's MOTD (by default).
- The support index and servers list files are internally managed by the MEDIATEX (see Chapter 5 [Conceptual model], page 53) system.

Involved licences:

- Copy/past codes

name	**licence**	**website**
getcgivar	MIT/NCSA?	from NCSA server exemples
e2fsck	GPLv2*	http://e2fsprogs.sourceforge.net/

(*) Author agree to re-license e2fsck's progress bar code under the LGPLv2

- Compilation

name	**licence**	**website**
automake	GPLv2+	http://www.gnu.org/software/automake/
bison	GPLv3+	http://www.gnu.org/software/bison/
gettext	GPLv3+	http://www.gnu.org/software/gettext/
flex	BSD	http://flex.sourceforge.net/
help2man	GPLv3+	http://www.gnu.org/software/help2man/
libavl	LGPLv2+	http://adtinfo.org/
libtool	GPLv2+	http://www.gnu.org/software/libtool/
libssl-dev	BSD	http://www.openssl.org/
make	GPLv3+	http://www.gnu.org/software/make/
texinfo	GPLv3+	http://www.gnu.org/software/texinfo/
transfig	MIT	http://www.xfig.org/ ?

- Documentation (others format than texinfo and man)

name	**licence**	**website**
imagemagick	Apache2	http://www.imagemagick.org/
texlive	LPPL	https://www.tug.org/texlive/

- Installation

name	licence	website
acl	GPLv2+	http://savannah.nongnu.org/projects/acl/
apache2	Apache2	http://httpd.apache.org/
bc	GPLv3+	http://www.gnu.org/software/bc/
bzip2	GPLv3?	http://www.bzip.org/
cpio	GPLv3+	http://www.gnu.org/software/cpio/
findutils	GPLv3+	http://www.gnu.org/software/findutils/
git	GPLv2+	https://git-scm.com/
gzip	GPLv2+	http://www.gnu.org/software/gzip/
ssh	BSD	http://www.openssh.com/
tar	GPLv3+	http://www.gnu.org/software/tar/
unzip	BSD	http://info-zip.org/
viewvc	BSD	http://www.viewvc.org/

- Optionnal: (and non-free)

name	licence	website
rar	EULA	http://www.rarlab.com/
afio	*	http://members.chello.nl/~k.holtman/afio.html

(*) not a standard OSI/FSF approved free software

1.6 How much

In order to prevent from deny of services, we do not use a centralise database but text files spread on all servers using GIT.

Consequently parsers needs a proportional amount of CPU and memory in regards to these file's sizes (whereas databases without indexe do not). The generated HTML catalogue also requires much more place than a dynamic web site does (moreover, limitation should comes from the number of available inodes on the partition where the HTML catalogue is stored).

All in all, the MEDIATEX system is not designed to handle collections having more than half a million archives (whereas databases easily handle millions). It should handle several such "not so big" collections, but not toot much too.

Following tests are based on the GIT upgrade plus HTML catalogue generation, which is the more consumming query (and which imply parsing most meta-data files). It gives an idea of resources (size on disk, amount of memory and CPU time) involved.

archives	GIT	RAM	HTML	HTML inodes	time
27,550	30M	74M	357M	88,717	1'06
54,950	59M	132M	598M	148,294	1'47
82,398	88M	191M	840M	207,985	3'25
110,006	118M	251M	1,1G	268,029	3'18
137,561	147M	310M	1,3G	328,002	5'21
165,104	177M	371M	1,6G	387,836	5'11
192,771	207M	432M	1,8G	447,971	5'47
220,346	237M	493M	2,1G	507,945	5'18

247,861	267M	553M	2,3G	567,664	
302,912	326M	674M	2,8G	687,278	8'31
330,371	356M	735M	3,0G	746,934	
358,005	386M	796M	3,2G	807,007	10'43
385,425	416M	856M	3,5G	866,551	11'17
412,848	446M	916M	3,7G	926,102	12'29
440,405		977M	3,9G	985,990	
467,899	505M	1038M	4,2G	1,045,755	23'37
495,383	535M	1098M	4,4G	1,105,445	14'33

Notice:

- This benchmark is run on a i686 operating system with an AMD Sempron(tm) Processor 3200+ and 2G of RAM. It make me aware that ADM64 system use double of memory.
- Benchmark is done sequentially in order to test the GIT merging with a constant amount of data added each time. Synchronisation times using GIT depends on the network connection, and the upper benchmark is using the local network interface.
- Time spent looks quiet linear (I was expecting $O(n * log(n))$. GIT push and HTML serialisation last most time, about 40% each. Parsing and serializing metadata last about 10% each.
- I was happily surprized that the logarithmic factor is not accented by the fact the files are serialised and parsed (into b-trees) using the same order.
- The audit report process is not optimised. It is possible to handle up to 1,000,000 archives or 500M of metadata by collection, but this particulary process has not been tested yet with more than 50,000 archives (as it last hours).

1.7 Why

Objective is to conserve, offer access and preserve intelligibility of electronic archives.

- Conception was made in order to provides redundancy, trace-ability and a shared HTML access.
- Mediatex make possible to rebuild the system mainly from its external supports (**Notice:** according that up to date catalogue and extraction meta-data are available on supports too). This feature avoid too much entropy and ease migration/restitution.
- It aims to provide a strong "storage" layer on which we should build a whole OAIS system. In that sense, it should provide a positive context for preservation operations, in prevision of time when the actual storage technologies will become outdated.
- It tries to provide a dedicated workaround for the CAP theorem, using GIT merges. (The also known as Brewer's theorem states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees: Consistency, Availability and Partition tolerance.)
- The just uploaded data are not safe: RPO (Recovery Point Objective) is roughly either 1 month (on supports) or 1 day (on caches). But next, for managed data, the RTO (Recovery Time Objective) should be null: DRP (Disaster Recovery Plan) simply consists on adding new servers to the collection (main configuration is already handle

by the collection) and to include the new public IP addresses into the actual DNS round-Robbin's pool.

Why not use MEDIATEX:

- Not designed to handle millions of archives.
- Is a prototype and still need to be optimised.
- Must not occult the forgetting right. As data are burned on optical disks, it is complicated to erase only a part of them.

2 Quick overview

The goal of this section is to help to install MEDIATEX software and to test it by setting an "hello world" collection example.

Notice: the scenario must be run sequentially (6 depends on 5...).

2.1 Install

Get the sources:

```
$ git clone git://git.savannah.nongnu.org/mediatex.git
```

Software requirements.

- Compilation: acl automake autopoint bison flex gettext help2man libacl1-dev libavl-dev libtool libssl-dev make texinfo transfig
- Documentation in others format than texinfo and man (optional): imagemagick texlive
- Installation: apache2 acl bc bsd-mailx bzip2 cgit cpio findutils git gzip libavl1 libacl1 logrotate rsyslog ssh tar unzip
- Debian package generation: libfile-fcntllock-perl

Here are the steps used to build the project from scratch:

```
$ autoscan
$ autoreconf -i
$ ./configure
$ make V=0
$ make check V=0
$ make distcheck V=0
```

However this have to be adapt to be really functional for installation or to enable debuging: (see `/usr/share/perl5/Debian/Debhelper/Buildsystem/autoconf.pm` from debhelper)

```
$ ./configure \
  --prefix=/usr \
  --includedir=/usr/include \
  --mandir=/usr/share/man \
  --infodir=/usr/share/info \
  --sysconfdir=/etc \
  --localstatedir=/var \
  --libexecdir=/usr/lib/mediatex
$ make CFLAGS="-Wall -g -D_FORTIFY_SOURCE=2 -O2" V=0
# make install
# mediatex adm init
```

Other formats than GNU INFO of this documentation are also available from the sources (optional):

```
$ cd doc
$ make html pdf
# make install-html install-pdf
```

Someone may also prefers to build and install a DEBIAN package:

```
$ make dist
$ tar -zxf mediatex-0.8.tar.gz
$ mv mediatex-0.8.tar.gz mediatex_0.8.orig.tar.gz
$ cd mediatex-0.8
$ cp -fr ../debian .
$ dpkg-buildpackage -us -uc
$ ls ..
```

The fully DEBIAN build process should finish using `pbuilder`:

```
$ cd mediatex-0.8
$ cp -fr ../debian .
# pbuilder create --distribution stretch \
  --basetgz ../stretch.tgz \
  --mirror http://ftp.fr.debian.org/debian
# pdebuild --use-pdebuild-internal \
  --debbuildopts -sa -- \
  --basetgz ../stretch.tgz
```

Next, a test script automate the 6 following scenarios:

```
# /usr/share/mediatex/examples/tests.sh
```

2.2 Scenario 1: manage supports

First of all, initialise the default `mdtx` server (**init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77)). **Notice:** this is already done by the DEBIAN package.

```
# mediatex adm init
```

You now have local access to the local meta-data history using this url:
`http://localhost/~mdtx`

Add a new PUBLISHER (see Section 1.1 [Who], page 3) user using an already existing user login (`username1`). Then, reload the Xsession or use `su` to re-login in order the PUBLISHER belongs effectively to his new `mdtx` group.

```
$ id
uid=1000(username1)...
# mediatex adm add user username1
$ su username1
$ id
...groupes=...,122(mdtx),
```

When initialising, an out-dated `mediatex`'s example support was added.

```
$ mediatex ls supp
```

As PUBLISHER, please check the "message of the day" a nd provide this wanted support.

```
$ mediatex motd
Please provide theses local supports:
...
$ mediatex check supp ex-cd1 on /usr/share/mediatex/misc/logoP1.iso
```

MOTD will not complain any more about this support as it was seen recently.

Add a new support.

```
$ mediatex add supp ex-cd2 on /usr/share/mediatex/misc/logoP2.iso
```

Notice: As it works the same way, we are using here ISO files instead of real supports (CD, USB key, ... from such `/dev/sd[bc...]` peripheral files).

Add a new File.

```
$ mediatex add file /usr/share/mediatex/misc/logoP1.iso
```

Moreover, your final archive may simply be a file on your filesystem. In this case MEDIATEX can automatically access it without asking for, using the message of the day.

Notice: There is a 10 chars string you can use to put what you want in order to record a state for each support.

```
$ mediatex note supp ex-cd2 as what_you_want
$ mediatex ls supp
```

2.3 Scenario 2: manage collections

Collections are managing shared support's contents, exposing them to the rest of the world.

Add the new "hello" collection. You will be asked for the "mdtx" password that will gives you full access next, on the HTML catalogue.

```
# mediatex adm add coll hello
```

Set the server's hostname on `/etc/mediatex/mdtx.conf` so as remote servers from latter scenario 5 and 6 may acces to the collection too.

```
<<<
host    localhost
---
host    hostname1
>>>
```

Build the "hello" Html catalogue.

```
$ mediatex make coll hello
```

Add a new USER (see Section 1.1 [Who], page 3) to the collection's APACHE users.

```
$ htdigest /etc/mediatex/mdtx-hello/apache2/htpasswd \
           mdtx-hello username2
```

Give him permissions on some of the HTML sections by editing the `/etc/mediatex/mdtx-hello/apache2/htgroup` file:

```
index: mdtx usename2
cache: mdtx usename2
score: mdtx
history: mdtx
upload: mdtx
```

(Sections are described here: see Section 5.1.2 [Apache], page 60).

Browses the remotely available collection's catalog via this URL:
https://*hostname1*/~mdtx-hello

Notice: for remote access, ensure *hostname1* is known, by adding it to the `/etc/hosts` file of clients.

Change the logo:

```
$ cp yourLogo.{png|jpg|gif} /etc/mediatex/mdtx-hello/logo
$ mediatex make coll hello
```

2.4 Scenario 3: extract an archive

First share supports with the collection

```
$ mediatex add supp ex-cd1 to coll hello
$ mediatex add supp ex-cd2 to coll hello
```

As a USER, browse the collection via this URL: https://HOSTNAME1/~mdtx-hello and ask for a content (by clicking on the floppy disk picture).

As the cache is empty, you will be prompt to give your email address. Please do it. If no mail sender is yet configured, you can use the "*usename1*" local mail address.

As PUBLISHER, please check the "message of the day" and provide wanted supports.

```
$ mediatex motd
$ mediatex check supp ex-cd1 on /usr/share/mediatex/misc/logoP1.iso
$ mediatex check supp ex-cd2 on /usr/share/mediatex/misc/logoP2.iso
```

If your host is configurated to send emails, you should have received an email telling the previous link you try is now be available. (If no mailer is configured you can use `$ mail`).

2.5 Scenario 4: adding new archives

Add a new container by providing extraction rules:

```
# apt-get install cowsay
$ mkdir cow1
$ cowsay mheu1 > cow1/cow1.txt
$ tar -zcf cow1.tgz cow1/
$ MD5_1=$(md5sum cow1/cow1.txt | cut -d' ' -f 1)
$ SIZE_1=$(ls -l cow1/cow1.txt | cut -d' ' -f 5)
$ MD5_2=$(md5sum cow1.tgz | cut -d' ' -f 1)
$ SIZE_2=$(ls -l cow1.tgz | cut -d' ' -f 5)

$ cat > cow1.cat <<EOF
Document "cow1": "animal"
  $MD5_1:$SIZE_1
EOF

$ cat > cow1.ext <<EOF
(TGZ
  $MD5_2:$SIZE_2
=>
```

```
  $MD5_1:$SIZE_1 cow1/cow1.txt
)
EOF

$ mediatex upload file cow1.tgz catalog cow1.cat rules cow1.ext \
   to coll hello
```

Add a file with no metadata:

```
$ cowsay mheu2 > cow2.txt
$ mediatex upload file cow2.txt to coll hello
```

Such incoming files still are not safe until a PUBLISHER get the file form the cache, store them somewhere and provide them from some new supports to the collection.

```
$ mediatex motd
```

Supports may be provided either as external devices :

```
$ mediatex add supp ex-cow1 on cow1.tgz
$ mediatex add supp ex-cow1 to coll hello
```

or either as part of the filesystem :

```
$ mediatex add file cow2.txt
$ mediatex add supp $PWD/cow2.txt to coll hello
```

Now rebuild the html catalogue, and you should have access to the new archive as seen previously (Section 2.4 [Scenario 3], page 14).

```
$ mediatex make coll hello
```

Notice: If you accept `www-data` user as a PUBLISHER (see Section 1.1 [Who], page 3), the HTTP catalog provides an upload form (see Section 6.4 [HTTP], page 124) accessible from the "cache" section.

2.6 Scenario 5: add a second server

On another (second) host, initialise the server. As we seen before (Section 2.2 [Scenario 1], page 12), *username1* is an already existing user login (on this server).

```
hostname2# mediatex adm init
hostname2# mediatex adm add user username1
hostname2$ su username1
```

Change the hostname in `/etc/mediatex/mdtx.conf` configuration file

```
<<<
host    localhost
---
host    hostname2
>>>
```

Try to join the "hello" collection. The host fingerprint should match with the one given in *hostname1* configuration file.

```
hostname2# mediatex adm add coll hello@hostname1
```

This has created your local collection key. Send it to a PUBLISHER (see Section 1.1 [Who], page 3) already managing this collection.

```
hostname2# cat ~mdtx-hello/.ssh/id_dsa.pub
```

PUBLISHER has to add your key to the collection.

```
hostname1$ mediatex add key KEYFILE to coll hello
```

Finally, retry to join the collection.

```
hostname2# mediatex adm add coll hello@hostname1
hostname2$ su username1
```

Now the second server should be connected. We will check it.

First, empty the *hostname1* cache and ask the daemon to scan it.

```
hostname1$ rm -fr ~mdtx/cache/mdtx-hello/logo*
hostname1$ rm -fr ~mdtx/cache/mdtx-hello/supports
hostname1$ mediatex srv scan
```

Secondly, remove support on *hostname1* and share the `ex-cd1` support with the collection on *hostname2*.

```
hostname1$ mediatex del supp ex-cd1
hostname1$ mediatex del supp ex-cd2
hostname2$ mediatex add supp ex-cd1 to coll hello
```

As a USER, browse the collection on the first server via this URL: `https://hostname1/~mdtx-hello` and ask once again for the logo image's archive, providing your email address.

When first server reload (scheduled by CRON), it will notify the second server.

```
hostname1$ mediatex srv notify
```

PUBLISHER on *hostname2* is now asked to provide the supports it shares

```
hostname2$ mediatex motd
hostname2$ mediatex check supp ex-cd1 on \
                          /usr/share/mediatex/misc/logoP1.iso
```

Once second server notify, first server will copy back the provided file it will needs, to serve the USER query.

```
hostname2$ mediatex srv notify
hostname1$ mediatex srv extract
hostname1$ ls ~mdtx/cache/mdtx-hello/logoP1.cat
```

Notice: as for every supports having a bad score, the uploaded files (Section 2.5 [Scenario 4], page 14) are duplicated too.

```
hostname2$ mediatex srv extract
hostname2$ ls ~mdtx/cache/mdtx-hello/cow*
```

2.7 Scenario 6: add a NAT server

Notice: By NAT we mean this third server should be located on a private network. It access the internet via the second server as gateway, but cannot be accessed from outside of his private network.

Network configuration example:

```
 hostname2$ cat /proc/net/dev
 eth0 ... (internet)
 eth1 ... (private network)
 hostname2# echo 1 > /proc/sys/net/ipv4/ip_forward
 hostname2# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 \
                           -j MASQUERADE
  hostname3# ping gnu.org (ok)
hostname1# ping HOSTNAME3 (ko: Destination Host Unreachable)
```

Join collection with the third server as we seen previously (Section 2.6 [Scenario 5], page 15).

Edit the configuration on both second and third server:

```
 hostname2$ cat /etc/mediatex/mdtx.conf
...
networks   private1, www
gateways   private1
...
  hostname3$ cat /etc/mediatex/mdtx.conf
...
networks   private1
...
 hostname2$ mediatex upgrade
  hostname3$ mediatex upgrade
```

With this settings, the second server will relay queries and contents between the first and the third server: (see Section 5.4.6 [notify], page 119)

- messages from *hostname1* are forwarded to *hostname3*.
- *hostname3* do not send messages to *hostname1* as its contents are retrieved back by *hostname2*.

Checking it:

First, share the `ex-cd2` support with the collection on *hostname3*.

```
  hostname3$ mediatex add supp ex-cd2 on \
                           /usr/share/mediatex/misc/logoP2.iso
  hostname3$ mediatex add supp ex-cd2 to coll hello
```

Once first server have reloaded, it will notify queries to the second server (via CRON) which will relay them to the third one.

```
hostname1$ mediatex upgrade
hostname1$ mediatex srv notify
```

PUBLISHER on *hostname3* is now asked to provide the supports it shares.

```
  hostname3$ mediatex motd
  hostname3$ mediatex check supp ex-cd2 on \
                           /usr/share/mediatex/misc/logoP2.iso
```

Once third server notify, second server will copy (and extract) the files needed by server 1.

```
 hostname3$ mediatex srv notify
hostname2$ mediatex srv extract
```

```
hostname2$ ls ~mdtx/cache/mdtx-hello/logo*
```

Finally second server notify and then first server copies the requested file and delivers a mail to advice the User it becomes available.

```
hostname2$ mediatex srv notify
hostname1$ mediatex srv extract
```

3 General help

MEDIATEX focus on providing an API to ask for, and retrieve archives from where they are stored.

3.1 Commands

MEDIATEX CLIENT (see Section 5.3 [Client], page 87)'s queries:

- Admin queries:

  `# mediatex adm init`
  `# mediatex adm remove`
  `# mediatex adm purge`
  : Initialise, remove and purge MEDIATEX software.

  `# mediatex adm add user` *`user`*
  `# mediatex adm del user` *`user`*
  : Manage PUBLISHER users.

  `# mediatex adm add coll [`*`serv-`*`]`*`coll`*`[@`*`host`*`[:`*`port`*`]]`
  `# mediatex adm del coll` *`coll`*
  : Create/subscribe or destroy/unsubscribe a collection.

- Data management:

  `$ mediatex add supp` *`supp`* `on` *`path`*
  : Add a new external support: a device or file not always accessible.

  `$ mediatex add file` *`path`*
  : Add a new external support's file: a file always accessible locally.

  `$ mediatex del supp` *`supp`*
  : Remove a support (external support or support's file).

  `$ mediatex ls supp`
  : List supports.

  `$ mediatex note supp` *`supp`* `as` *`text`*
  : Associate a short text (10 character's) with a support, which will be display on the list.

  `$ mediatex check supp` *`supp`* `on` *`file`*
  : Provide an external support (that is accessible now).

  `$ mediatex add supp` *`supp`* `to (all|coll` *`coll`*`)`
  `$ mediatex del supp` *`supp`* `from (all|coll` *`coll`*`)`
  : Manage sharing support with collections.

  `$ mediatex upload[+] [file` *`file`* `[as` *`target`*`]]* [catalog` *`file`*`] [rules` *`file`*`] to coll` *`coll`*
  : Upload an incoming archive into the cache. Appending one or two '+' will run "uprade" or "upgrade+make" queries too.

- Meta-data management:

`$ mediatex ls [master] coll`
: List collections.

`(motd)` : Display actions PUBLISHER have to perform.

`$ mediatex add key` *`file`* `to coll` *`coll`*
`$ mediatex del key` *`fingerprint`* `from coll` *`coll`*
: Manage remote servers subscription.

`$ mediatex upgrade[+] [coll` *`coll`*`]`
: Synchronise local server. Appending one '+' will run "make" queries too.

`$ mediatex make [coll` *`coll`*`]`
`$ mediatex clean [coll` *`coll`*`]`
: Manage the local HTML catalogue.

`$ mediatex su [coll` *`coll`*`]`
: Change to MEDIATEX (see Chapter 5 [Conceptual model], page 53) or collection system user.

`$ mediatex audit for` *`mail`* `coll` *`coll`*
: Extract all archives from a collection.

- Queries to daemon:

`$ mediatex srv save`
: Ask SERVER (see Section 5.4 [Server], page 107) to dump its state into disk.

`$ mediatex srv extract`
: Ask SERVER to perform extractions.

`$ mediatex srv notify`
: Ask SERVER to communicate its state to other servers.

`$ mediatex srv [quick] scan`
: Ask SERVER to manage file added manually to the cache. The files must be describes into meta-data, otherwise they are deleted from the cache. On quick scan, only file's sizes are matche, not checksums.

`$ mediatex srv trim`
: Ask SERVER to remove from the cache all files that can be extracted by using containers present into the cache.

`$ mediatex srv clean`
: Ask SERVER to remove from the cache all files that are safe and can be extracted locally using local supports.

`$ mediatex srv purge`
: Ask SERVER to remove from the cache all files that are safe.

`$ mediatex srv status`
: Ask SERVER to log its memory status and available cache's sizes.

- Internal/debug queries:

`$ mediatex adm update [coll coll]`
`$ mediatex adm commit [coll coll]`
: Manage GIT synchronisation (already managed by "upgrade" query)

`$ mediatex adm make [coll coll]`
: Build the local HTML catalogue without GIT synchronisation.

`$ mediatex adm bind`
`$ mediatex adm unbind`
: Manage collection repository binding on the chrooted jail for SSH remote access (already manage SERVER).

`$ mediatex adm mount iso on path`
`$ mediatex adm umount path`
: Manage mounting ISO devices.

`$ mediatex adm get path as coll on path`
: Retrieve a remote collection's file via SSH.

3.2 Command line Options

Theses options apply to the 3 binaries provided by MEDIATEX.

`-h`
`--help` Display the help message.

`-V`
`--version`
: Display the mediatex software version.

`-f facility`
`--facility facility`
: Set facility to use for logging. See syslog(3) : mainly `file` to log to the standard output or `local2` to log to SYSLOG.

`-l path`
`--log-file path`
: Log into a file. Needs the `file` facility (default).

`-s severity[:module(,module)*]`
`--severity severity[:module(,module)*]`
: Set severity to log, by modules. Choose severity among `err`, `warning`, `notice`, `info` and `debug`. You may provide modules among `alloc`, `script`, `misc`, `memory`, `parser`, `common` and `main`. If you provide no module, all except the `alloc` module will be selected.

`-v`
`--dry-run`
: increase verbosity (wrapper for -s), from `-s "info"`, `-s "debug:main,script"` to `-s "debug:memory,misc"` for `-vvv`

`-n`
`--dry-run`
: Do a dry run: do not write anything.

`-p`
`--no-progbar`
: Do not display the progression bar.

`-m` *number*
`--memory-limit` *number*
: Set a nice limit for malloc in Mo. When SERVER overcome this limit, it will try to release objects not used.

`-L,`
`--debug-lexer`
: Enable debug information from lexers.

`-c` *mdtxLabel*
`--conf-label` *mdtxLabel*
: Over hide 'mdtx.conf' configuration file and use *mdtxLabel* system user instead of `mdtx`. This permit to run independent MEDIATEX systems on the same machine. This is mainly used for tests.

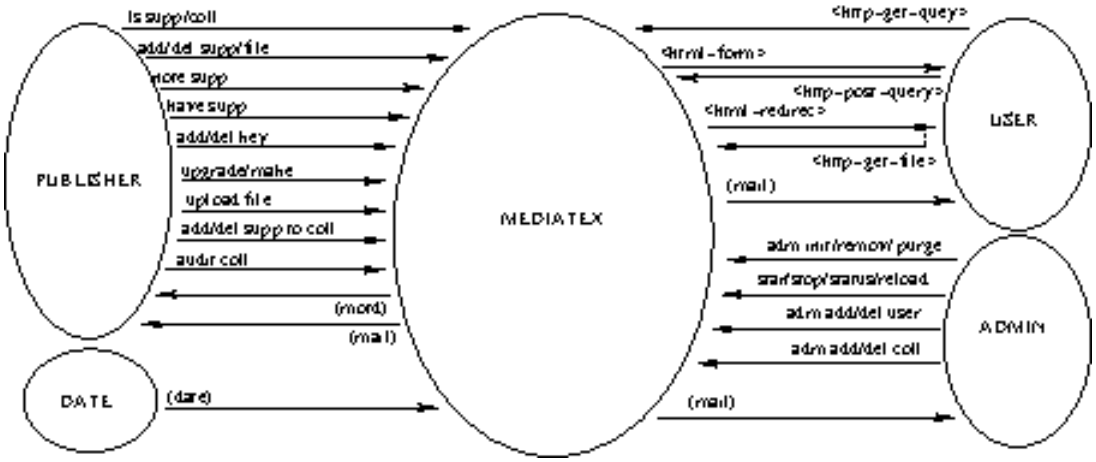
`-a`
`--alone`
: Do not make GIT queries. This may be useful if you temporary loose your network connection, or if you want to be temporally isolated for tests. **Notice:** This option is only available on CLIENT (see Section 5.3 [Client], page 87) as SERVER (see Section 5.4 [Server], page 107) and **cgiClient** (see Section 5.2.2 [cgiClient], page 75) never perform GIT queries.

3.3 Communication model

The following specifications use the MERISE conception method. Here are the typographical conventions used:

- Caps are use for ACTORS and ACTIVITIES,
- Bold is use for **processing**s,
- "<>" are use for object's <*states*>,
- Verbatim is used for `events`, which may be:
  - `# command` : ADMIN or DATE command line
  - `$ command` : PUBLISHER command line
  - `(message)` : final message like mail and motd
  - `<message>` : synchronous message using a socket
  - `[register]` : asynchronous message using signal
  - `function()` : internal call

Communication model:



- 4 actors interact with the MEDIATEX activity
  - USER (see Section 1.1 [Who], page 3)
  - PUBLISHER (see Section 1.1 [Who], page 3)
  - ADMIN (see Section 1.1 [Who], page 3)
  - DATE (see Section 1.1 [Who], page 3)
- Events in

```
<http-get-query>
<http-post-query>
<http-get-file>
```

from USER

```
$ mediatex add key file to coll coll
$ mediatex del key fingerprint from coll coll
$ mediatex upgrade[+] [coll coll]
$ mediatex add supp supp on path
$ mediatex add file path
$ mediatex del supp supp
$ mediatex note supp supp as text
$ mediatex check supp supp on file
$ mediatex ls supp
$ mediatex ls [master] coll
$ mediatex add supp supp to (all|coll coll)
$ mediatex del supp supp from (all|coll coll)
$ mediatex upload[+] [file file [as target]]* [catalog file] [rules file]
to coll coll
$ mediatex make [coll coll]
$ mediatex audit for mail coll coll
```

from PUBLISHER

`# /etc/init.d/mediatexd start`
`# /etc/init.d/mediatexd stop`
`# /etc/init.d/mediatexd status`
`# /etc/init.d/mediatexd reload`
`# mediatex adm init`
`# mediatex adm remove`
`# mediatex adm purge`
`# mediatex adm add user` *`user`*
`# mediatex adm del user` *`user`*
`# mediatex adm add coll [`*`serv-`*`]`*`coll`*`[@`*`host`*`[:`*`port`*`]]`
`# mediatex adm del coll` *`coll`*
: from ADMIN

`(date)` : from DATE

Commands are detailed into the MEDIATEX (see Chapter 5 [Conceptual model], page 53) specifications.

- Events out

`<html-form>`
`<html-redirect>`
`(mail to USER)`
: to USER

`(motd)`
`(mail to PUBLISHER)`
: to PUBLISHER

`(mail to ADMIN)`
: to ADMIN

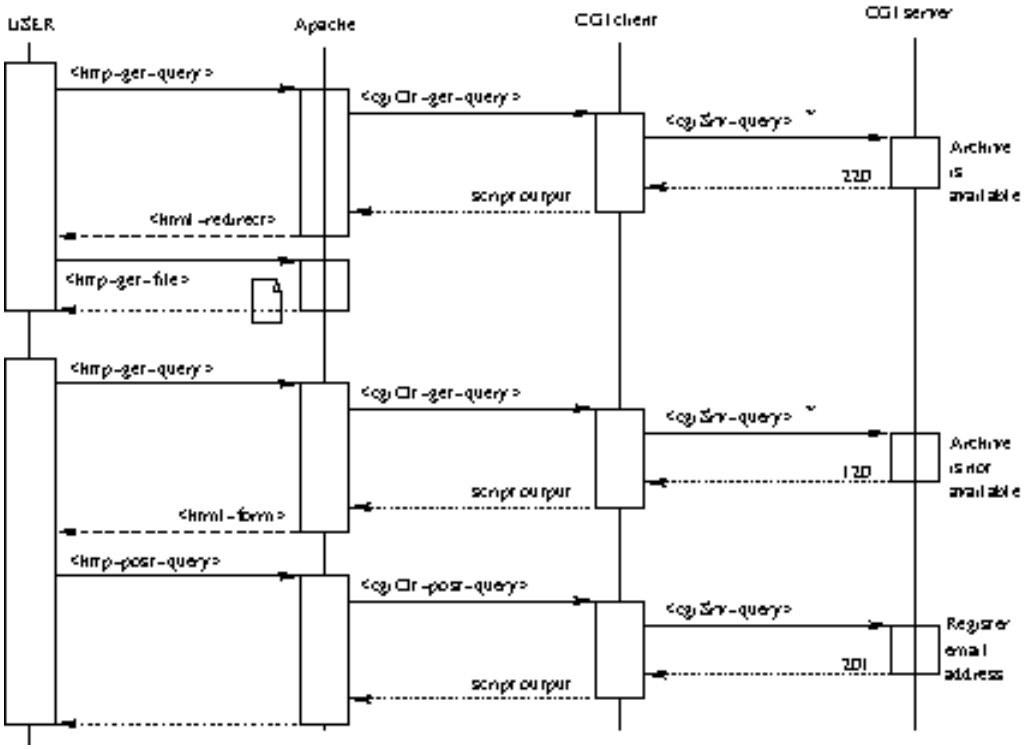
- Data in/out
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)

  Files involved are detailed in the next chapter (see Section 1.1 [Who], page 3).

3.4 Accessing an archive

This diagram describes what happen when a USER (see Section 1.1 [Who], page 3) access an archive:

- available on some SERVER (see Section 5.4 [Server], page 107)'s caches
- not available on any SERVER (see Section 5.4 [Server], page 107)'s cache.



3.5 System users and groups

MEDIATEX use `acl` to share files between system users.

Systems users and groups:

`root` ADMIN (see Section 1.1 [Who], page 3), DATE (see Section 1.1 [Who], page 3)

`www-data` USER (see Section 1.1 [Who], page 3)

`mdtx` SERVER (see Section 5.4 [Server], page 107), CLIENT (see Section 5.3 [Client], page 87) (default)

`mdtx-`*`coll`*
CLIENT (see Section 5.3 [Client], page 87) (collection context)

`username` PUBLISHER (see Section 1.1 [Who], page 3)s

PUBLISHER's system users belongs to the `mdtx` group. **addUser-delUser** (see Section 5.2.5 [addUser-delUser], page 79)

3.6 Meta-data consistency

GIT pull and push operations are done implicitly when loading and serialising files. This may be view like a stack model ensuring consistency on meta-data files (see Chapter 4 [Relational schema], page 28).

The so-called meta-data files are mentioned bellow using the followings numbers:

1. `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
2. `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)
3. `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)

4. `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
5. `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)
6. `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)

All calls to CLIENT follow the same operations:

Stage	**Description**
[A	Scripting, which do not need to load the meta-data
[B	GIT pull and parse configuration meta-data (1, 2)
[C	GIT pull and parse collection meta-data (3, 4, 5, 6)
[D	Generic upgrade works: upgrade SSH and GIT settings and re-compute the local image's score
D]	Working on meta-data
C]	Serialise collection meta-data and GIT push (3, 4, 5, 6)
B]	Serialise configuration meta-data and GIT push (1, 2)
A]	Send `HUP` message to SERVER (see Section 5.4 [Server], page 107)

Notice:

- Stage [D (Generic upgrade works) is only processed when (6) is loaded. This stage insure connection consistency between remote servers :

```
   MASTER          shared           SLAVES
-------------------------------------------
  mdtx.conf <--> servers.txt <--> mdtx.conf
                       |
.ssh/       <--------+----------> .ssh/
.git/config <--------+----------> .git/config
```

- On stage C] (Serialise meta-data files), the default behaviour is to not serialise (4) and (5) unless they provide a new `NN.txt` post-fixed file. This intents to speed-up some queries endings (as `make`). In order to force serialising theses files you have to use the `upgrade` query.

Bellow details how meta-data files are affected by CLIENT's queries.

- ADMIN (see Section 1.1 [Who], page 3) queries:

Query	**[A**	**[B**	**[C**	**[D**	**D]**	**C]**	**B]**	**A]**
# mediatex adm init	y						1,2	y
# mediatex adm remove	y							
# mediatex adm purge	y							
# mediatex adm add user *user*	y							
# mediatex adm del user *user*	y							
# mediatex adm add coll [*serv-*]*coll*[@*host*[:*port*]]	y	1,2	6	y	y	6	1,2	y
# mediatex adm del coll *coll*	y	1			y		1	y

- CLIENT to himself queries (or debugging queries):

Query	**[A**	**[B**	**[C**	**[D**	**D]**	**C]**	**B]**	**A]**

$ mediatex adm update [coll *coll*]		1						
$ mediatex adm commit [coll *coll*]		1						
$ mediatex adm make [coll *coll*]		1,2	4,5,6	y	y			
$ mediatex adm bind	y							
$ mediatex adm unbind	y							
$ mediatex adm mount *iso* on *path*	y							
$ mediatex adm umount *path*	y							
$ mediatex adm get *path* as *coll* on *path*		1	6		y	y		

- Queries to operate on SERVER:

Client query	**[A**	**[B**	**[C**	**[D**	**D]**	**C]**	**B]**	**A]**
$ mediatex srv save								y
$ mediatex srv extract								y
$ mediatex srv notify								y
$ mediatex srv [quick] scan								y
$ mediatex srv trim								y
$ mediatex srv clean								y
$ mediatex srv purge								y
$ mediatex srv status								y

- PUBLISHER data management's queries:

Client query	**[A**	**[B**	**[C**	**[D**	**D]**	**C]**	**B]**	**A]**
$ mediatex add supp *supp* on *path*		1,2			y		2	
$ mediatex add file *path*		1,2			y		2	
$ mediatex del supp *supp*		1,2	6	y	y	6	1,2	y
$ mediatex ls supp		1,2			y			
$ mediatex note supp *supp* as *text*		1,2		y	y		2	
$ mediatex check supp *supp* on *file*		1,2	5		y		2	
$ mediatex add supp *supp* to (all|coll *coll*)		1,2	6	y	y	6	1,2	y
$ mediatex del supp *supp* from (all|coll *coll*)		1,2	6	y	y	6	1,2	y
$ mediatex upload[+] [file *file* [as *target*]]* [catalog *file*] [rules *file*] to coll *coll*		1	5		y	5		y

- PUBLISHER meta-data management's queries:

Client query	**[A**	**[B**	**[C**	**[D**	**D]**	**C]**	**B]**	**A]**
$ mediatex ls [master] coll		1			y			
$ mediatex motd		1,2	3,5,6	y	y	6	2	y
$ mediatex add key *file* to coll *coll*		1	6	y	y	6	2	y
$ mediatex del key *fingerprint* from coll *coll*		1	6	y	y	6	2	y
$ mediatex upgrade[+] [coll *coll*]		1	4,5,6	y		4,5,6	1,2	y
$ mediatex make [coll *coll*]		1	4,5,6	y	y	4,5,6	1,2	y
$ mediatex clean [coll *coll*]	y							
$ mediatex su [coll *coll*]		1			y			
$ mediatex audit for *mail* coll *coll*			5					

4 Data files

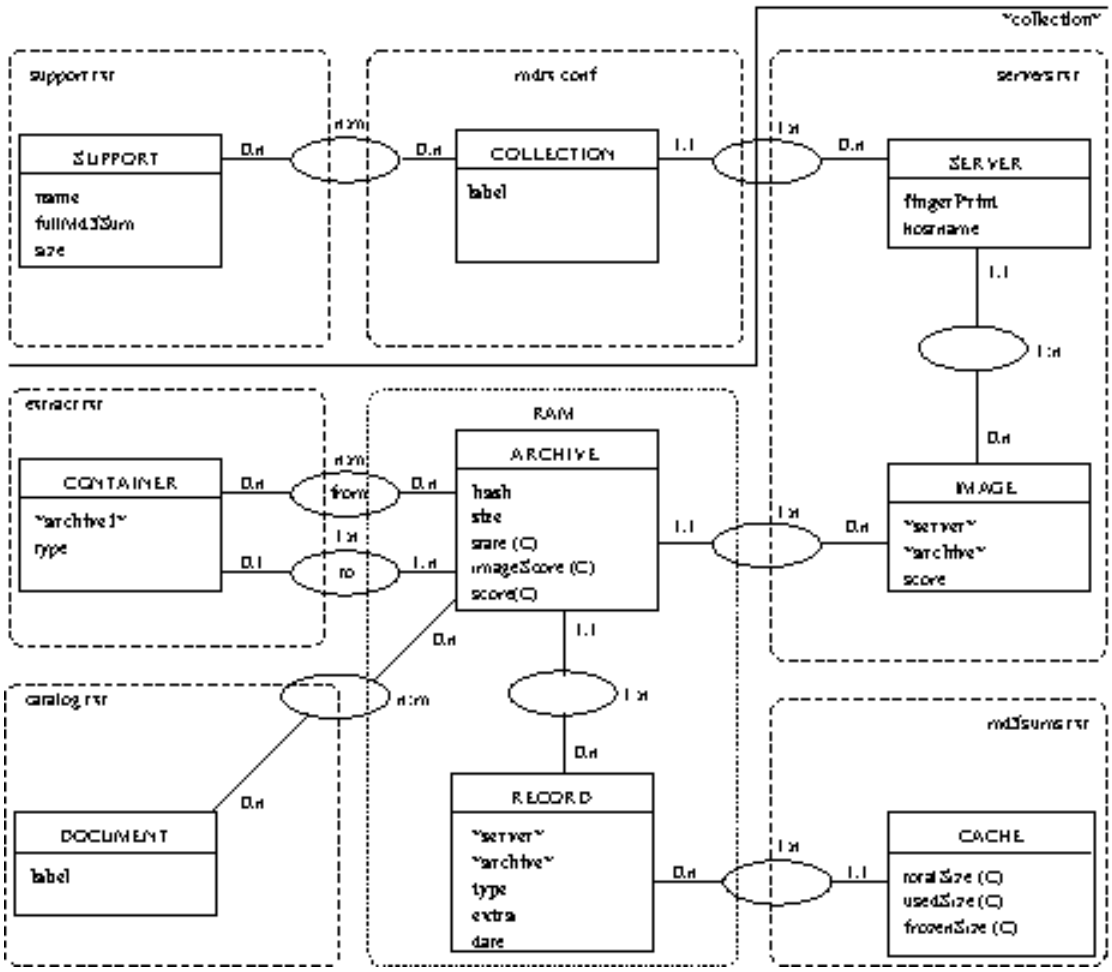
There are 6 main meta-data files share between 3 categories.

- 2 Local server files
- 1 Local collection file
- 3 Remotely shared collection files

All of theses file are locked by the kernel when parsing and serializing using the reader/writer algorithm. This ensure the different processes will not read and modify the same ressource at same time.

The 3 shared collection files are processed to build the same browsable HTML index on each servers (see Section 5.3.4 [misc], page 100).

Relational schema:



- Entities
  - The SUPPORT entity lists all the local supports.
  - The COLLECTION entity lists all the collections the local server belongs to.

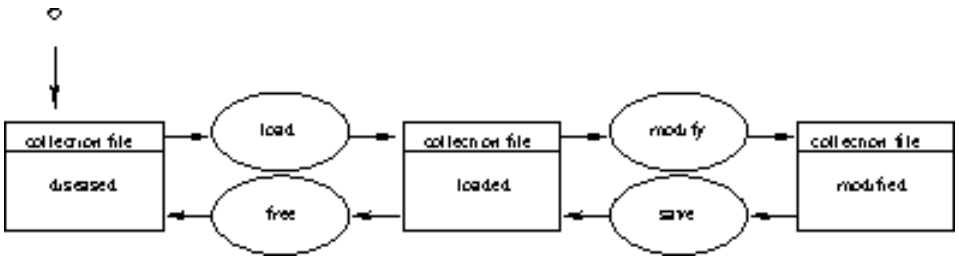
  Below entities are all dupplicated for each collection:

- The SERVER entity lists all the servers.
- The IMAGE entity lists on ISO images hold by a server.
- The ARCHIVE entity lists all archives known at running time.
- The CONTAINTER entity gives rule to extract archives from each others.
- The DOCUMENT entity groups archives in order to describe them in a catalog.
- The RECORD entity lists archives we works with.

- Data in/out

File	PUBLISHER	CLIENT	**cgi**	SERVER
mediatex.conf	rw	rw	r	r
supports.txt	r	rw	-	r
md5sums.txt	r	r	-	rw
catalog.txt	rw	rw	-	-
extract.txt	rw	rw	-	r
servers.txt	rw	rw	r	r

- Life cycle for local files:



- States

  `diseased` data remains on disk (file is up to date)

  `loaded` the file is loaded into memory

  `modified` changes on data only remains in memory

- Events in

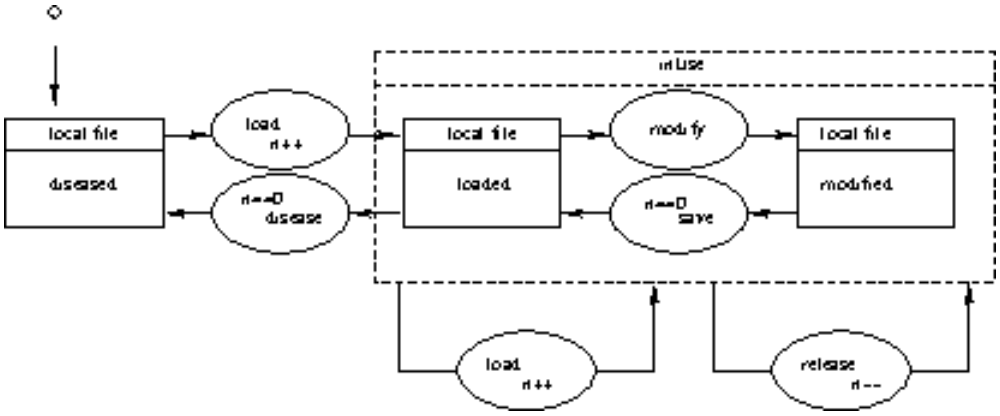
  `load` from <*diseased*> to <*loaded*>:
  a **Process** tells it needs the file to be load

  `modify` from <*loaded*> to <*modified*>:
  a **Process** tells it has modified the data

  `save` from <*modified*> to <*loaded*>:
  an ACTIVITY saves all modifications on disk

  `free` from <*loaded*> to <*diseased*>:
  an ACTIVITY ends or will reload the data

- Life cycle for shared files:



Notice: The <*loaded*> and <*modified*> states are grouped into the <*inUse*> meta-state.

- States

  `diseased` data remains on disk (file is up to date)

  `loaded` the file is loaded into memory

  `modified` changes on data remains in memory

- Events in
  Notice: the number of threads using the data (n) is registered in order to know if we may or not release memory.

  `load (n+=1)`
  from <*diseased*> to <*loaded*> if n==0 or
  from <*inUse*> to <*inUse*> if n>0
  a **Process** tells it needs the file to be load

  `release (n-=1)`
  from <*inUse*> to <*inUse*>:
  a **Process** tells it no more need the file

  `modify` from <*loaded*> to <*modified*>:
  a **Process** tells it has modified the data

  `save` from <*modified*> to <*loaded*> if n==0:
  an ACTIVITY saves all modifications on disk

  `disease` from <*loaded*> to <*diseased*> if n==0:
  an ACTIVITY that needs place on memory unloads the data

Code:

`src/misc/locks.c`
`src/memory/archive.h`
`src/common.openClose.c`

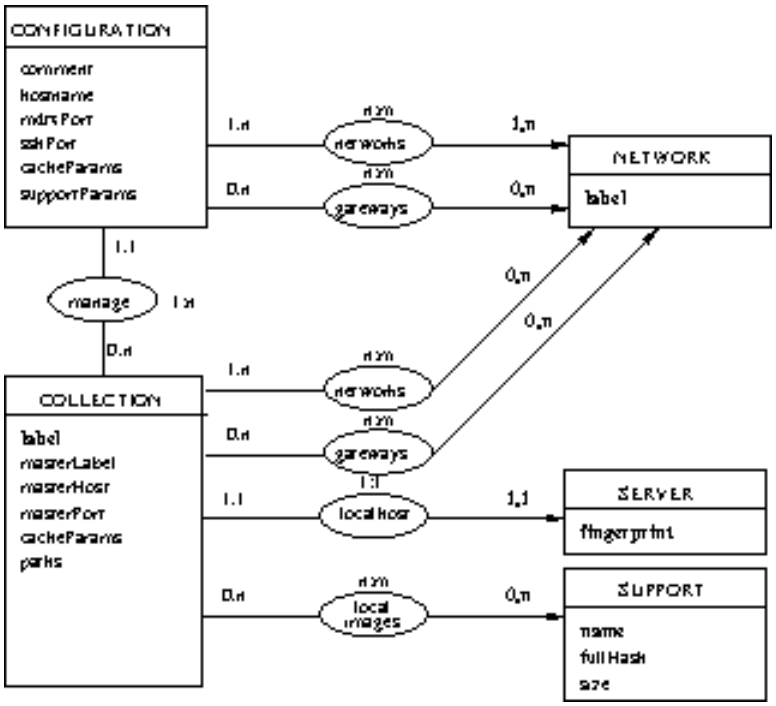
4.1 mdtx.conf

The `/etc/mediatex/mdtx.conf` file is used locally as the main MEDIATEX configuration file. It is read by the 3 binaries:

- CLIENT (see Section 5.3 [Client], page 87)
- **cgiClient** (see Section 5.2.2 [cgiClient], page 75)
- SERVER (see Section 5.4 [Server], page 107)

Both PUBLISHER (see Section 1.1 [Who], page 3) and CLIENT may modify it.

Relational schema:



- Proper entities
  - The CONFIGURATION entity is the single main contexte.
  - The COLLECTION entity lists all the collections shared.
  - The NETWORK entity lists logical networks the local host belongs to.
- Linked entities
  - The SUPPORT entity lists all the local supports.
  - The SERVER entity lists all servers sharing a given collection.
- Proper relations
  - The MANAGE relationship lists all the collections our local server share.
  - The NETWORKS relationship lists all the networks the local host belongs to. This relationship may be overwriten for each collection.
  - The GATEWAYS relationship lists all the networks for which our local server act like a NAT gateway. (see **notify** (see Section 5.4.6 [notify], page 119) and see Section 2.7 [Scenario 6], page 16) This relationship may be overwriten for each collection.

- The LOCALHOST relationship gives our local server related to the collection context.
- The LOCALIMAGES relationship lists all the supports shared within a specific collections.

Notice: `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49) summerize the IMAGEFILES, NETWORKS and GATEWAYS relationships from all the hosts sharing a collection.

Example:

```
# This is the MediaTeX software configuration file.

# comment: greeter for this server
comment    "Put your personal greeting here"

# host: hostname used in urls
host       127.0.0.1

# port: listening port for incoming requests
mdtxPort   6001

# port: listening port for SSHd
sshPort    22

# port: listening ports for Apache
httpPort   80
httpsPort  443

# networks: networks the host belongs to
networks   www

# gateways: networks the host is a gateway for

# default cache parameters
cacheSize  100 Mo
cacheTTL   15 Day
queryTTL   1 Week

# local support parameters
checkTTL   6 Month
fileTTL    2 Month
suppTTL    5 Year
maxScore   10.00
badScore   1.00
powSupp    2.00
factSupp   2.00

# motd policy
motdPolicy most
```

```
# The below section is also managed by MediaTeX software.
# You should not edit by hand parameters bellow the -- line.
# (host fingerprint: 89ff8cc8968fac6bc456757ce563b0b4)

Collection serv1-hello@127.0.0.1:22
# --
localhost 540321ab656ad5c287e76f9dc361cfff
share   /usr/share/mediatex/misc/logoP2.iso,
  iso1
end
```

Grammar:

```
stanzas: stanzas confLine
       | stanzas collectionStanza
       | confLine
       | collectionStanza

collectionStanza: collection collectionLines ENDBLOCK

collection: coll string - string @ string : number

collectionLines: collectionLines collectionLine
               | collectionLine

collectionLine: SHARE supports
              | LOCALHOST string
              | NETWORKS networks
              | GATEWAYS gateways
              | CACHESIZE number SIZE
              | CACHETTL number TIME
              | QUERYTTL number TIME
              | MOTDPOLICY (most|all)

supports: supports , support
        | support

support: string

confLine: HOST string
       | NETWORKS networks
       | GATEWAYS gateways
       | COMMENT string
       | MDTXPORT number
       | SSHPORT number
       | HTTPPORT number
       | HTTPSPORT number
```

```
            | CACHESIZE number SIZE
            | CACHETTL number TIME
            | QUERYTTL number TIME
            | CHECKTTL number TIME
            | FILETTL number TIME
            | SUPPTTL number TIME
            | MAXSCORE score
            | BADSCORE score
            | POWSUPP score
            | FACTSUPP score
            | MOTDPOLICY (most|all)

    networks: networks , network
            | network

    gateways: gateways , gateway
            | gateway

    network: string

    gateway: string

    string: [^[:blank:],\r\n]{1,511} | \"[^\r\n\"]{1,509}\"
    coll:   [^\r\n[:blank:]:\-@]{1,20}
    size:   GO | MO | KO | O
    time:   YEAR | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND
    score:  [[:digit:]]+\.[[:digit:]]+
    number: [[:digit:]]+
```

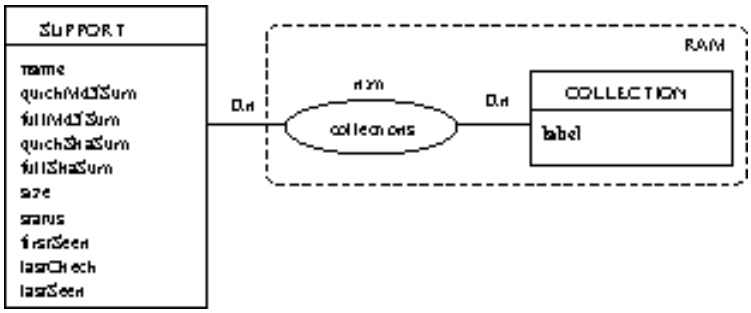
Code:

```
src/memory/confTree.h
src/memory/confTree.c
src/parser/confFile.l
src/parser/confFile.y
```

4.2 supports.txt

CLIENT (see Section 5.3 [Client], page 87) backup the support's status into the `~mdtx/git/mdtx/supports.txt` file. Support's checksum is compute using `md5sum` for internal identification like for all MEDIATEX archives, but is also using `sha1`... who know.

Relational schema:



- Proper entities
  - The SUPPORT entity list all the local supports.

  There is 2 kind of records
  - external support: `support's name`
    PUBLISHER (see Section 1.1 [Who], page 3) provides theses supports manually only when they are requiered by CLIENT (see Section 5.3 [Client], page 87). Else the supports are not accessed by MEDIATEX. External supports are devices (CDROM, USB key ...), or files PUBLISHER know how to access from the local host (backups, S3, ...).
  - external file: `absolute path`
    Theses supports are files supposed to be permantly available on the local host. CLIENT access diretely to them without asking PUBLISHER to provide them.
- Linked entities
  - The COLLECTION entity, from `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31), lists all the collections the server share.
- Relations
  - The COLLECTIONS relationship list all the collection a specific support is shared with. This relation is not recorded into this file but into `/etc/mediatex/mdtx.conf`.

Example: (line is cut to display it here)

```
#         firstSeen           lastCheck            lastSeen...
2010-01-01,01:00:00 2010-01-01,01:00:00 2010-01-01,01:00:00...

...                      quickMd5sum                         fullMd5sum...
...de5008799752552b7963a2670dc5eb18 de5008799752552b7963a2670dc5eb18...

...                               quickShasum...
...6ed0e67ccd882f6e94f931a07834bc6bc6394583...
```

```
...                                fullShasum...
...6ed0e67ccd882f6e94f931a07834bc6bc6394583...

...size       status name
...391168          ok ex-cd1
```

Grammar:

```
file: lines
    | //empty file

lines: lines line
     | line

line: date date date md5 md5 sha sha [[:digit:]]+ status name

date:   {year}-{month}-{day},{hour}:{min}:{sec}

year:   [[:digit:]]{4}
month:  [[:digit:]]{2}
day:    [[:digit:]]{2}
hour:   [[:digit:]]{2}
min:    [[:digit:]]{2}
sec:    [[:digit:]]{2}

md5:    [[:xdigit:]]{32}
sha:    [[:xdigit:]]{40}
status: [^[:blank:]\r\n]{1,10}
name:   [^[:blank:]\r\n]{1,64}
```

Code:

```
src/memory/supportTree.h
src/memory/supportTree.c
src/parser/supportFile.l
src/parser/supportFile.y
```

This function is used to differenciate external supports from external supports's files:

```
int  isSupportFile(Support* self)
{
  int rc = FALSE;

  checkSupport(self);
  rc = (*self->name == '/');

 error:
  return rc;
}
```

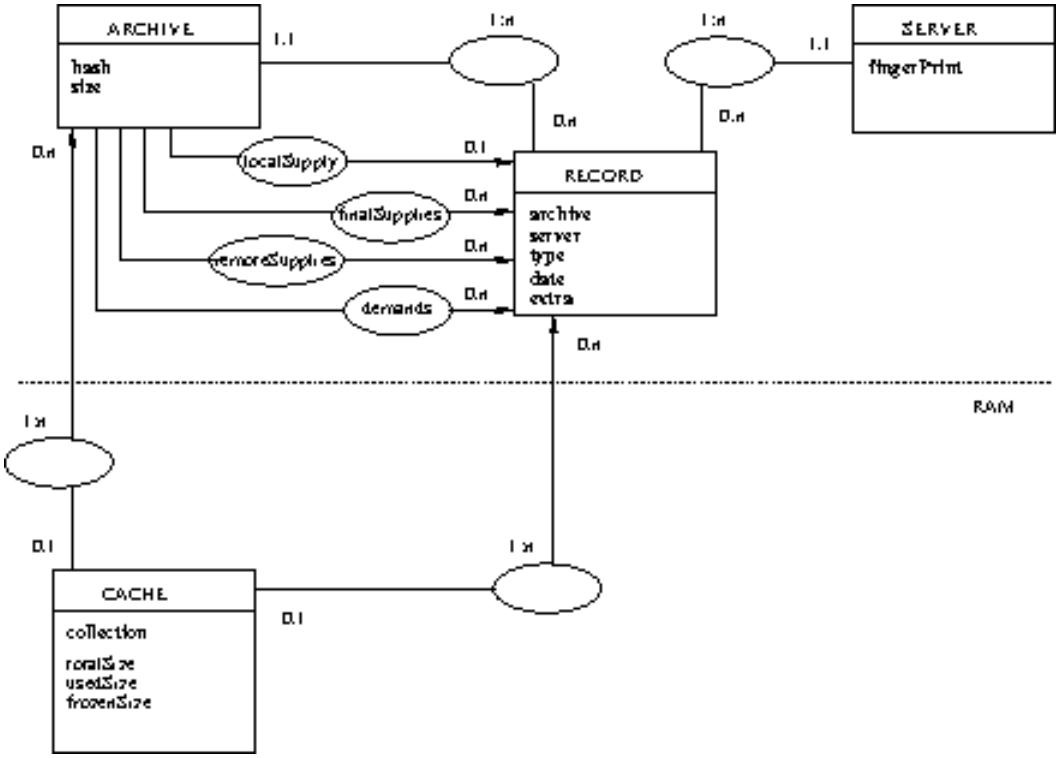
4.3 mdtx-COLL.md5

SERVER (see Section 5.4 [Server], page 107) backup its collection's status into the `~mdtx/md5sums/mdtx-COLL.md5` file. When PUBLISHER or CLIENT want to read the lastest status, they need to ask SERVER the refresh this file.

The Messages the **cgiClient** (see Section 5.2.2 [cgiClient], page 75), **supp** (see Section 5.3.2 [supp], page 94) and **notify** (see Section 5.4.6 [notify], page 119) processes exchange via socket are using the same grammar language. There is type for such messages:

- `DISK`: used to serialize the `~mdtx/md5sums/mdtx-COLL.md5` file
- `NOTIFY`: used by SERVERS to talk each others
- `UPLOAD`: used to add new incoming archives
- `CGI`: used by **cgiClient** (see Section 5.2.2 [cgiClient], page 75) to send USER queries
- `HAVE`: used when PUBLISHER notify that a support is provided

These messages are cyphered using AES algorithm and a key defined into `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49) file.
When loaded into memory, such messages are handled using the following relational schema:



- Proper entities
  - The RECORD entity lists all informations we need to load/save and exchange about archives. **TODO:**MEDIATEX should use an hash table to manage records to speed-up cache management.

  There is 7 types of records

Type	type	host	extra
FINAL_DEMAND	D	local	mail
LOCAL_DEMAND	D	local	!wanted
REMOTE_DEMAND	D	remote	!wanted
FINAL_SUPPLY	S	local	absolute path
MALLOC_SUPPLY	S	local	!malloc
LOCAL_SUPPLY	S	local	relative path
REMOTE_SUPPLY	S	remote	relative path

Notice: for FINAL_SUPPLY the extra field is more accurately *path1*[:*path2*], where:

- *path1* is the absolute path of an external file (archive to upload or available support),
- *path2* is a potential relative path to the place where the file should be copied into the cache.

The MALLOC_SUPPLY is an internal state used when allocating the archive's data file into the SERVER's cache.

- Linked entities
  - The ARCHIVE entity reminds all archives the programm is currently using per collection.
  - The SERVER entity, from /etc/mediatex/mdtx-*COLL*/servers.txt (see Section 4.6 [servers.txt], page 49), lists all the remote servers per collection.
- Relations
  - The LOCALSUPPLY relationhip gives the archive's file when available (LOCAL_DEMAND) on the local cache.
  - The FINALSUPPLIES relationhip lists supports presentely available on local host (FINAL_SUPPLY).
  - The REMOTESUPPLIES relationhip lists all archives available remotely (REMOTE_SUPPLY) from the other SERVER.
  - The DEMANDS relationhip lists all demand (FINAL_DEMAND, LOCAL_DEMAND and REMOTE_DEMAND) for an archive

All the time cache is loaded, archive objects are never free but eventually marked as deleted. The cache api is thread safe. 3 locks are used in order to allow concurrent access:

1. MUTEX_ALLOC: when modifying the cache size
2. MUTEX_COMPUTE: when computing archive state
3. MUTEX_KEEP: when adjusting the archive time to live into the cache
4. MUTEX_TARGET: when creating a new target file

Example:

```
Headers
  Collection hello
  Type       DISK
  Server     adc2f7b78c43354df5a86efae3dfe562
  DoCypher   FALSE
```

```
Body
#                date    host    hash   size extra
D 2015-10-04,23:40:13 adc2... 022a...  24075 test@test.org
S 2015-10-04,23:40:28 adc2... 1a16...  20480 logoP1.cat
S 2015-10-04,23:40:24 aed9... 1a16...  20480 logoP1.cat
S 2015-10-04,23:34:30 adc2... 99b8...   1004 2015-10/mediatex.css
```

Grammar:

```
file: //empty file
    | header
    | header lines

header: HEADERS hLines BODY

hLines: hLines hLine
     | hLine

hLine: COLLECTION string
     | SERVER string
     | TYPE msgval
     | DOCYPHER bool

lines: lines newLine
     | newLine

newLine: line

line: type date hash hash [[:digit:]]+ string

msgval: DISK | CGI | HAVE | NOTIFY | UPLOAD
bool: FALSE | TRUE
type: S | D

date:   {year}-{month}-{day},{HOUR}:{min}:{sec}

year:   [[:digit:]]{4}
month:  [[:digit:]]{2}
day:    [[:digit:]]{2}
HOUR:   [[:digit:]]{2}
min:    [[:digit:]]{2}
sec:    [[:digit:]]{2}

hash:   [[:xdigit:]]{32}
string: [^[:blank:]\r\n]{1,511}
```

Code:

```
src/misc/cypher.c
src/memory/archiveTree.h
src/memory/archiveTree.c
src/memory/recordTree.h
src/memory/recordTree.c
src/parser/recordFile.l
src/parser/recordFile.y
src/common/openClose.c
```

This code is use to get the record's type:

```
RecordType rc = UNDEF_RECORD;

switch (self->type & 0x3) {
case DEMAND:
  if (!self->server->isLocalhost)
    rc = REMOTE_DEMAND;
  else if (self->extra[0] != '!')
    rc = FINAL_DEMAND;
  else if (self->extra[1] == 'w')
    rc = LOCAL_DEMAND;
  break;

case SUPPLY:
  if (!self->server->isLocalhost)
    rc = REMOTE_SUPPLY;
  else if (self->extra[0] == '/')
    rc = FINAL_SUPPLY;
  else if (self->extra[0] != '!')
    rc = LOCAL_SUPPLY;
  else if (self->extra[1] == 'm')
    rc = MALLOC_SUPPLY;
  break;
}
```

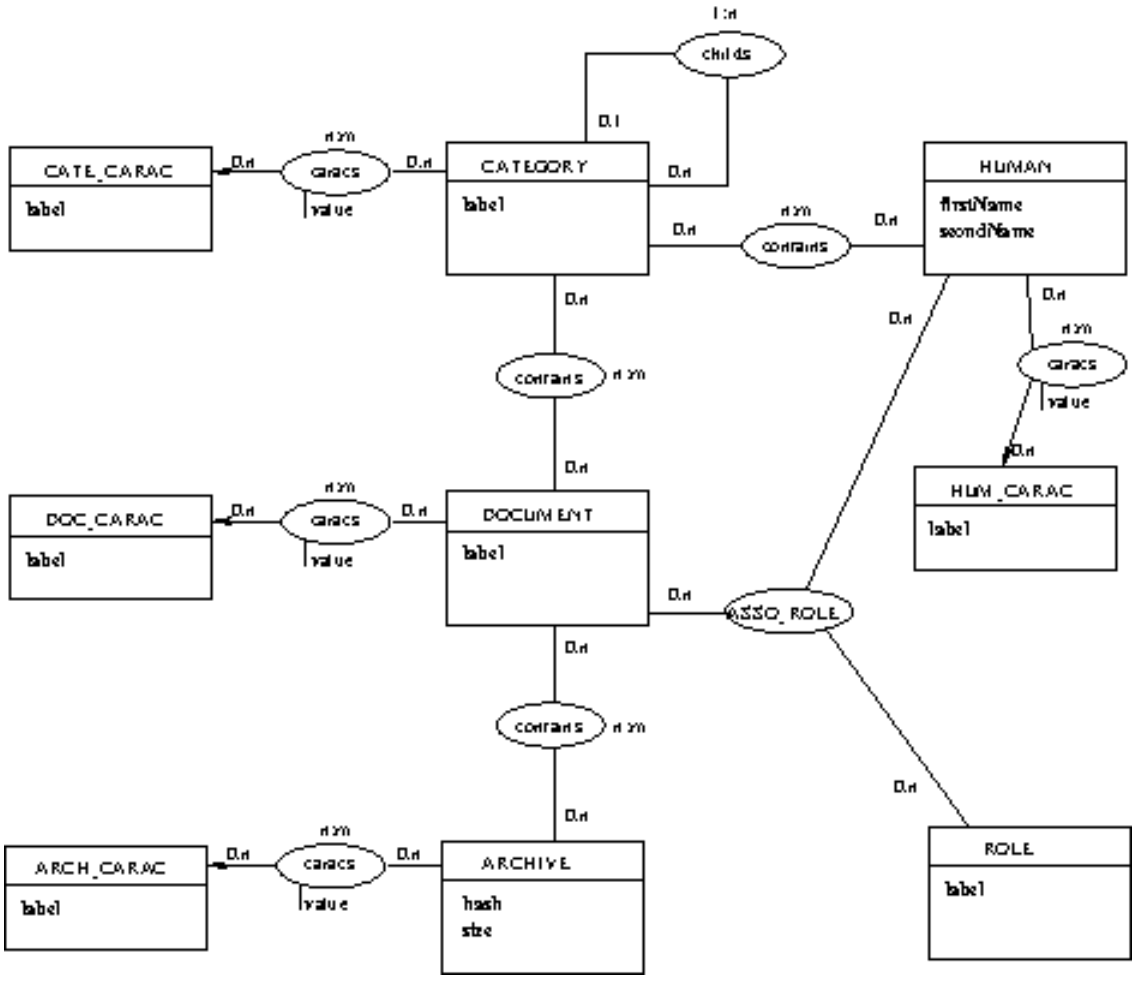
4.4 catalog.txt

The `/etc/mediatex/mdtx-COLL/catalogXXX.txt` files handle the descriptive meta-data of a collection.

Using the "$ mediatex upload[+] [file *file* [as *target*]]* [catalog *file*] [rules *file*] to coll *coll*" query, PUBLISHER (see Section 1.1 [Who], page 3) should provide related descriptive meta-data file for the new archive, using the same grammar language.

The format of this file is oriented to describe a simple movie database. It does not conform to any specific descriptive meta-data schema, like the Dublin Core (`http://dublincore.org/`). However, the translation from and to all meta-data formats remains possible, by adding characteristics to the objects.

Relational schema:



- Entities
  - The ARCHIVE entity lists all archive's data files.
  - The DOCUMENT entity groups archives (data, parts, versions time-stamps ...) related to an archived document. Document are stored using an AVL tree.
  - The CATEGORY entity groups documents by themes (onthology).

    - The HUMAN entity lists humans (using an AVL tree) related to a document.
    - The ROLE entity lists all titles a human contributing to a document may have (ex: producer, actor, ... for a movie).
    - The *_CARAC entities are unidirectional in order to consume less memory.
- Relations
    - The ASSO_ROLE relationship matches a human, a role and a document.
    - The CONTAINS relationships matches ether humans or documents to categories, or records to documents.

Example:

```
# Categories:

Top Category "media"
  "topo" = "Welcome !"
Top Category "image": "media"
Top Category "drawing": "image"
Top Category "animal"
Category "hand": "drawing", "animal"

# Humans:

Human  "Me" "": "drawing"
  "another thing?" = "no"

# Archives:

Archive  022a34b2f9b893fba5774237e1aa80ea:24075
  "format" = "PNG"
Archive  b281449c229bcc4a3556cdcc0d3ebcec:815
  "format" = "XPM"
  "licence" = "GPLv3"

# Documents:

Document "panthere": "animal", "drawing", "hand"
  With "designer" = "Me" ""
  "info" = "[P]erenial [A]rchive [N]etwork [There]"
  022a34b2f9b893fba5774237e1aa80ea:24075
  b281449c229bcc4a3556cdcc0d3ebcec:815
```

Grammar:

```
file: stanzas
    | //empy file

stanzas: stanzas stanza
       | stanza
```

```
stanza: defCategory
      | defHuman
      | defDocument
      | defArchive

caracs: caracs carac
      | carac

carac: string = string

defCategory: newCategory : categories caracs
           | newCategory : categories
           | newCategory caracs
           | newCategory

newCategory: TOP CATEGORY string
           | CATEGORY string

categories: categories , category
          | category

category: string

defHuman: newHuman : categories caracs
        | newHuman : categories
        | newHuman caracs
        | newHuman

newHuman: HUMAN human

human: string string

defArchive: newArchive archCaracs
         | newArchive

newArchive: ARCHIVE archive

archive: hash : number

defDocument: newDocument docCategories docWiths docCaracs docArchives

newDocument: DOCUMENT document

document: string

docCategories: : categories
             |
```

```
docCaracs: caracs
         |

docWiths: roles
        |

roles: roles role
     | role

role: WITH string = human

docArchives: archives
           |

archives: archives archive
        | archive

hash:   [[:xdigit:]]{32}
string: \"(\\\"|[^\n\"]){0,511}\"
number: [[:digit:]]+
```

Code:

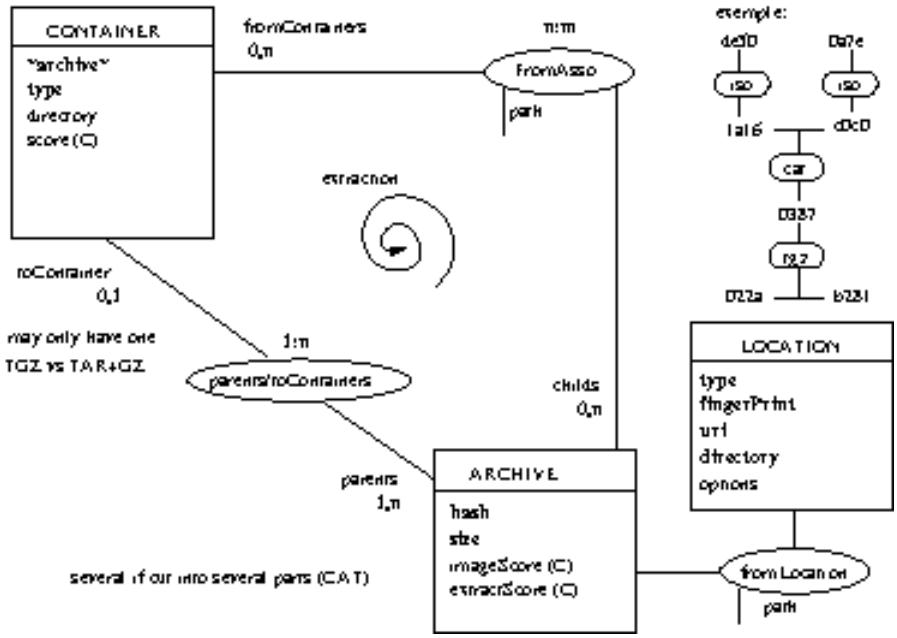
```
src/memory/catalogTree.h
src/memory/catalogTree.c
src/parser/catalogFile.l
src/parser/catalogFile.y
```

4.5 extract.txt

The `/etc/mediatex/mdtx-COLL/extractXXX.txt` files handle the extraction rules for a collection.

Using the "$ mediatex upload[+] [file *file* [as *target*]]* [catalog *file*] [rules *file*] to coll *coll*" query, PUBLISHER (see Section 1.1 [Who], page 3) should provide extraction rules to the new archive to upload, using the same grammar language.

Relational schema:



- Proper entities
  - The CONTAINER entity lists (using an AVL tree) all the container files from where archives are extracted. A Container is identified by its type (ISO,TAR...) and the first archive needed for extraction (ex: `archive-part1.rar`), also called its first parent. Containers may provide a relative directory path to use when extracting the archive. This allow for instance to extract TAR container, that handle files on the root, into a directory. **Notice:** 2 dedicated CONTAINER instances are used internally by MEDIATEX (see Chapter 5 [Conceptual model], page 53) to manage the incoming (recently uploaded) archives, and where to extract supports into the cache.
  - The LOCATION entity lists external ressources (directories). MEDIATEX offers facility to serve files located into a local directory, for instance because theses files are already managed by another software and still may evolve. Locations are not cover by the scoring and geographical duplication algorithms like containers.
- Linked entities
  - The ARCHIVE entity lists the archive's data files
- Relations
  - The FROMASSO relationship enumerate (using an AVL tree) the content of a container. For instance, all files provided by a tarball archive: `tar -tf archive.tar`.

Notice: for the incoming CONTAINER instance, the FROMASSO relationship gives the date the archive was uploaded.

- The PARENTS relationship gives all the parts a container is using to perform extraction. (`archive-part1.rar`, ..., `archive-part`*N*`.rar`).
- The TOCONTAINER relationship gives the container object an Archive may provides.

  Notice: we can only have on containers for an archive: in the case of a compressed tarball for instance, we have to choose between the two possibilities (TAR+GZ or TGZ).
- The FROMLOCATION relationship gives files accessible by the collection user from the location directory. These "located files" are not true archives ; they still may be uploaded as "archives" later.

Exemple:

```
(INC
=>
0a7ecd447ef2acb3b5c6e4c550e6636f:374784 1994-01-01,00:00:00
de5008799752552b7963a2670dc5eb18:391168 2010-01-01,01:00:00
)

(IMG
=>
3a04277dd1f43740a5fe17fd0ae9a5aa:24457  here/logo.gz
)

(ISO
de5008799752552b7963a2670dc5eb18:391168
=>
1a167d608e76a6a4a8b16d168580873c:20480 logoP1.cat
)

(ISO
0a7ecd447ef2acb3b5c6e4c550e6636f:374784
=>
c0c055a0829982bd646e2fafff01aaa6:4066  logoP2.cat
)

(CAT
1a167d608e76a6a4a8b16d168580873c:20480,
c0c055a0829982bd646e2fafff01aaa6:4066
=>
0387eee9820fa224525ff8b2e0dfa9be:24546  logo.tgz
)

(TGZ
0387eee9820fa224525ff8b2e0dfa9be:24546
```

```
=>
directory: here
022a34b2f9b893fba5774237e1aa80ea:24075  logo/logo.png,
b281449c229bcc4a3556cdcc0d3ebcec:815    logo/logo.xpm
)

(COPY
746d6ceeb76e05cfa2dea92a1c5753cd /location
=>
022a34b2f9b893fba5774237e1aa80ea:24075 logo.png
b34bb9bf9ae4ec5b4a5bc2ab3e2a18c5:25088 logo.cpio
)
```

Notice:

- The `INC` container is used to remind the date new incoming where uploaded in order to let a delay, `uploadTTL` `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` for PUBLISHERbefore the uploaded file's score impact the whole collection score.
- The `IMG` container is used to remind an extraction path to used when a support need to be copied into the cache. This feature is optional ; else the `supports/` dirname concatenated with the support's name/basename will be used.

Grammar:

```
file: stanzas
    | //empty file

stanzas: stanzas stanza
       | stanza

stanza: ( container => options childs )
      | ( location =>  options childs )

container: orphaneContainer
         | stdContainer
         | stdContainer parents

orphaneContainer: type

stdContainer: type archive

options: DIRECTORY: string
       | /* none */

location: type hash string string
        | type hash string

parents: parents parent
       | parent
```

```
childs: childs child
      | child

parent: archive

child: archive string

archive: hash : size

hash: [[:xdigit:]]{32}
string: [^[:blank:]\)\n\r]{1,511}
type: INC | IMG | ISO | CAT | TGZ | TBZ | TAR | CPIO
    | GZIP | BZIP | ZIP | RAR | LOCATION
size: [[:digit:]]+
```

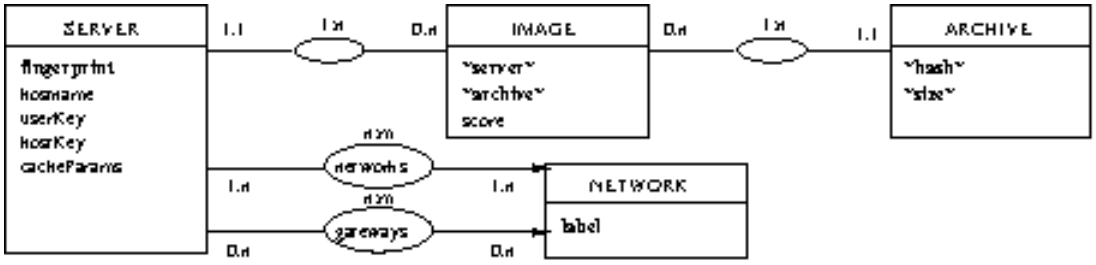
Code:

```
src/memory/extractTree.h
src/memory/extractTree.c
src/parser/extractFile.l
src/parser/extractFile.y
```

4.6 servers.txt

The `/etc/mediatex/mdtx-COLL/servers.txt` file is used to share the latest information between servers including collection's configuration and connectivity keys.

Relational schema:



- Proper entities
  - The SERVER entity lists all the servers participating to a collection.
  - The IMAGE entity lists all the images files known by the collection. Images are identifyed by one archive and one server. So, 2 CDROMs located on the same host are represented by a uniq image object.
  - The NETWORK entity lists public and private networks we logically defined.
- Linked entities
  - The ARCHIVE entity reminds all archives the programm is currently using per collection.
- Relations
  - The NETWORKS relationship lists all the networks a server belongs to.
  - The GATEWAYS relationship lists all the networks for which a server acts like a NAT gateway.

Exemple:

```
# This file is managed by MediaTeX software.
# MediaTeX serv1-hello@127.0.0.1:22 servers list:

master     a3219b5a0512d08a291aa73b85d81674
dnsHost    dns.round.robbin.name.org
collKey    01234567890abcde
https      yes

# self-ingestion parameters
logApache  no
logGit     no
logAudit   no

# score parameters
uploadTTL  1 Month
serverTTL  2 Week
suppTTL    5 Year
```

```
maxScore   10.00
badScore   1.00
powSupp    2.00
factSupp   2.00
fileScore  10.00
minGeoDup  2

Server 58225510312a82b017b07aabf7b89657
comment   "Put your personal greeting here"
label     serv2
host      127.0.0.1
lastCommit 2015-10-05,22:52:28
mdtxPort  6002
sshPort   22
wwwPort   443
networks  private, www
gateways  private
cacheSize 100 Mo
cacheTTL  15 Day
queryTTL  1 Week
# keys:
userKey   "..."
hostKey   "..."
end

Server a3219b5a0512d08a291aa73b85d81674
comment   "Put your personal greeting here"
label     serv1
host      127.0.0.1
lastCommit 2015-10-05,22:52:35
mdtxPort  6001
sshPort   22
wwwPort   443
networks  www
cacheSize 100 Mo
cacheTTL  15 Day
queryTTL  1 Week
provide   de5008799752552b7963a2670dc5eb18:391168=10.00
# keys:
userKey   "..."
hostKey   "..."
end
```

Grammar:

```
file: //empty file
    | headers
    | headers stanzas
```

```
headers: headers header
       | header

header: DNSHOST string
      | COLLKEY string
      | HTTPS boolean
      | LOGAPACHE boolean
      | LOGGIT boolean
      | LOGAUDIT boolean
      | SERVERTTL number time
      | SUPPTTL number time
      | UPLOADTTL number time
      | MAXSCORE score
      | BADSCORE score
      | POWSUPP score
      | FACTSUPP score
      | FILESCORE score
      | MINGEODUPP number

stanzas: stanzas stanza
       | stanza

stanza: SERVER server lines END

server: HASH

lines: lines line
     | line

line: LABEL string
    | COMMENT string
    | HOST string
    | LASTCOMMIT date
    | MDTXPORT number
    | SSHPORT number
    | NETWORKS networks
    | GATEWAYS gateways
    | MDTXPORT number
    | SSHPORT number
    | WWWPORT number
    | USERKEY string
    | HOSTKEY string
    | CACHESIZE number size
    | CACHETTL number time
    | QUERYTTL number time
    | PROVIDE images
```

```
networks: networks , string
        | string

gateways: gateways , string
        | string

images: images , image
      | image

image: hash : number = score

size:    GO | MO | KO | O
time:    MONTH | WEEK | DAY | HOUR | MINUTE | SECOND
boolean: yes | no
score:   [[:digit:]]+\.[[:digit:]]+
number:  [[:digit:]]+
string:  [^[:blank:]:,=\r\n]{1,511} | \"[^\n\"]{1,1023}\"
hash:    [[:xdigit:]]{32}

date:   {year}-{month}-{day},{HOUR}:{min}:{sec}

year:   [[:digit:]]{4}
month:  [[:digit:]]{2}
day:    [[:digit:]]{2}
HOUR:   [[:digit:]]{2}
min:    [[:digit:]]{2}
sec:    [[:digit:]]{2}
```

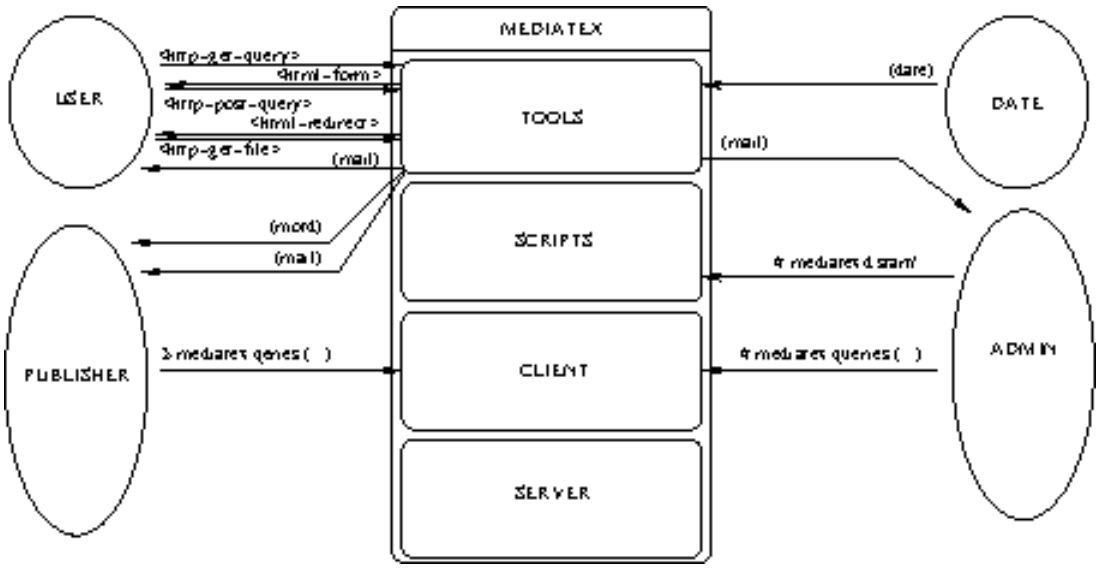
Code:

```
src/memory/serverTree.h
src/memory/serverTree.c
src/parser/serverFile.l
src/parser/serverFile.y
```

5 Specifications

The information system is divided into 4 activities:

Flood diagram:



- Events in

  `# /etc/init.d/mediatexd start`
  `# /etc/init.d/mediatexd stop`
  `# /etc/init.d/mediatexd status`
  `# /etc/init.d/mediatexd reload`
  from ADMIN (see Section 1.1 [Who], page 3) to **init.d** (see Section 5.2.1 [init.d], page 74): start/stop the MEDIATEX daemon.

  `# mediatex adm init`
  `# mediatex adm remove`
  `# mediatex adm purge`
  `# mediatex adm add user` *`user`*
  `# mediatex adm del user` *`user`*
  from ADMIN to **misc** (see Section 5.3.4 [misc], page 100): configure the MEDIATEX system.

  `# mediatex adm add coll [`*`serv-`*`]`*`coll`*`[@`*`host`*`[:`*`port`*`]]`
  `# mediatex adm del coll` *`coll`*
  from ADMIN to **conf** (see Section 5.3.1 [conf], page 92): configure collections.

  `(date)` from DATE (see Section 1.1 [Who], page 3) to **Cron** (see Section 5.1.6 [Cron], page 68): scheduled tasks.

`$ mediatex add key file to coll coll`
`$ mediatex del key fingerprint from coll coll`
`$ mediatex upgrade[+] [coll coll]`
: from PUBLISHER (see Section 1.1 [Who], page 3) to **serv** (see Section 5.3.3 [serv], page 97): manage collection settings.

`$ mediatex ls [master] coll`
`$ mediatex add supp supp to (all|coll coll)`
`$ mediatex del supp supp from (all|coll coll)`
: from PUBLISHER to **conf** (see Section 5.3.1 [conf], page 92): list collections ; share/withdraw local supports to collections.

`$ mediatex ls supp`
`$ mediatex add supp supp on path`
`$ mediatex add file path`
`$ mediatex del supp supp`
`$ mediatex note supp supp as text`
`$ mediatex check supp supp on file`
: from PUBLISHER to **supp** (see Section 5.3.2 [supp], page 94): manage local supports.

`$ mediatex upload[+] [file file [as target]]* [catalog file] [rules file] to coll coll`
: from PUBLISHER to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload an incoming archive into the cache.

`$ mediatex make [coll coll]`
`$ mediatex clean [coll coll]`
`$ mediatex su [coll coll]`
`$ mediatex audit for mail coll coll`
: from PUBLISHER to **misc** (see Section 5.3.4 [misc], page 100): build/clean the HTML catalogue ; change to server or collection system user ; simulate retrieving all archives.

`<http-get-query>`
`<http-post-query>`
`<http-get-file>`
: from USER (see Section 1.1 [Who], page 3) to **Apache** (see Section 5.1.2 [Apache], page 60): USER's queries generated by its internet browser.

- Processing

  see Section 1.2 [What], page 3

- Events out

`(motd)`
: from **Cron** (see Section 5.1.6 [Cron], page 68) to PUBLISHER: message of the day the PUBLISHER see at log-in.

`<html-form>`
: (if archive is not available) ...

`<html-redirect>`
: (if archive is available)
from **Apache** (see Section 5.1.2 [Apache], page 60) to USER: replies to the USER's internet browser.

`(mail to USER)`
: from **Sendmail** (see Section 5.1.3 [Exim], page 64) to USER: archive availability notification.

`(mail to PUBLISHER)`
: from **Sendmail** (see Section 5.1.3 [Exim], page 64) to PUBLISHER: audit availability notification.

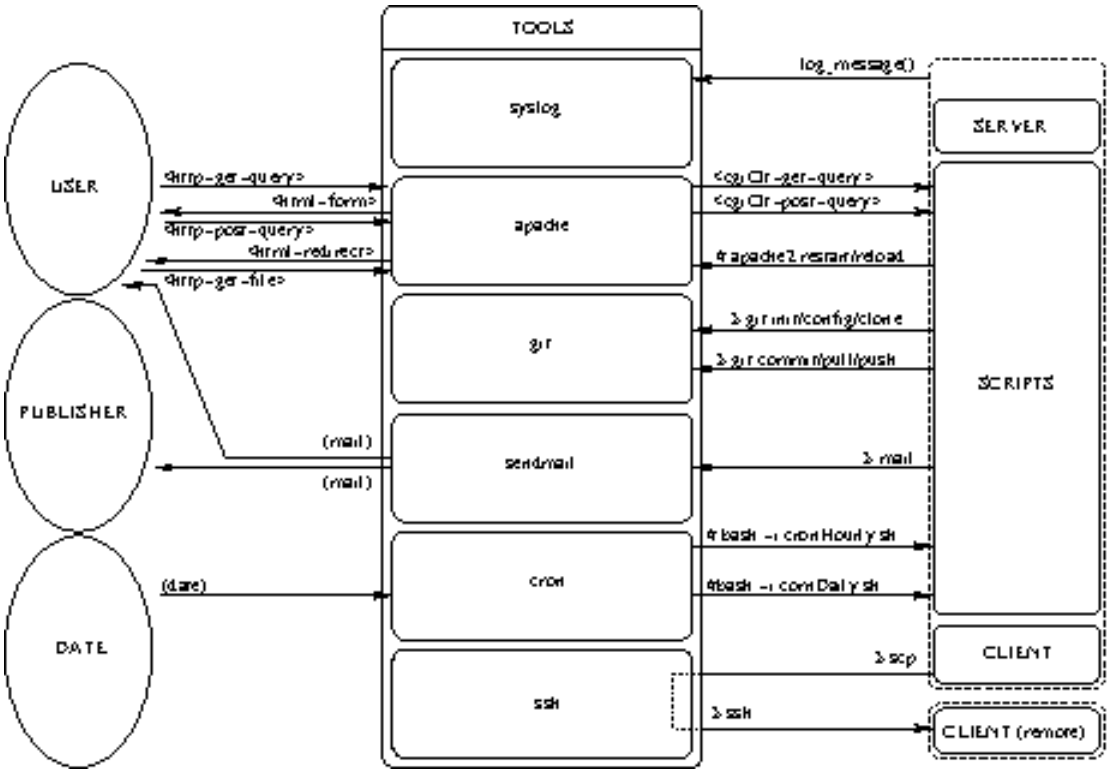
`(mail to ADMIN)`
: from **Sendmail** (see Section 5.1.3 [Exim], page 64) to ADMIN: scheduled task error notification.

- Data in/out
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)
  - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
  - `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)
  - `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)

5.1 Tools

The TOOLS activity gather external GNU system tools MEDIATEX is using. This activity is divided into 6 process:

Flood diagram:



- Events in

  `log_message()`
  : from CLIENT (see Section 5.3 [Client], page 87), SERVER (see Section 5.4 [Server], page 107) and SCRIPTS (see Section 5.2 [Scripts], page 70) to **Syslog** (see Section 5.1.1 [Syslog], page 58): log MEDIATEX operations.

  `<http-get-query>`
  `<http-post-query>`
  `<http-get-file>`
  : from USER (see Section 1.1 [Who], page 3) to **Apache** (see Section 5.1.2 [Apache], page 60): USER's queries generated by its internet browser.

  `# /etc/init.d/apache2 reload`
  `# /etc/init.d/apache2 restart`
  : from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) to **Apache** (see Section 5.1.2 [Apache], page 60): re-configure APACHE.

  `$ mail`
  : from **deliver** (see Section 5.2.10 [deliver], page 85), **audit** (see Section 5.2.11 [audit], page 85) or **Cron** (see Section 5.1.6 [Cron], page 68) to **Sendmail** (see Section 5.1.3 [Exim], page 64): mail notifications.

`$ scp`
: from **misc** (see Section 5.3.4 [misc], page 100) to **Ssh** (see Section 5.1.4 [Ssh], page 65): copy file from one remote server's cache.

`$ git init --bare`
`$ git config`
: from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) or **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) to **Git** (see Section 5.1.5 [Git], page 66): initialise a GIT repository.

`$ git clone`
: from **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) to **Git** (see Section 5.1.5 [Git], page 66): checkout a GIT collection module.

`$ git config`
`$ git commit`
`$ git pull`
`$ git push`
: from **upgrade-commit-pull-push** (see Section 5.2.7 [upgrade-commit-pull-push], page 82) to **Git** (see Section 5.1.5 [Git], page 66): synchronise a GIT module.

`(date)`
: from DATE (see Section 1.1 [Who], page 3) to **Cron** (see Section 5.1.6 [Cron], page 68): scheduled tasks.

- Processing

  see Section 1.5 [How], page 7

- Events out

`<cgiClt-get-query>`
`<cgiClt-post-query>`
: from **Apache** (see Section 5.1.2 [Apache], page 60) to **cgiClient** (see Section 5.2.2 [cgiClient], page 75): forward the USER's HTTP query.

`<html-form>`
: (if archive is not available) ...

`<html-redirect>`
: (if archive is available)
from **Apache** (see Section 5.1.2 [Apache], page 60) to USER

`(mail to USER)`
: from **Sendmail** (see Section 5.1.3 [Exim], page 64) to USER: archive availability notification.

`(mail to PUBLISHER)`
: from **Sendmail** (see Section 5.1.3 [Exim], page 64) to PUBLISHER: audit availability notification.

`(mail to ADMIN)`
: from **Sendmail** (see Section 5.1.3 [Exim], page 64) to ADMIN: scheduled task error notification.

`# bash -i /usr/share/mediatex/scripts/cron_hourly.sh`
`# bash -i /usr/share/mediatex/scripts/cron_daily.sh`
`# bash -i /usr/share/mediatex/scripts/cron_monthly.sh`
from **Cron** (see Section 5.1.6 [Cron], page 68) to **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76): scheduled tasks.

`$ mediatex upload[+] [file` *`file`* `[as` *`target`*`]]* `[catalog` *`file`*`] [rules` *`file`*`] to coll` *`coll`*
from **Syslog** (see Section 5.1.1 [Syslog], page 58) to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload collection's APACHE access logs to the cache so as to backup them.

- Data in

```
/etc/apache2/conf-available/mediatex.conf
/etc/apache2/conf-available/mediatex-mdtx.conf
/etc/ssh/sshd_config
/etc/cron.d/mediatex
~mdtx/jail/etc/group
~mdtx/jail/etc/password
~mdtx-COLL/.ssh/authorized_keys
~mdtx-COLL/.ssh/known_hosts
~mdtx-COLL/.ssh/id_dsa
~mdtx-COLL/.ssh/id_dsa.pub
~mdtx-COLL/.ssh/config
~mdtx-COLL/public_html/.htaccess
~mdtx-COLL/public_html/score/.htaccess
~mdtx-COLL/public_html/cgi/.htaccess
~mdtx-COLL/public_html/cache/.htaccess
```

- Data in/out

```
/etc/mediatex/mdtx-COLL/apache2/htgroup
/etc/mediatex/mdtx-COLL/apache2/htpasswd
/var/lib/mediatex/mdtx/mdtx
/var/lib/mediatex/mdtx/mdtx-COLL
~/mdtx/git/mdtx
~/mdtx/git/mdtx-COLL
```

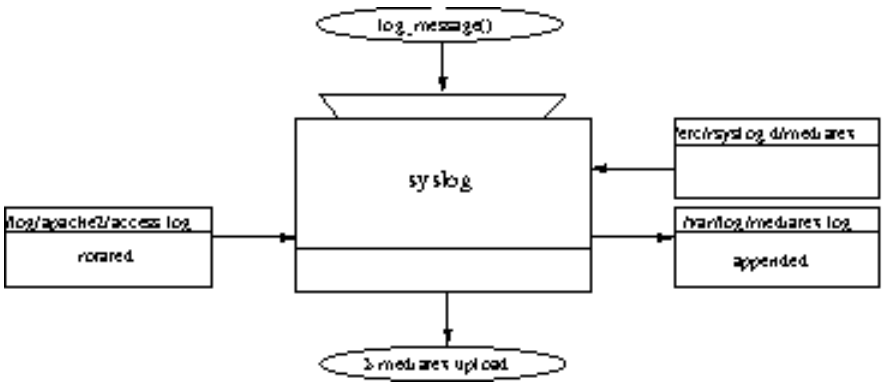
5.1.1 Syslog

syslog (actually RSYSLOG: see `man rsyslogd (8)` and LOGROTATE: see `man logrotate (8)`) is used by all MEDIATEX's binaries and scripts.

3 environment variables manage the content to log:

- *MDTX_LOG_FACILITY*
- *MDTX_LOG_FILE*
- *MDTX_LOG_SEVERITY*

These variables are overwritten when using the corresponding command line options (see Section 3.2 [Options], page 21).



- Events in

  `log_message()`
  : from all **processes**

- Processing

  Log the activity (and rotate the logs).

- Events out

  `$ mediatex upload[+] [file` *`file`* `[as` *`target`*`]]* `[catalog` *`file`*`] [rules` *`file`*`] to coll` *`coll`*
  : to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload collection's APACHE access logs to the cache so as to backup them.

- Data in

  `/etc/rsyslog.d/mediatex_log.conf`
  : `local2.info /var/log/mediatex.log`

    Notice: This configuration tel RSYSLOG to ignore `debug` messages (even if MEDIATEX send debug messages, you will not get them). The higher the level is and the less messages you get: `err > warning > notice > info > debug`

  `/etc/rsyslog.conf`
  : Also, RSYSLOG will drop messages when they are coming too quickly, and you may want to change this setting (see Chapter 8 [Reporting bugs], page 136).

  `/etc/logrotate.d/mediatex_logrotate.conf`
  : Manage MEDIATEX's logs rotation.

  `/etc/logrotate.d/httpd-prerotate/mediatex_logrotate`
  : Filtre and upload the **Apache** (see Section 5.1.2 [Apache], page 60)'s latest logs to the related collections, after rotation, if enabled into `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49).

  `/var/log/apache2/access.log`

- Data out

```
/var/log/mediatex.log
```

Code:

```
etc/rsyslog.src
etc/logrotate.src
etc/logrotate_apache.src
/src/misc/log.c
```

5.1.2 Apache

User (see Section 1.1 [Who], page 3) use Apache to browse the HTML catalogue and access archives. (For tips, you should maybe read this section first : see Section 6.4 [HTTP], page 124). Apache is configured to query the Server (see Section 5.4 [Server], page 107) via the **cgiClient** (see Section 5.2.2 [cgiClient], page 75) CGI script.

The followings Apache's modules are enabled by Mediatex:

`auth_digest`
: to cypher password (even if using HTTP).

`authz_groupfile`
: to filter access to the catalogue, cache, extraction rules and GIT history.

`autoindex`
: to display the cache's content.

`cgi` : to call get.cgi and cgit.cgi scripts.

`env` : to use absolute paths into the HTML page's headers and to retrieve the configuration of cgit.

`include` : SSI (server side includes) are used to reduce the size of the HTML catalogue.

`rewrite` : to force using HTTPS protocol (if wanted).

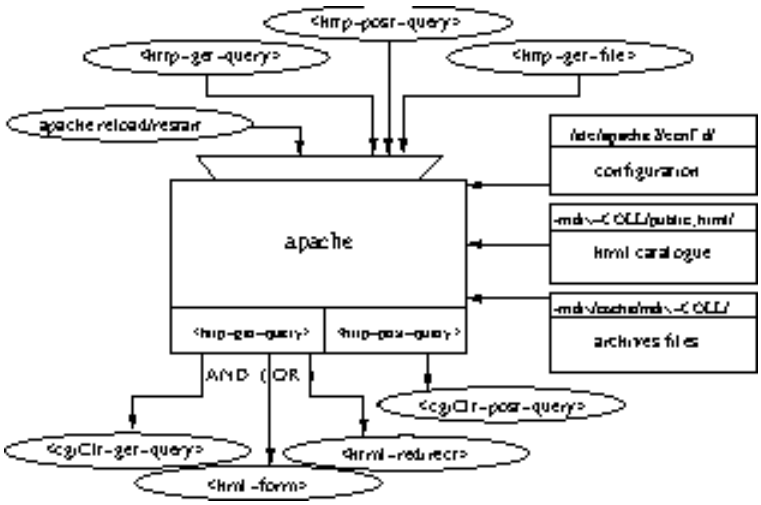
`userdir` : to access collection with a similar URL on every servers, by using the tilde ("~").

`setenvif` : to provide switch to disable authentification, as cache's .htaccess file must be generated by make.

`ssl` : to enable the HTTPS protocol.

They are left when mediatex software is removed, as they could still be in used for another purpose.

Process conceptual model:



- Events in

  `# /etc/init.d/apache2 restart`
  : from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77): re-configure APACHE.

  `# /etc/init.d/apache2 reload`
  : from **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): make APACHE aware of new collection's system groups it belongs to.

  `<http-get-query>`
  : from USER: resulting from a click on an archive's URL.

  `<http-post-query>`
  : from USER: requesting the submitted HTML form to provide its eMAIL address, in order to be called back when archive will become available.

  `<http-get-file>`
  : from USER: final request for a file available into the cache.

- Processing

  Manage USER's queries generated by its internet browser and replies from **cgiClient** (see Section 5.2.2 [cgiClient], page 75).

- Events out

  `<cgiClt-get-query>`
  `<cgiClt-post-query>`
  : to **cgiClient** (see Section 5.2.2 [cgiClient], page 75): forward the USER's HTTP query.

  `<html-form>`
  : (if archive is not available) ...

`<html-redirect>`
: (if archive is available)
to USER: send back either a form asking for an eMAIL address, or a redirection link to the archive into one server's cache.

The APACHE's configuration files given bellow are generated by **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) and **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) scripts.

- Data in

`/etc/apache2/conf-available/mediatex.conf`

```
Alias /mediatex /var/cache/mediatex
<Directory /var/cache/mediatex>
Require all denied
AllowOverride None
</Directory>
```

`/etc/apache2/conf-available/mediatex-mdtx.conf`

```
<IfModule mod_userdir.c>
<Directory /var/cache/mediatex/mdtx/public_html>
# enable cgi for ~mdtx
Options +ExecCGI +FollowSymLinks
AddHandler cgi-script .cgi
SetEnv CGIT_CONFIG /var/cache/mediatex/mdtx/cgitrc
Require local
Require all denied
</Directory>
<Directory /var/cache/mediatex/mdtx/home/mdtx-*/public_html>
Require all granted
AllowOverride All
</Directory>
</IfModule>
```

This file (and only this one) is generated by $ mediatex make [coll *coll*].

`~mdtx-COLL/public_html/cache/.htaccess`

```
# fancy index for cache
Options +Indexes
SetEnv HOME /~mdtx-hello
HeaderName /mediatex/mdtx/home/mdtx-hello/public_html/cacheHeader.shtml
ReadmeName /mediatex/mdtx/home/mdtx-hello/public_html/footer.html

# login/password
Require env NO_AUTH
Require group cache
```

The following files are shared into each collection by servers using GIT (they are symbolic links). For instance, the first lines of the above file let the PUBLISHER modify the security behaviours for all servers.

`~mdtx-COLL/public_html/.htaccess`

```
# uncomment to force https (no more http available)
```

```
     #SSLOptions +StrictRequire
     #SSLRequireSSL

     # uncomment to disable authentification
     #SetEnvIf Request_Protocol "^H" NO_AUTH

     # server side includes
     Options +Includes
     DirectoryIndex index.shtml

     # login/password
     AuthType Digest
     AuthName "mdtx-hello"
     AuthDigestProvider file
     AuthUserFile  /etc/mediatex/mdtx-hello/apache2/htpasswd
     AuthGroupFile /etc/mediatex/mdtx-hello/apache2/htgroup
~mdtx-COLL/public_html/index/.htaccess
     # login/password
     Require env NO_AUTH
     Require group index
~mdtx-COLL/public_html/score/.htaccess
     # login/password
     Require env NO_AUTH
     Require group score
~mdtx-COLL/public_html/cgi/.htaccess
     # enable cgi
     Options +ExecCGI +FollowSymLinks
     AddHandler cgi-script .cgi

     <Files get.cgi>
            # set log severity
            SetEnv MDTX_LOG_FACILITY local2
            SetEnv MDTX_LOG_SEVERITY_MAIN info

            # login/password
            Require env NO_AUTH
            Require group index
     </Files>

     <Files cgit.cgi>
            # cgit configuration
            SetEnv CGIT_CONFIG ../../cgitrc

            # login/password
            Require env NO_AUTH
            Require group history
```

```
</Files>

<Files put.*>
       # login/password
       Require group upload
</Files>
```

User (see Section 1.1 [Who], page 3) logins and passwords are manage by the 2 files bellow. A first entry is generated by the **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) script using the server label as login (`mdtx` by default) and the password you provide.

`/etc/mediatex/mdtx-COLL/apache2/htgroup`

```
index: mdtx, username1
cache: mdtx
score: mdtx, username1
history: mdtx
```

- "index" section is the descriptive catalog.
- "cache" section gives access to the archive files.
- "score" section details the extraction meta-data. It provides scores for archives, servers and the whole collection.
- "version" section shows the history of meta-data.
- optional "upload" section provides a form to add archives.

`/etc/mediatex/mdtx-COLL/apache2/htpasswd`

```
mdtx:mdtx-COLL:75a895c47530b5177d8ba1616f49d648
username1:mdtx-COLL:855531b7d2e25b190c6d1da662da5f90
```

You may want to automate the encryption in order to add new users (the `htdigest` tool does not accept pipe redirection). Simplest way is:

```
printf "username1:mdtx-COLL:yourPassword" | md5sum
855531b7d2e25b190c6d1da662da5f90  -
```

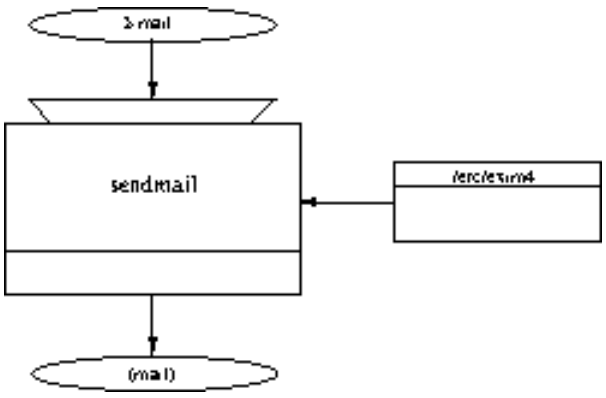
Code:

`scripts/libs/htdocs.sh`

5.1.3 Sendmail

Mediatex uses Exim4 (see *Specification of the Exim Mail Transfer Agent*) to send mails so as to notify Actors. Please configure Exim (or any other mail agent) so as your host can effectively delivers sent mails.

Process conceptual model:



- Events in

  `$ mail` from **deliver** (see Section 5.2.10 [deliver], page 85), **audit** (see Section 5.2.11 [audit], page 85) or **Cron** (see Section 5.1.6 [Cron], page 68)

- Processing

  Send a mail.

- Events out

  `(mail to USER)`
  : to User (see Section 1.1 [Who], page 3): when an archive becomes available into the cache.

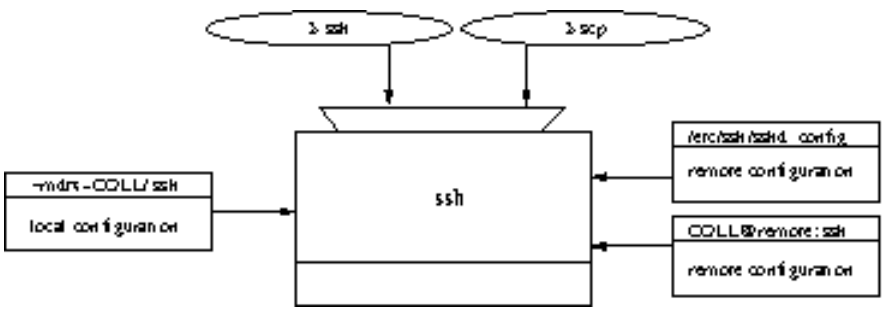
  `(mail to PUBLISHER)`
  : to Publisher (see Section 1.1 [Who], page 3): when an audit was delivered.

  `(mail to ADMIN)`
  : to Admin (see Section 1.1 [Who], page 3): when **Cron** (see Section 5.1.6 [Cron], page 68) complains.

5.1.4 Ssh

Ssh (see `man ssh(1)`) is used to exchange meta-data and archives files by using a "chroot" jail for each collection account, as **bind-unbind** (see Section 5.2.8 [bind-unbind], page 83) script provides the cache and gitbare directories into that jail.

Process conceptual model:



- Events in

`$ ssh` from **Git** (see Section 5.1.5 [Git], page 66): access the `GIT` bare repository.

`$ scp` from **misc** (see Section 5.3.4 [misc], page 100): retrieve an archive from a remote SERVER (see Section 5.4 [Server], page 107)'s cache.

- Data in The SSH server's configuration files bellow is are managed by **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) and **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) scripts.

`/etc/ssh/sshd_config`

```
...
# <<<mdtx-*
Match user mdtx-*
          ChrootDirectory /var/cache/mediatex/mdtx/jail
          X11Forwarding no
          AllowTcpForwarding no
# mdtx-*>>>
```

`/var/cache/mediatex/mdtx/jail/etc/group`

```
root:x:0:
www-data:x:33:www-data
mdtx:x:120:www-data
mdtx_md:x:123:mdtx,www-data
mdtx-COLL:x:124:www-data,mdtx
```

`/var/cache/mediatex/mdtx/jail/etc/password`

```
root:x:0:0:root:/root:/bin/bash
www-data:x:33:33:www-data:/var/www:/bin/sh
mdtx:x:112:120::/var/cache/mediatex/mdtx:/bin/bash
mdtx-COLL:x:114:124::../home/mdtx-COLL:/bin/bash
```

`~mdtx-COLL/.ssh/authorized_keys`
`~mdtx-COLL/.ssh/known_hosts`
`~mdtx-COLL/.ssh/id_dsa`
`~mdtx-COLL/.ssh/id_dsa.pub`

The SSH client's configuration file bellow is managed during MEDIATEX's upgrades.

`~mdtx-COLL/.ssh/config`

```
# Do not ask for password
BatchMode yes
Compression yes
Host hostname1
        Port 22
```

Code:

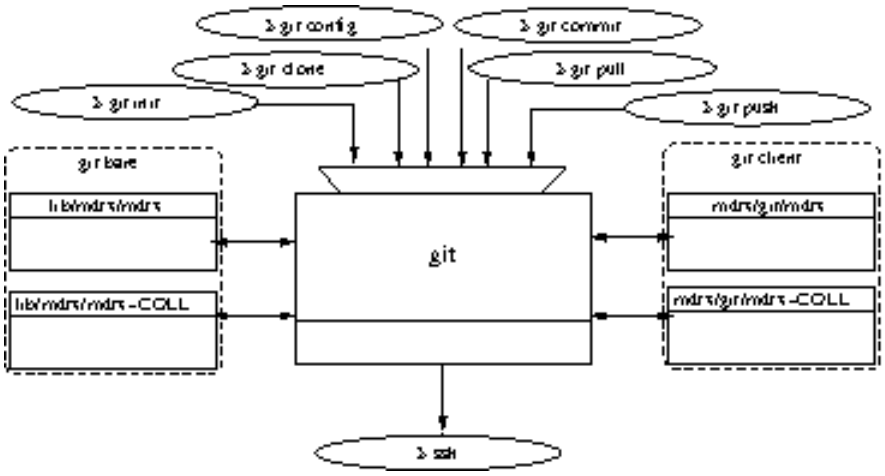
`scripts/lib/ssh.sh`
`scripts/lib/jail.sh`

5.1.5 Git

GIT (`man 1 git`) is used to publish meta-data and to log operations performed on these files. There is one `gitbare` repository for each server one each host. This repository handle

one module for the local server's meta-data, and one for each collection on the "master" servers only. Remote servers access the "master" (or so called "bare") GIT collection's repository by using **Ssh** (see Section 5.1.4 [Ssh], page 65). The **Apache** (see Section 5.1.2 [Apache], page 60) configuration files and the **Ssh**'s public keys are also distributed among servers using GIT. The GIT configuration and collection modules are setup respectively by **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) and **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) scripts.

Process conceptual model:



- Events in

  `$ git init --bare`
  `$ git config`
  : from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) or **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): initialise a GIT repository.

  `$ git clone`
  : from **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): checkout a GIT collection module.

  `$ git config`
  `$ git commit`
  `$ git pull`
  `$ git push`
  : from **upgrade-commit-pull-push** (see Section 5.2.7 [upgrade-commit-pull-push], page 82): synchronise a GIT module.

- Processing

  Synchronise the meta-data files between servers.

- Events out

  `$ ssh` to **Ssh** (see Section 5.1.4 [Ssh], page 65): access the GIT bare repository.

- Data

`/var/lib/mediatex/mdtx/mdtx`
`/var/lib/mediatex/mdtx/mdtx-COLL`
: The GIT bare repositories

`~/mdtx/git/mdtx`
`~/mdtx/git/mdtx-COLL`
: The GIT cloned modules.

The configuration of the client repository is manage by MEDIATEX upgrade (see Section 7.5 [Scenario E], page 134).

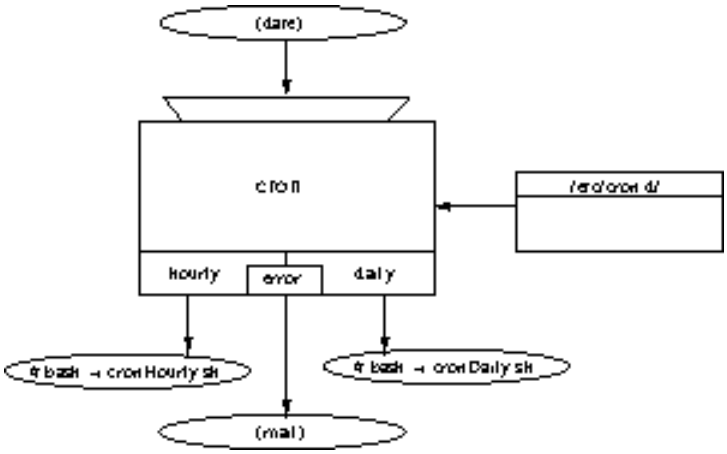
Code:

`scripts/lib/git.sh`

5.1.6 Cron

Cron periodicaly run queries. It implement the DATE actor.

Process conceptual model:



- Events in

  `(date)` from DATE (see Section 1.1 [Who], page 3)

- Processing
  - Scripts should run silently. If it is not the case, or on error, a mail containing the script's outpouts will be sent to the ADMIN.

- Events out

  ```
  # bash -i /usr/share/mediatex/scripts/cron_hourly.sh
  # bash -i /usr/share/mediatex/scripts/cron_daily.sh
  # bash -i /usr/share/mediatex/scripts/cron_monthly.sh
  ```
  to **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76): scheduled tasks.

  `$ mail` to **Sendmail** (see Section 5.1.3 [Exim], page 64): notify scheduled task errors to ADMIN (see Section 1.1 [Who], page 3).

- Data in

```
/etc/cron.d/mediatex
     SHELL=/bin/sh
     PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
     MAILTO=root

     # m  h dom mon dow user command
     XX  *   *   *   * root /usr/share/mediatex/scripts/cron_hourly.sh
     XX XX   *   *   * root /usr/share/mediatex/scripts/cron_daily.sh
     XX XX   X   *   * root /usr/share/mediatex/scripts/cron_monthly.sh
```

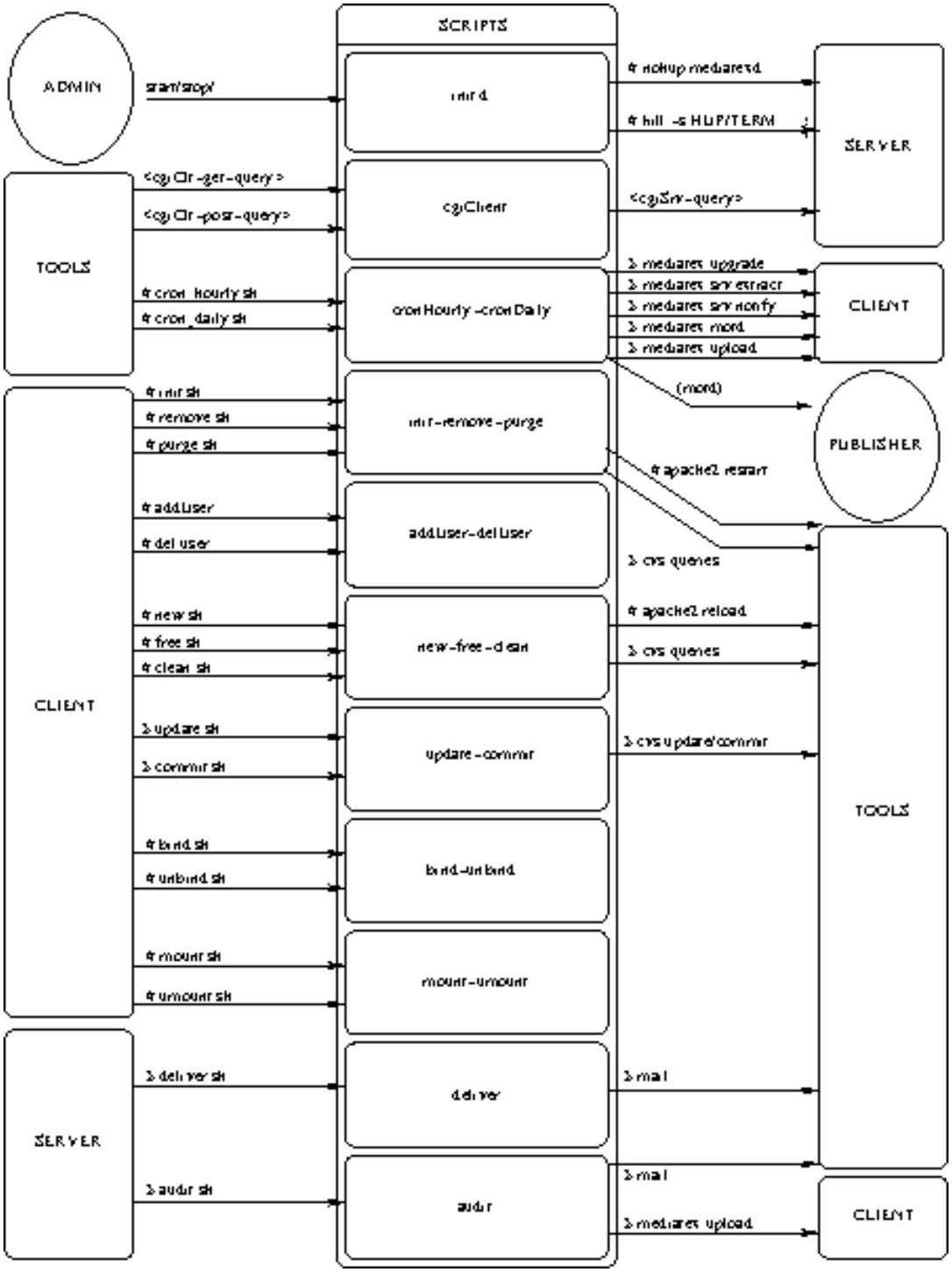
Notice: the triger dates are choosen randomly during MEDIATEX initialisation.

Code:

`misc/mediatex_cron`

5.2 Scripts

The SCRIPTS activity gather system level operations MEDIATEX is using. This activity provides basic fonctions on BASH files that are not describe in this documentation. This activity is divided into 11 process:



- Events in

```
# /etc/init.d/mediatexd start
# /etc/init.d/mediatexd stop
# /etc/init.d/mediatexd status
# /etc/init.d/mediatexd reload
```
from ADMIN (see Section 1.1 [Who], page 3) to **init.d** (see Section 5.2.1 [init.d], page 74): start/stop the MEDIATEX daemon.

```
<cgiClt-get-query>
<cgiClt-post-query>
```
from **Apache** (see Section 5.1.2 [Apache], page 60) to **cgiClient** (see Section 5.2.2 [cgiClient], page 75): USER's HTTP forwarded query.

```
# bash -i /usr/share/mediatex/scripts/cron_hourly.sh
# bash -i /usr/share/mediatex/scripts/cron_daily.sh
# bash -i /usr/share/mediatex/scripts/cron_monthly.sh
```
from **Cron** (see Section 5.1.6 [Cron], page 68) to **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76): scheduled tasks.

```
# /usr/share/mediatex/scripts/init.sh
# /usr/share/mediatex/scripts/remove.sh
# /usr/share/mediatex/scripts/purge.sh
```
from **misc** (see Section 5.3.4 [misc], page 100) to **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77): initialise, remove and purge MEDIATEX software.

```
# /usr/share/mediatex/scripts/add-user.sh
# /usr/share/mediatex/scripts/del-user.sh
```
from **serv** (see Section 5.3.3 [serv], page 97) to **addUser-delUser** (see Section 5.2.5 [addUser-delUser], page 79): manage PUBLISHER users.

```
# /usr/share/mediatex/scripts/new.sh
# /usr/share/mediatex/scripts/free.sh
```
from **conf** (see Section 5.3.1 [conf], page 92) to **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): create/subscribe or destroy/unsubscribe a collection.

```
# /usr/share/mediatex/scripts/clean.sh
```
from **misc** (see Section 5.3.4 [misc], page 100) to **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): clean the local HTML catalogue.

```
$ /usr/share/mediatex/scripts/upgrade.sh
$ /usr/share/mediatex/scripts/commit.sh
$ /usr/share/mediatex/scripts/commit.sh
$ /usr/share/mediatex/scripts/commit.sh
```
from CLIENT (see Section 5.3 [Client], page 87) to **upgrade-commit-pull-push** (see Section 5.2.7 [upgrade-commit-pull-push], page 82): manage GIT synchronisation.

`# /usr/share/mediatex/scripts/bind.sh`
`# /usr/share/mediatex/scripts/unbind.sh`
: from **misc** (see Section 5.3.4 [misc], page 100) to **bind-unbind** (see Section 5.2.8 [bind-unbind], page 83): manage collection repository binding on the chrooted jail for SSH remote access.

`# /usr/share/mediatex/scripts/mount.sh`
`# /usr/share/mediatex/scripts/umount.sh`
: from **supp** (see Section 5.3.2 [supp], page 94) to **mount-umount** (see Section 5.2.9 [mount-umount], page 84): manage mounting ISO devices.

`$ /usr/share/mediatex/scripts/deliver.sh`
: from **delivering** (see Section 5.4.7 [delivering], page 121) to **deliver** (see Section 5.2.10 [deliver], page 85): manage archive availability notification.

`$ /usr/share/mediatex/scripts/audit.sh`
: from **delivering** (see Section 5.4.7 [delivering], page 121) to **audit** (see Section 5.2.11 [audit], page 85): audit an archive.

- Processing

  Provide a BASH library.

- Events out

`# nohup mediatexd`
`# kill -s TERM` *PID*
`# kill -s HUP` *PID*
: from **init.d** (see Section 5.2.1 [init.d], page 74) or **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) to SERVER (see Section 5.4 [Server], page 107): standard UNIX process management (HUP signal for reload...).

`# /etc/init.d/apache2 restart`
: from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) to **Apache** (see Section 5.1.2 [Apache], page 60): re-configure APACHE.

`# /etc/init.d/apache2 reload`
: from **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) to **Apache** (see Section 5.1.2 [Apache], page 60): make APACHE aware of changes into its configuration.

`<cgiSrv-query>`
: from **cgiClient** (see Section 5.2.2 [cgiClient], page 75) to **cgiServer** (see Section 5.4.3 [cgiServer], page 115): socket query asking for an archive.

`$ git init --bare`
`$ git config`
: from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77) or **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) to **Git** (see Section 5.1.5 [Git], page 66): initialise a GIT repository.

`$ git clone`
: from **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80) to **Git** (see Section 5.1.5 [Git], page 66): checkout a GIT collection module.

`$ git config`
`$ git commit`
`$ git pull`
`$ git push`
: from **upgrade-commit-pull-push** (see Section 5.2.7 [upgrade-commit-pull-push], page 82) to **Git** (see Section 5.1.5 [Git], page 66): synchronise a GIT module.

`$ mail`
: from **deliver** (see Section 5.2.10 [deliver], page 85) or **audit** (see Section 5.2.11 [audit], page 85) to **Sendmail** (see Section 5.1.3 [Exim], page 64): send a mail.

`$ mediatex upload[+] [file` *file* `[as` *target*`]]* [catalog` *file*`] [rules` *file*`] to coll` *coll*
: from **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) or **audit** (see Section 5.2.11 [audit], page 85) to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload an incoming archive into the cache.

`$ mediatex srv extract`
`$ mediatex srv notify`
`$ mediatex make [coll` *coll*`]`
`$ mediatex motd`
: from **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) respectively to CLIENT (see Section 5.3 [Client], page 87), **misc** (see Section 5.3.4 [misc], page 100) and **motd** (see Section 5.3.6 [motd], page 105): scheduled tasks.

`(motd)`
: from **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) to PUBLISHER (see Section 1.1 [Who], page 3): update the (above generated) message of the day.

- Data in or creation
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)
  - `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
  - `/etc/mediatex/mdtx-`*COLL*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)
- Data out

```
/var/run/mediatex/mdtxd.pid
/etc/motd
~mdtx/
~mdtx-COLL/
```

To debug a script, you have to launch it using at least the *MDTX* environment variable:

```
# cd /usr/share/mediatex/scripts
# MDTX_MDTXUSER=mdtx MDTX_LOG_SEVERITY_SCRIPT=debug ./bind.sh
```

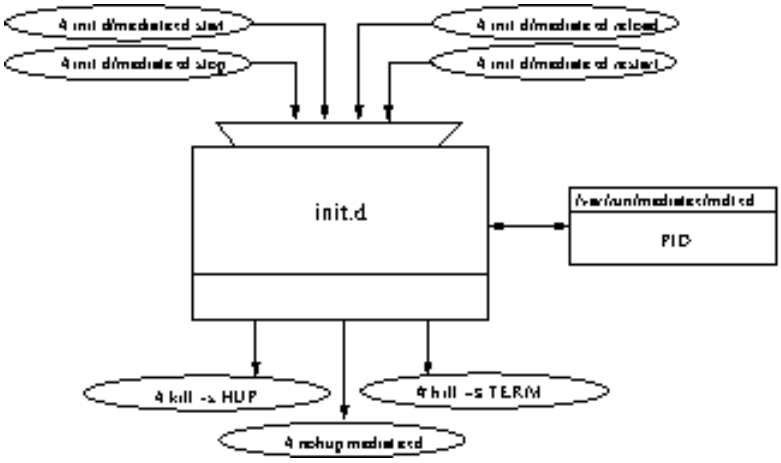
Codes:

```
scripts/lib/git.sh
scripts/lib/htdocs.sh
scripts/lib/include.sh
scripts/lib/jails.sh
scripts/lib/log.sh
scripts/lib/ssh.sh
scripts/lib/users.sh
```

5.2.1 init.d

init.d is the daemon launcher for the SERVER.

Process conceptual model:



- Events in

  ```
  # /etc/init.d/mediatexd start
  # /etc/init.d/mediatexd stop
  # /etc/init.d/mediatexd status
  # /etc/init.d/mediatexd reload
  ```

  from ADMIN (see Section 1.1 [Who], page 3): start/stop the MEDIATEX daemon.
- Processing Manage the SERVER's processus as a daemon.
- Events out

  ```
  # nohup mediatexd
  # kill -s TERM PID
  # kill -s HUP PID
  ```

  to SERVER (see Section 5.4 [Server], page 107): standard UNIX process management
- Data in/out

  ```
  /var/run/mediatex/mdtxd.pid
  ```

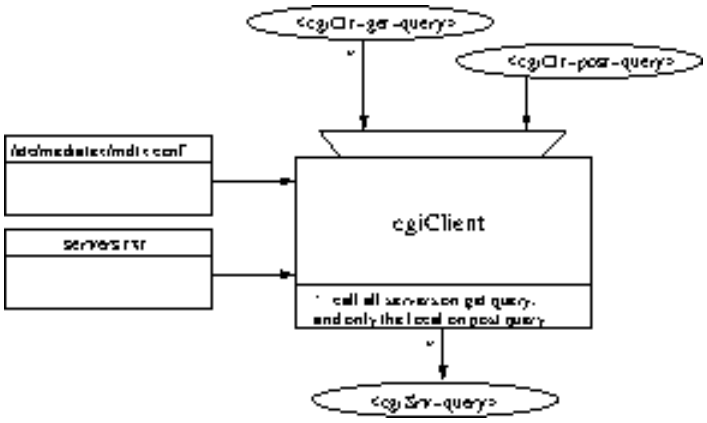
Code:

```
/script/mediatexd.src
```

5.2.2 cgiClient

cgiClient is a frontend that dispatch the USER (see Section 1.1 [Who], page 3) queries around all the remote collection's SERVER (see Section 5.4 [Server], page 107).

Process conceptual model:



- Events in

  `<cgiClt-get-query>`
  `<cgiClt-post-query>`
  from **Apache** (see Section 5.1.2 [Apache], page 60)

- Processing
  - Relays USER's queries comming from Apache to the SERVERs:
    - On GET query, connects all SERVERs until it gets a positive answer.
    - On POST query, only connects the local SERVER.
  - Interprets the SERVERs answers and return a related HTML page or redirection that Apache will relay to the USER's internet browser.

- Events out

  `<cgiSrv-query>`
  to **cgiServer** (see Section 5.4.3 [cgiServer], page 115)

- **Notice:** the standard output of this scripts result on 2 events sent by **Apache** (see Section 5.1.2 [Apache], page 60):

  `<html-form>`
  (if archive is not available) ...

  `<html-redirect>`
  (if archive is available)
  to USER: send back either a form asking for an eMAIL address, or a redirection link to the archive into one server's cache.

- Data in
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)

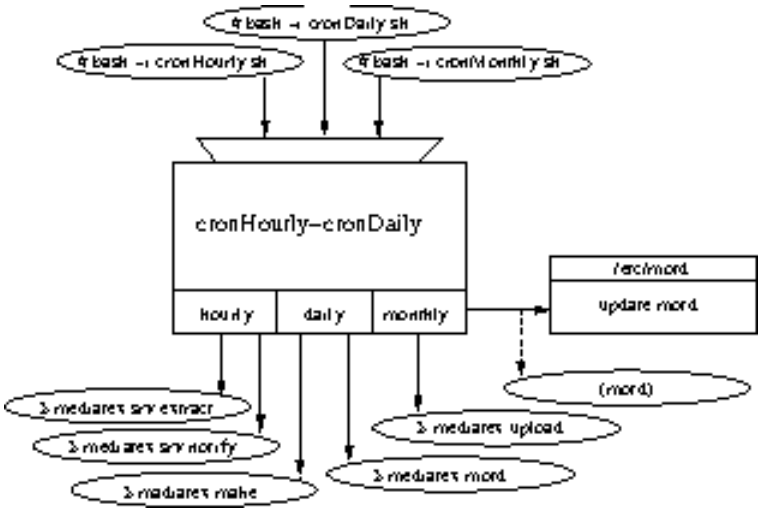
Code:

```
src/common/connect.h
src/common/connect.c
src/cgi.c
```

5.2.3 cronHourly-cronDaily

Theses 3 scripts are run fron **Cron** (see Section 5.1.6 [Cron], page 68).

Process conceptual model:



- Events in

  ```
  # bash -i /usr/share/mediatex/scripts/cron_hourly.sh
  # bash -i /usr/share/mediatex/scripts/cron_daily.sh
  # bash -i /usr/share/mediatex/scripts/cron_monthly.sh
  ```

  from **Cron** (see Section 5.1.6 [Cron], page 68): scheduled tasks.

- Processing, for each collections:
  - Hourly: ask SERVER to extract and notify.
  - Daily: ask CLIENT to build the HTML catalog and the message of the day.
  - Monthly: ask CLIENT to backup the GIT bare local repositories.
- Events out

  ```
  $ mediatex srv extract
  $ mediatex srv notify
  ```

  to CLIENT (see Section 5.3 [Client], page 87): ask server to perform extractions or to communicate its state to other servers.

  `eventClientMiscMake`

  to **misc** (see Section 5.3.4 [misc], page 100): build the local HTML catalogue without GIT synchronisation.

  `$ mediatex upload[+] [file` *`file`* `[as` *`target`*`]]* [catalog` *`file`*`] [rules` *`file`*`] to coll` *`coll`*

  to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload GIT bare modules into the cache so as to backup them, if enabled into `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49).

`$ mediatex motd`
: to **motd** (see Section 5.3.6 [motd], page 105): build the message of the day (actions PUBLISHER have to perform).

`(motd)`
: to PUBLISHER (see Section 1.1 [Who], page 3): store the message of the day into `/etc/motd`.

- Data out

  `/etc/motd`

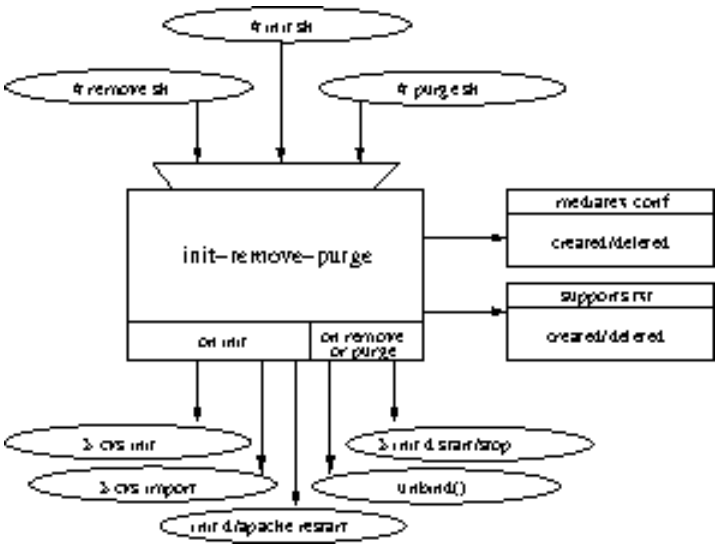
Code:

```
scripts/cron_hourly.src
scripts/cron_daily.src
scripts/cron_monthly.src
```

5.2.4 init-remove-purge

These 3 scripts manage MEDIATEX's installation (install, remove and purge). "`remove.sh`" only uninstall the software but do not touch to the collection's meta-data so as to ease an upgrade by installing a new software version (called by DEBIAN's `postrm` script for instance).

Process conceptual model:



- Events in

  ```
  # /usr/share/mediatex/scripts/init.sh
  # /usr/share/mediatex/scripts/remove.sh
  # /usr/share/mediatex/scripts/purge.sh
  ```
  from CLIENT (see Section 5.3 [Client], page 87)

- Processing
  - Manage `mdtx` system user and 2 system groups:

    `mdtx` server's user and group

`mdtx_md` common group for all collections related to the server

- Install the MEDIATEX repositories

- Events out

`unbind()` to **bind-unbind** (see Section 5.2.8 [bind-unbind], page 83):

```
# /etc/init.d/mediatexd start
# /etc/init.d/mediatexd stop
```
to **init.d** (see Section 5.2.1 [init.d], page 74): start/stop the MEDIATEX daemon.

```
$ git init --bare
$ git config
```
to **Git** (see Section 5.1.5 [Git], page 66): initialise the GIT repository dedicated to the local configuration.

```
eventToolsApacheReload
eventToolsApacheRestart
```
to **Apache** (see Section 5.1.2 [Apache], page 60): make APACHE aware of potential new module and system users this script has configured.

- Data creation
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)
- Data out

```
/var/cache/mediatex/mdtx/jail/etc/group
/var/cache/mediatex/mdtx/jail/etc/password
~mdtx/
```

- Other files involved:

```
/
|- etc/
|  +- mediatex/
|     + mdtx.conf              (symlink -> 3)
|- var/
   +- lib/
    |  +- mediatex/
    |     +- mdtx/                        (1)
    |       +- mdtx/
    |          |- mdtx.conf
    |          +- supports.txt
    +- cache/
       +- mediatex/
          +- mdtx/
             |- cache/                    (2)
             |- git/
             |  +- mdtx/
             |     |- mdtx.conf           (3)
             |     +- supports.txt
```

```
|- jail/
|  +- etc/
|  |  |- group
|  |  +- passwd
|  +- usr/
|  |  +- bin/
|  |     |- git
|  |     +- scp
|  +- var/
|     |- lib        (bind -> 1)
|     |- cache      (bind -> 2)
|- md5sums/
|- public_html
|  |- index.html
|  +- viewvc.cgi
|- tmp/
+- viewvc.conf
```

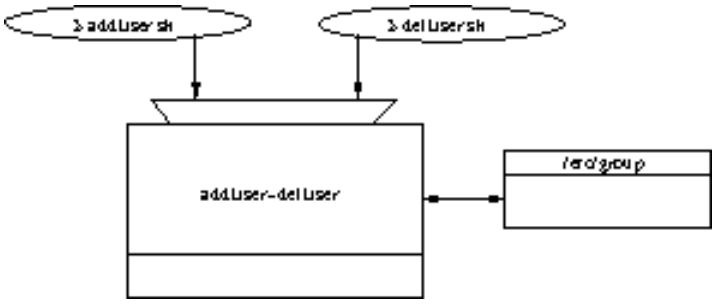
Code:

```
scripts/lib/users.sh
scripts/init.sh
scripts/remove.sh
scripts/purge.sh
```

5.2.5 addUser-delUser

Theses 2 scripts manage PUBLISHER (see Section 1.1 [Who], page 3) users by adding/removing them from the `mdtx` system group and collection's groups too.

Process conceptual model:



- Events in

  ```
  # /usr/share/mediatex/scripts/add-user.sh
  # /usr/share/mediatex/scripts/del-user.sh
  ```

  from CLIENT (see Section 5.3 [Client], page 87)
- Processing

  Manage PUBLISHER users.
- Data in/out

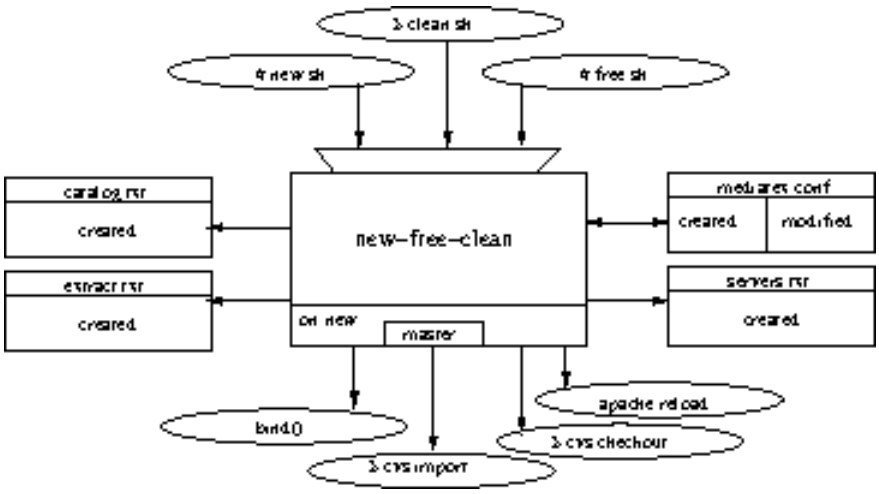
```
/etc/passwd
/etc/group
      mdtx:x:120:www-data,USERNAME1
      mdtx_md:x:123:mdtx,www-data,mdtx-COLL1,mdtx-COLL2,USERNAME1
      mdtx-COLL1:x:124:www-data,mdtx,USERNAME1
      mdtx-COLL2:x:125:www-data,mdtx,USERNAME1
```

Code:

```
scripts/lib/users.sh
scripts/addUser.sh
scripts/delUser.sh
```

5.2.6 new-free-clean

Theses 2 scripts manage collection subscribing or unsubscribing. **Notice:** if local host is the collection master host, theses scripts also manage the creation/destruction of the collection. The third script clean the HTML catalogue.



Process conceptual model:

- Events in

  ```
  # /usr/share/mediatex/scripts/new.sh
  # /usr/share/mediatex/scripts/free.sh
  ```
  from **conf** (see Section 5.3.1 [conf], page 92): manage collection subscribing/unsubscribing.

  ```
  # /usr/share/mediatex/scripts/clean.sh
  ```
  from **misc** (see Section 5.3.4 [misc], page 100): clean the HTML catalogue.

- Processing
  - Manage `mdtx-COLL` collection system user and group.
  - Manage the collection's repositories.
  - Setup the GIT bare module for a new collection or connect to a remote GIT bare module to join an already existing collection.

- Events out

  `bind()` to **bind-unbind** (see Section 5.2.8 [bind-unbind], page 83): make collection repository accessible from the chrooted jail for SSH remote access.

  ```
  $ git init --bare
  $ git config
  $ git clone
  ```
  to **Git** (see Section 5.1.5 [Git], page 66): initialise and/or checkout a GIT collection module.

- Data creation
  - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
  - `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)
- Data out

  `~mdtx-COLL/`
- Other files involved:

```
/
|- etc/
|  +- mediatex/
|     + mdtx-COLL              (symlink -> 1)
|- var/
   +- lib/
   |  +- mediatex/
   |     +- mdtx/
   |       +- mdtx-COLL/
   |          |- apache2/
   |          |- catalog00.txt
   |          |- extract00.txt
   |          +- servers.txt
   +- cache/
      +- mediatex/
         +- mdtx/
            |- cache/
            |  +- mdtx-COLL/
            |- git/
            |  +- mdtx-COLL/             (1)
            |     |- apache2/
            |     |- catalog00.txt
            |     |- extract00.txt
            |     +- servers.txt
            |- jail/
            |  +- var/
            |     |- lib
```

```
|     |  +- mdtx-COLL/
|     |- cache
|        +- mdtx-COLL/
|- md5sums/
|  +- mdtx-COLL.md5
|- tmp/
+- home/
   +- mdtx-COLL/
      |- .ssh/
      +- public_html/
```

Examples:

- Create a new collection: see Section 2.3 [Scenario 2], page 13.
- Join an already existing collection: see Section 2.6 [Scenario 5], page 15.

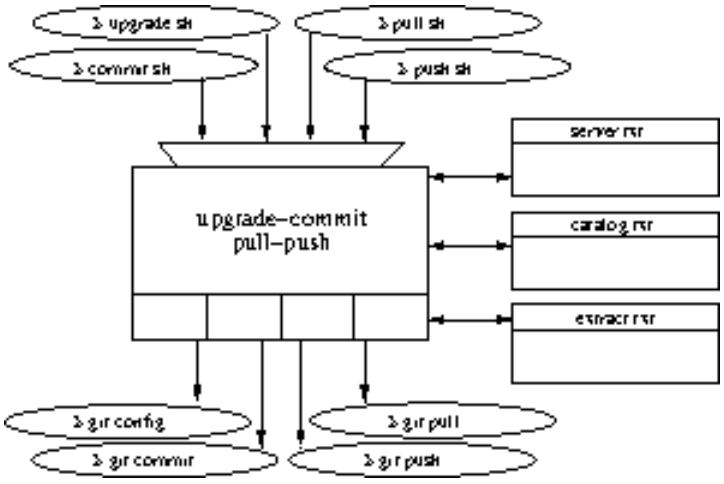
Code:

```
scripts/lib/users.sh
scripts/new.sh
scripts/free.sh
scripts/clean.sh
```

5.2.7 upgrade-commit-pull-push

Theses 2 scripts manage collections update from remote GIT modules. They are also used by CLIENT (see Section 5.3 [Client], page 87) to log changes in the local meta-data.

Process conceptual model:



- Events in

  ```
  $ /usr/share/mediatex/scripts/upgrade.sh
  $ /usr/share/mediatex/scripts/commit.sh
  $ /usr/share/mediatex/scripts/commit.sh
  $ /usr/share/mediatex/scripts/commit.sh
  ```

  from CLIENT (see Section 5.3 [Client], page 87)

`Processing`
: The purpose of this script is to provide an API for CLIENT in order to use the SCRIPTS's BASH library already used during initialisation of configuration and collections.

- Events out

  `$ git config`
  `$ git commit`
  `$ git pull`
  `$ git push`
  : to **Git** (see Section 5.1.5 [Git], page 66): synchronise a GIT module.

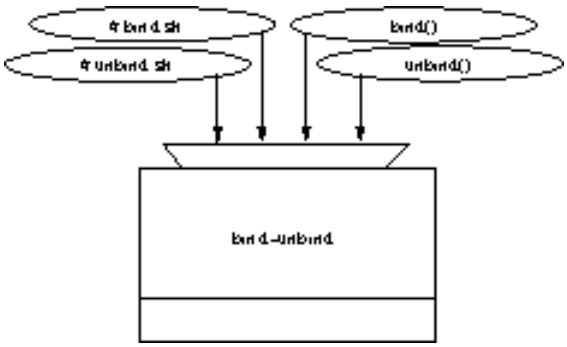
Code:

`scripts/upgrade.sh`
`scripts/commit.sh`
`scripts/pull.sh`
`scripts/push.sh`

5.2.8 bind-unbind

These 2 scripts mount and unmount both gitbare and cache directories into the chroot jail, using the bind option (see `man 8 mount`). They enable or disable remote meta-data upgrades and remote data copies when SERVER (see Section 5.4 [Server], page 107) respectively start or stop.

Process conceptual model:



- Events in

  `# /usr/share/mediatex/scripts/bind.sh`
  `# /usr/share/mediatex/scripts/unbind.sh`
  : from **misc** (see Section 5.3.4 [misc], page 100)

  `bind()` from **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80)

  `unbind()` from **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77)

- Processing: Manage collection repository binding on the chrooted jail for SSH remote access.

  ```
  function JAIL_bind()
  {
  ```

```
        ...
        mount --bind $MDTXHOME/cache $JAIL/var/cache
        mount --bind $GITBARE $JAIL/var/lib/gitbare
        ...
      }
```

see Section 5.2.4 [init-remove-purge], page 77:

```
/
|- var/
   +- lib/
    |  +- mediatex/
    |     +- mdtx/                        (1)
    |       +- mdtx/
    |          |- mdtx.conf,v
    |          +- supports.txt,v
    +- cache/
       +- mediatex/
          +- mdtx/
             |- cache/                    (2)
             +- jail/
                +- var/
                   |- lib        (bind -> 1)
                   |- cache      (bind -> 2)
```

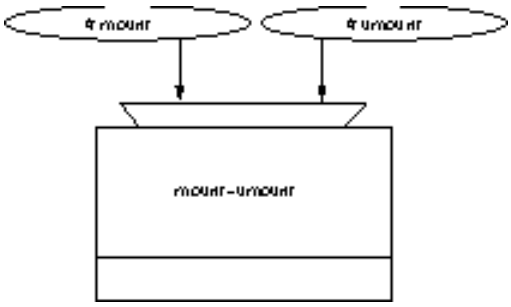
Code:

```
scripts/bind.sh
scripts/unbind.sh
scripts/lib/jail.sh
```

5.2.9 mount-umount

These 2 scripts mount or un-mount an ISO disk or file. (`mount (3)` function and the `loop` device related to `ioctl` may do the same job).

Process conceptual model:



- Events in

  ```
  # /usr/share/mediatex/scripts/mount.sh
  # /usr/share/mediatex/scripts/umount.sh
  ```

  from **supp** (see Section 5.3.2 [supp], page 94)

- Processing

  This script mount both ISO devices or ISO image files.

Code:

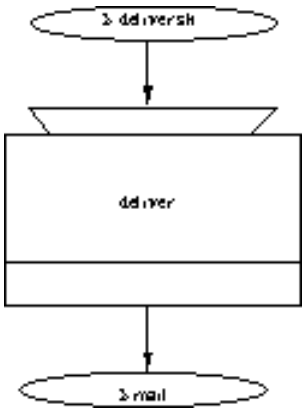
`scripts/mount.sh`
`scripts/umount.sh`

5.2.10 deliver

This script finalise the delivering process, by sending a mail to the User (see Section 1.1 [Who], page 3).

TODO: this script should check if a replacement hook is available for the related collection and use it.

Process conceptual model:



- Events in

  `$ /usr/share/mediatex/scripts/deliver.sh`
  from **delivering** (see Section 5.4.7 [delivering], page 121)

- Processing Use a template to write a notification mail.
- Events out

  `$ mail` to **Sendmail** (see Section 5.1.3 [Exim], page 64): notify archive availability to User.

Code:

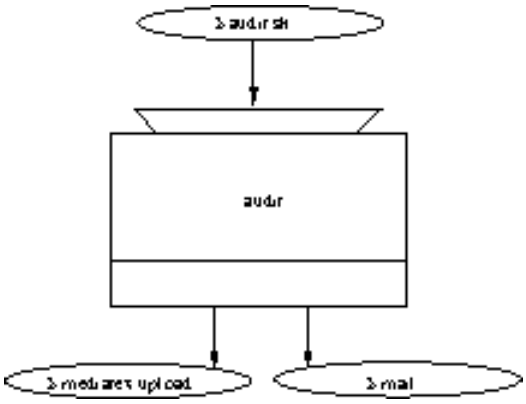
`/usr/share/mediatex/scripts/deliver.sh`

5.2.11 audit

This script manage the build of a collection's audit report.

TODO: still in development.

Process conceptual model:



- Events in

  `$ /usr/share/mediatex/scripts/audit.sh`
  from **delivering** (see Section 5.4.7 [delivering], page 121)

- Processing
  - Emulate the **deliver** (see Section 5.2.10 [deliver], page 85) script: instead of sending a USER notification, notify the archive into an audit report.
  - When all archives involved by the report have raised this script, uploads the report to the collection and send a mail to the PUBLISHER (see Section 1.1 [Who], page 3).

- Events out

  `$ mediatex upload[+] [file` *`file`* `[as` *`target`*`]]* `[catalog` *`file`*`] [rules` *`file`*`] to coll` *`coll`*
  to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload the audit report, if enabled into `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49).

  `$ mail` to **Sendmail** (see Section 5.1.3 [Exim], page 64): notify PUBLISHER the audit report was competed.

- Data in/out

  `~mdtx/tmp/mdtx-COLL/audit_YYYYMMDD-HHmmSS_FINGERPRINT.txt`

Code:

`/usr/share/mediatex/scripts/audit.sh`

5.3 Client

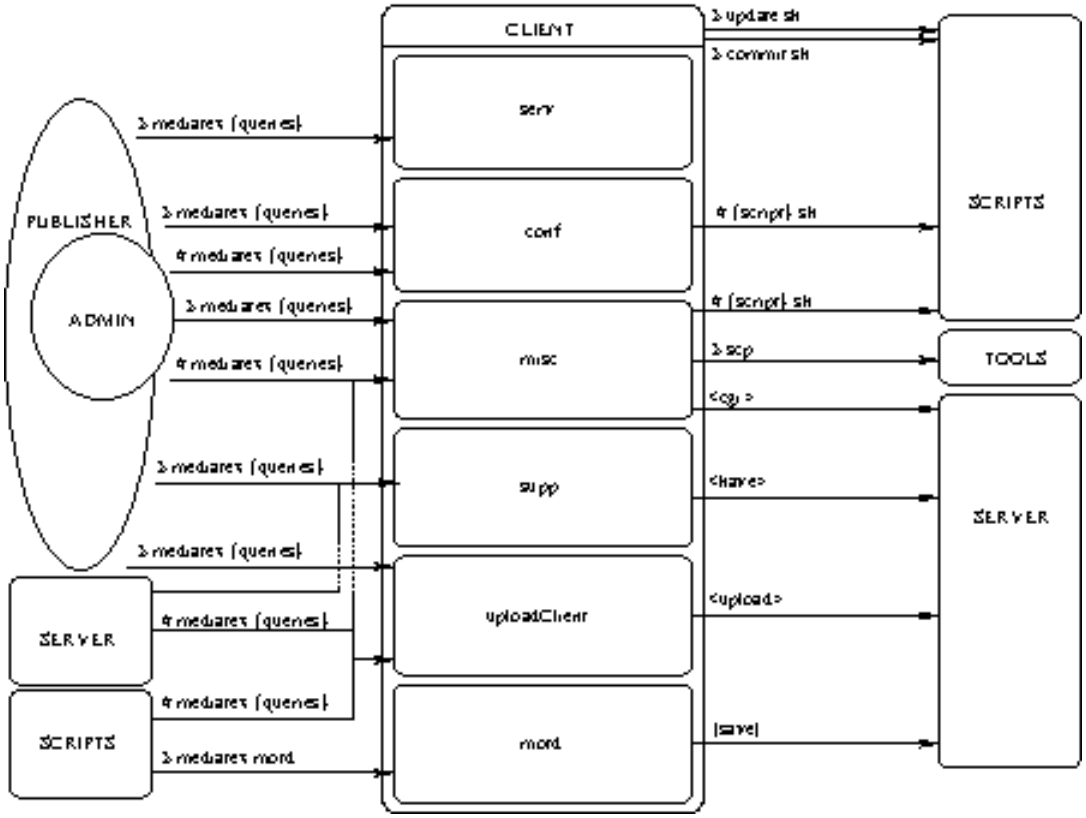
CLIENT is the front-end for the MEDIATEX system. It runs as `setuid root` so as to switch between the different collection contents, but also to do some administrative works such as mounting and binding directories, create new system accounts... Before performing any collection works, CLIENT update the meta-data using **Git** (see Section 5.1.5 [Git], page 66). And before leaving, it commit modifications performed into the configuration and meta-data files.

There is 2 ways CLIENT use to communicate with SERVER:

- Such as **cgiClient** (see Section 5.2.2 [cgiClient], page 75), CLIENT send queries to the SERVER (see Section 5.4 [Server], page 107) over TCP sockets.
- CLIENT also share a virtual register with the SERVER, using share memory and signals.

The CLIENT activity is divided into 6 process:

Flood diagram:



- Events in

```
# mediatex adm init
# mediatex adm remove
# mediatex adm purge
# mediatex adm add user user
# mediatex adm del user user
```

from ADMIN (see Section 1.1 [Who], page 3) to **misc** (see Section 5.3.4 [misc], page 100): configure the MEDIATEX system.

`# mediatex adm add coll [`*`serv-`*`]`*`coll`*`[@`*`host`*`[:`*`port`*`]]`
`# mediatex adm del coll `*`coll`*

from ADMIN to **conf** (see Section 5.3.1 [conf], page 92): configure collections.

`$ mediatex add key `*`file`*` to coll `*`coll`*
`$ mediatex del key `*`fingerprint`*` from coll `*`coll`*
`$ mediatex upgrade[+] [coll `*`coll`*`]`

from PUBLISHER to **serv** (see Section 5.3.3 [serv], page 97): manage collection settings.

`$ mediatex ls [master] coll`
`$ mediatex add supp `*`supp`*` to (all|coll `*`coll`*`)`
`$ mediatex del supp `*`supp`*` from (all|coll `*`coll`*`)`

from PUBLISHER to **conf** (see Section 5.3.1 [conf], page 92): list collections ; share/withdraw local supports to collections.

`$ mediatex ls supp`
`$ mediatex add supp `*`supp`*` on `*`path`*
`$ mediatex add file `*`path`*
`$ mediatex del supp `*`supp`*
`$ mediatex note supp `*`supp`*` as `*`text`*
`$ mediatex check supp `*`supp`*` on `*`file`*

from PUBLISHER (see Section 1.1 [Who], page 3) to **supp** (see Section 5.3.2 [supp], page 94): manage local supports.

`$ mediatex upload[+] [file `*`file`*` [as `*`target`*`]]* [catalog `*`file`*`] [rules `*`file`*`]`
`to coll `*`coll`*

from PUBLISHER, **Syslog** (see Section 5.1.1 [Syslog], page 58), **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) or **audit** (see Section 5.2.11 [audit], page 85) to **uploadClient** (see Section 5.3.5 [uploadClient], page 103): upload an incoming archive into the cache.

`$ mediatex make [coll `*`coll`*`]`
`$ mediatex clean [coll `*`coll`*`]`
`$ mediatex su [coll `*`coll`*`]`
`$ mediatex audit for `*`mail`*` coll `*`coll`*

from PUBLISHER to **misc** (see Section 5.3.4 [misc], page 100): build/clean the HTML catalogue ; change to server or collection system user ; simulate retrieving all archives.

`$ mediatex srv save`
`$ mediatex srv [quick] scan`
`$ mediatex srv trim`
`$ mediatex srv clean`
`$ mediatex srv purge`
`$ mediatex srv status`

from PUBLISHER to SERVER (see Section 5.4 [Server], page 107): repectively ask SERVER (see Section 5.4 [Server], page 107) to dump its state into disk, to manage cache contents or to log its memory status.

`$ mediatex srv extract`
`$ mediatex srv notify`
: from PUBLISHER or **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) to SERVER (see Section 5.4 [Server], page 107): repectively ask SERVER to perform extractions or to communicate its state to other servers.

`$ mediatex motd`
: from **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) to **motd** (see Section 5.3.6 [motd], page 105): build the message of the day (actions PUBLISHER have to perform).

`$ mediatex adm bind`
`$ mediatex adm unbind`
: from SERVER (see Section 5.4 [Server], page 107) to **misc** (see Section 5.3.4 [misc], page 100): manage collection repository binding on the chrooted jail for SSH remote access.

`$ mediatex adm get` *`path`* `as` *`coll`* `on` *`path`*
: from **extract** (see Section 5.4.2 [extract], page 113) to **misc** (see Section 5.3.4 [misc], page 100): retrieve a remote collection's file via SSH.

`$ mediatex adm mount` *`iso`* `on` *`path`*
`$ mediatex adm umount` *`path`*
: from **extract** (see Section 5.4.2 [extract], page 113) to **supp** (see Section 5.3.2 [supp], page 94): manage mounting ISO devices.

- Processing
  - Lock concurent access from any other CLIENT (see Section 5.3 [Client], page 87)'s UNIX process.
  - Catalog and extraction rules meta-data files are:
    - split into 500 ko numbered files, the maximum size VIEWVC advice.
    - automatically re-serialized and upgraded if the last `NN.txt` file was present when loading.
- Events out

`# /usr/share/mediatex/scripts/init.sh`
`# /usr/share/mediatex/scripts/remove.sh`
`# /usr/share/mediatex/scripts/purge.sh`
: from **misc** (see Section 5.3.4 [misc], page 100) to **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77): start/stop the MEDIATEX daemon.

`# /usr/share/mediatex/scripts/new.sh`
`# /usr/share/mediatex/scripts/free.sh`
: from **conf** (see Section 5.3.1 [conf], page 92) to **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): create/subscribe or destroy/unsubscribe a collection.

`# /usr/share/mediatex/scripts/clean.sh`
: from **misc** (see Section 5.3.4 [misc], page 100) to **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): clean the local HTML catalogue.

`# /usr/share/mediatex/scripts/mount.sh`
`# /usr/share/mediatex/scripts/umount.sh`
: from **supp** (see Section 5.3.2 [supp], page 94) to **mount-umount** (see Section 5.2.9 [mount-umount], page 84): manage mounting ISO devices.

`# /usr/share/mediatex/scripts/add-user.sh`
`# /usr/share/mediatex/scripts/del-user.sh`
: from **misc** (see Section 5.3.4 [misc], page 100) to **addUser-delUser** (see Section 5.2.5 [addUser-delUser], page 79): manage PUBLISHER users.

`$ /usr/share/mediatex/scripts/commit.sh`
`$ /usr/share/mediatex/scripts/commit.sh`
: from **serv** (see Section 5.3.3 [serv], page 97) to **upgrade-commit-pull-push** (see Section 5.2.7 [upgrade-commit-pull-push], page 82): manage GIT synchronisation.

`$ scp`
: from **misc** (see Section 5.3.4 [misc], page 100) to **Ssh** (see Section 5.1.4 [Ssh], page 65): copy file from one remote server's cache.

`[save]`
`[extract]`
`[notify]`
: from CLIENT (see Section 5.3 [Client], page 87) to SERVER (see Section 5.4 [Server], page 107): wrap the queries to SERVER.

`<have>`
: from **supp** (see Section 5.3.2 [supp], page 94) to **have** (see Section 5.4.4 [have], page 117): tells SERVER a support is now available.

`<upload>`
: from **misc** (see Section 5.3.4 [misc], page 100) to **cache** (see Section 5.4.1 [cache], page 111): ask SERVER to upload an incoming archive into its cache.

- Data in

`~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)

- Data in/out

`/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
`~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)
`/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
`/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
`/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)

Code:

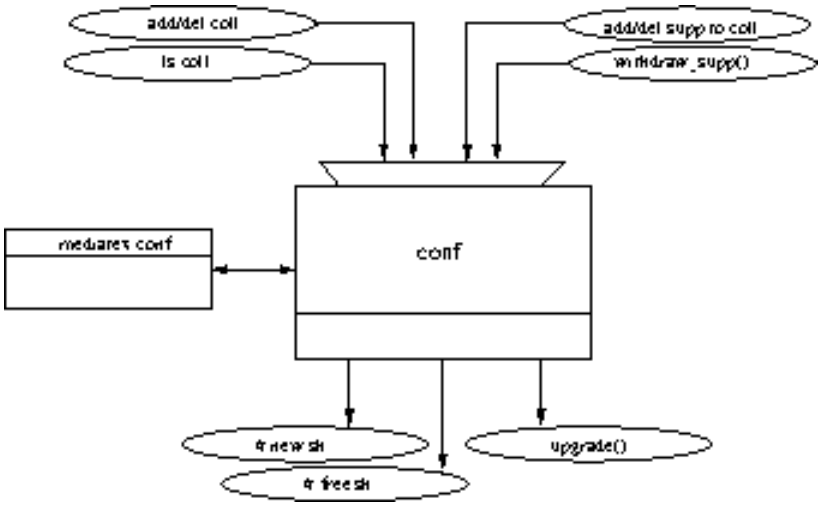
```
src/common/connect.h
src/common/connect.c
src/common/register.h
src/common/register.c
src/client/shellQuery.l
src/client/shellQuery.y
src/client/serv.c
src/mediatex.c
```

5.3.1 conf

The **Conf** process handle queries about collections.

Notice: A single query is used to create a new collection or to join an existing remote collection. Also, a single query is used too to retract from a remote collection or to remove a local collection.

Process conceptual model:



- Events in

  `# mediatex adm add coll [`*`serv-`*`]`*`coll`*`[@`*`host`*`[:`*`port`*`]]`
  `# mediatex adm del coll `*`coll`*

  from ADMIN (see Section 1.1 [Who], page 3): configure collections.

  `$ mediatex ls [master] coll`
  `$ mediatex add supp `*`supp`*` to (all|coll `*`coll`*`)`
  `$ mediatex del supp `*`supp`*` from (all|coll `*`coll`*`)`

  from PUBLISHER (see Section 1.1 [Who], page 3): respectively list collections or share a local support to collections.

  `withdraw_supp()`

  from PUBLISHER (see Section 1.1 [Who], page 3) or **supp** (see Section 5.3.2 [supp], page 94): withdraw a local supports from collections.

- Processing
  - Manage collection (new/join/unsubscribe/remove)
  - Manage sharing or withdrawing suports to/from collections.
- Events out

  `upgrade()`

  to **serv** (see Section 5.3.3 [serv], page 97): upgrade the meta-data.

  `# /usr/share/mediatex/scripts/new.sh`
  `# /usr/share/mediatex/scripts/free.sh`

  to **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): create/subscribe or destroy/unsubscribe a collection.

- Data in/out
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)

Examples:

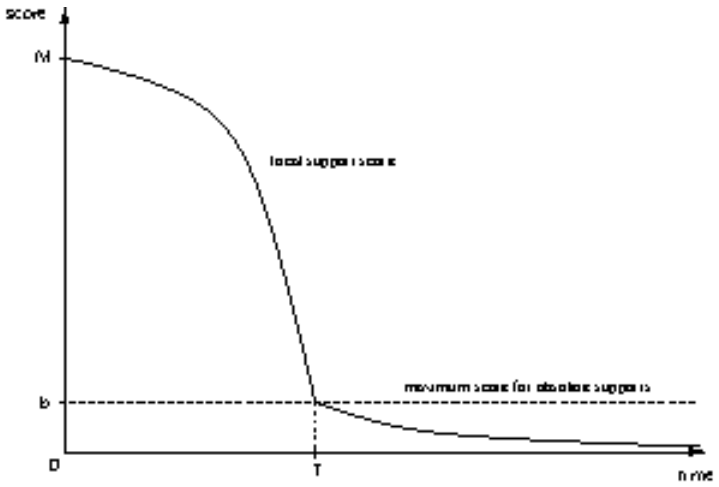
- Create a new collection: see Section 2.3 [Scenario 2], page 13.
- Sharing supports with a collection: see Section 2.4 [Scenario 3], page 14.
- Join an already existing collection: see Section 2.6 [Scenario 5], page 15.

Code:

`src/client/conf.c`

5.3.2 supp

The **Supp** process handle PUBLISHER queries about supports. A score is computed for each local support, based on its age (t). However if support wasn't checked recently (`checkTTL`) its score is set to 0 as it is supposed to be corrupted or lost.



$t <= T : score = (\frac{T-t}{T})^{-a} * (M - b)) + b$
$t > T : score = (1 + c * (t - T)^{-1}) * b$

where: (default values)

$T = 10 years$ is the support time to live (`suppTTL`)

t is the current support age

$a = 2$ power that reduce score from age of supports (`powSupp`)

$M = 10.00$ the maximum score (`maxScore`)

$b = 1.00$ maximum score for outdated supports (`badScore`)

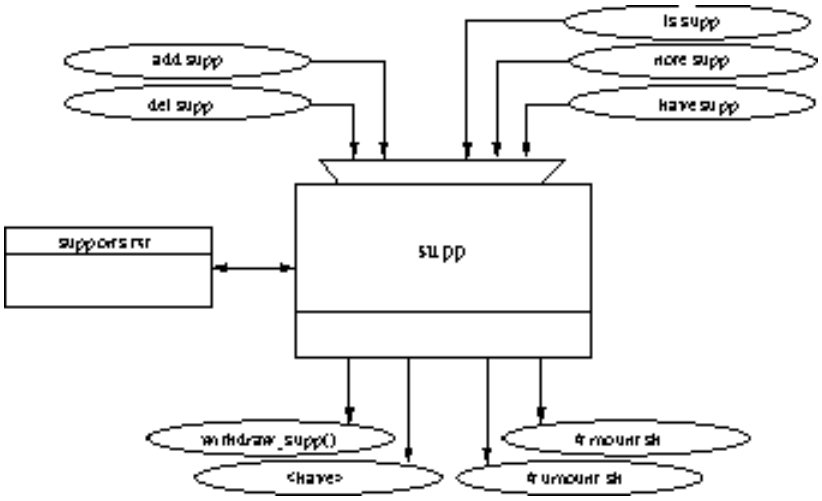
$c = 2$ factor that reduce score of out-dated supports (`factSupp`)

Notice: with the default values this gives:
$t <= T : score = (\sqrt{\frac{T-t}{T}} * (9)) + 1$
$t > T : score = \frac{1}{1+2*(t-T)}$

Notice:

- The same formula apply for local and shared scores (respectively given by `$ mediatex ls supp` or written into `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` file). So theses parameters appears both into `/etc/mediatex/mdtx.conf` and `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt`.
- for support's files, the score is constant and given by the `fileScore` parameter from `/etc/mediatex/mdtx.conf` file was checked recently (`fileTTL`), else score is set to 0 as it is supposed to be deleted from the file-system.

Process conceptual model:



- Events in

  `$ mediatex ls supp`
  `$ mediatex add supp` *`supp`* `on` *`path`*
  `$ mediatex add file` *`path`*
  `$ mediatex del supp` *`supp`*
  `$ mediatex note supp` *`supp`* `as` *`text`*
  `$ mediatex check supp` *`supp`* `on` *`file`*
  from PUBLISHER (see Section 1.1 [Who], page 3): manage local supports.

- Processing
  - Manage supports information related to the `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35) file.
  - Automatically withdraw a support from all collection on removal.
  - When PUBLISHER provides a support, ensure the SERVER (see Section 5.4 [Server], page 107) may work with it.

- Events out

  `withdraw_supp()`
  to **conf** (see Section 5.3.1 [conf], page 92): withdraw a local supports from collections.

  `# /usr/share/mediatex/scripts/mount.sh`
  `# /usr/share/mediatex/scripts/umount.sh`
  to **mount-umount** (see Section 5.2.9 [mount-umount], page 84): manage mounting ISO devices.

  `<have>` to **have** (see Section 5.4.4 [have], page 117): tels SERVER a support is now available.

- Data in/out
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)

- `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49)

Examples:

- Manage supports: see Section 2.2 [Scenario 1], page 12.

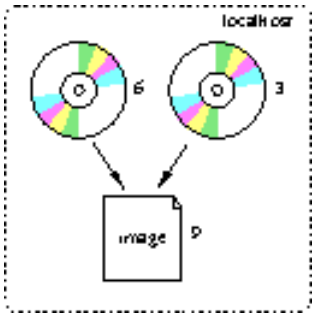
Code:

```
src/common/upgrade.c
src/client/supp.c
```

5.3.3 serv

The **Serv** process maintains the integrity of the meta-data files. Moreover, it manage a perennial score on every archives. For bellow computation, values and parameters are read from the `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37) file.

- A local image's score is updated into `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49) by each servers. It is computed by summing the local support's scores computed by **supp** (see Section 5.3.2 [supp], page 94).



$imageScore(x) = \min\left(\sum_{i=1}^{N} supportScore(x_i), M\right)$

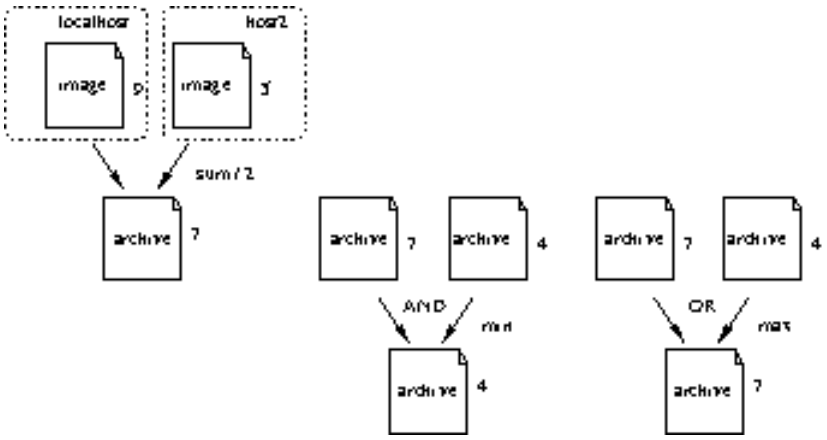
When there are many local supports burned from the same image, this sum may be greater than the maximum score (`maxScore`), so we truncate it.

Notice: if the server's `lastCommit` value is outdated compared to the current date added to `serverTTL` parameter, the image's score if set to 0 for this server as server is supposed crashed.

- The server score equals the worst local image's score on the server.

$serverScore(x) = \min_{i=1..N} imageScore(x_i)$

- An archive score is computed from all local and remote images:



$archiveScore(x) = \min\left(\frac{\sum_{i=1}^{N} imageScore(x_i)}{d}, M\right)$

where: (default value)

$d = 2$ is the minimum number of geographical duplication required (`minGeoDup`)

$M = 10.00$
: is the maximum score (`maxScore`)

When there are more image geographically duplicated than required, this sum may be greater than the maximum score, so we truncate it.

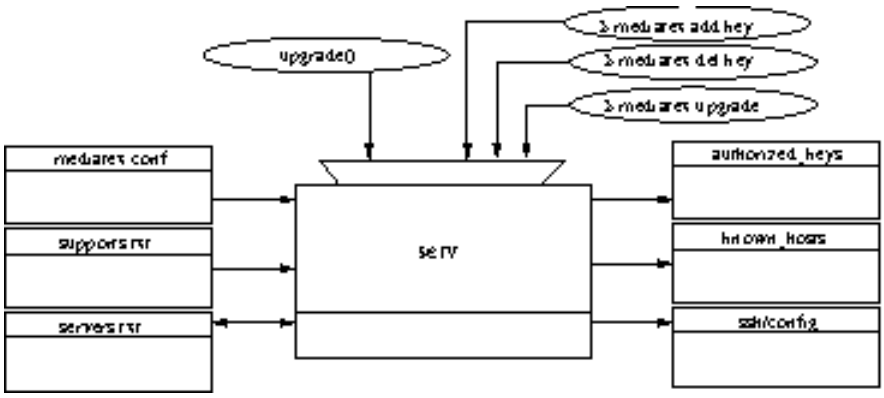
Next, archive's scores are computed recursively:

- using max when archive is provided by several containers,
- or using min when archive comes from combined containers (CAT or RAR).

- Finally, the global collection score equals to the worst archive score.
  $collectionScore = \min_{i=1..N} archiveScore(x_i)$

However, new uploaded archives (since `uploadTTL`) that still have a bad score ($< \frac{M}{2}$) are not taken into account for this global score.

Process conceptual model:



- Events in

  `$ mediatex add key` *`file`* `to coll` *`coll`*
  `$ mediatex del key` *`fingerprint`* `from coll` *`coll`*
  `$ mediatex upgrade[+] [coll` *`coll`*`]`
  : from PUBLISHER (see Section 1.1 [Who], page 3): manage collection settings.

  `upgrade()`
  : from **conf** (see Section 5.3.1 [conf], page 92): upgrade the meta-data.

- Processing
  - The upgrade processing is automatically raised when needed. However it permit to force an upgrade now, which may be useful when modifying manually the configuration files.
  - Adding a key will add a server entry into the `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` file and so, allow connections (subscription, synchronisations, copies) from a new remote server to this collection directory on every servers already added.
- Data in/out
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)

    - `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49)
- Data out

  `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49)
  `~mdtx-COLL/.ssh/authorized_keys`
  `~mdtx-COLL/.ssh/known_host`
  `~mdtx-COLL/.ssh/config`

Examples:

- Join an already existing collection: see Section 2.6 [Scenario 5], page 15.
- Synchronise servers together via upgrade query: see Section 2.5 [Scenario 4], page 14.

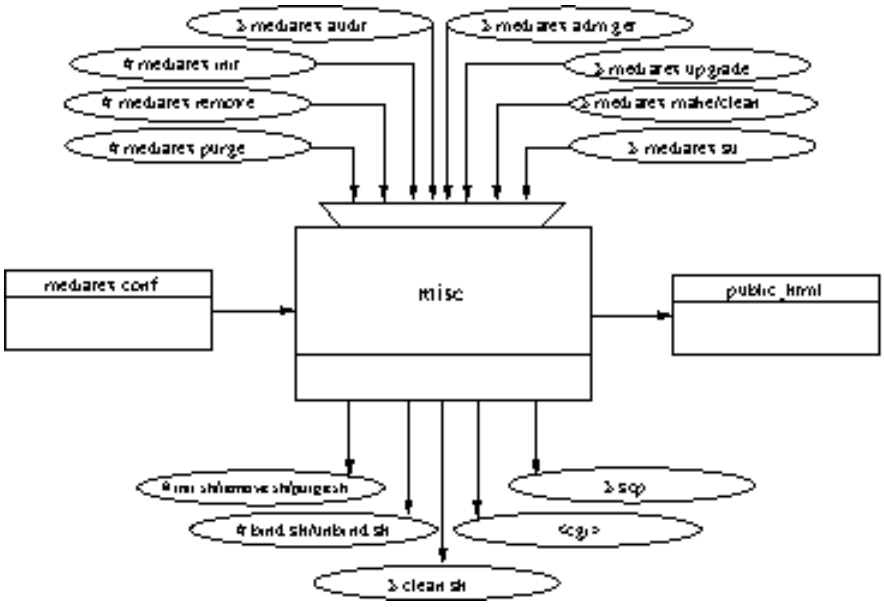
Code:

```
src/common/ssh.c
src/common/upgrade.c
src/common/extractScore.c
src/client/serv.c
```

5.3.4 misc

The **Misc** process handle miscellaneous tasks.

Process conceptual model:



- Events in

  ```
  # mediatex adm init
  # mediatex adm remove
  # mediatex adm purge
  # mediatex adm add user user
  # mediatex adm del user user
  ```
  from ADMIN (see Section 1.1 [Who], page 3): configure the MEDIATEX system.

  ```
  $ mediatex make [coll coll]
  $ mediatex clean [coll coll]
  $ mediatex su [coll coll]
  $ mediatex audit for mail coll coll
  ```
  from PUBLISHER (see Section 1.1 [Who], page 3): build/clean the HTML catalogue ; change to server or collection system user ; simulate retrieving all archives.

  ```
  $ mediatex adm bind
  $ mediatex adm unbind
  ```
  from SERVER (see Section 5.4 [Server], page 107): manage collection repository binding on the chroot-ed jail for SSH remote access.

  ```
  $ mediatex adm get path as coll on path
  ```
  from **extract** (see Section 5.4.2 [extract], page 113): retrieve a remote collection's file via SSH.

- Processings

- Build the HTML catalogues.
- Wraps queries from ADMIN, PUBLISHER and SERVER to SCRIPTS. **Notice:** some operations required the `root` privileges. For security reason, only the CLIENT allows it thanks to its "setuid" bit set.

```
$ ls -l /usr/bin/mediatex
-rwsr-xr-x 1 root root ...
```

- Begin an audit processing.

- Events out

`# /usr/share/mediatex/scripts/init.sh`
`# /usr/share/mediatex/scripts/remove.sh`
`# /usr/share/mediatex/scripts/purge.sh`
: to **init-remove-purge** (see Section 5.2.4 [init-remove-purge], page 77): start/stop the MEDIATEX daemon.

`# /usr/share/mediatex/scripts/clean.sh`
: to **new-free-clean** (see Section 5.2.6 [new-free-clean], page 80): clean the local HTML catalogue.

`# /usr/share/mediatex/scripts/add-user.sh`
`# /usr/share/mediatex/scripts/del-user.sh`
: to **addUser-delUser** (see Section 5.2.5 [addUser-delUser], page 79): manage PUBLISHER users.

`# /usr/share/mediatex/scripts/bind.sh`
`# /usr/share/mediatex/scripts/unbind.sh`
: to **bind-unbind** (see Section 5.2.8 [bind-unbind], page 83): manage collection repository binding on the chrooted jail for SSH remote access.

`<cgiSrv-query>`
: to **cgiServer** (see Section 5.4.3 [cgiServer], page 115): socket query asking for all archives from a collection (for a collection audit).

`$ scp`
: to **Ssh** (see Section 5.1.4 [Ssh], page 65): copy file from one remote server's cache.

- Data in
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)
- Data out

`~mdtx-COLL/public_html/`
`~mdtx/tmp/mdtx-COLL/audit_YYYYMMDD-HHmmSS_FINGERPRINT.txt`

Examples:

- Initialise MEDIATEX: see Section 2.2 [Scenario 1], page 12.
- Build the HTML catalogue: see Section 2.6 [Scenario 5], page 15.

Code:

`src/misc/html.h`
`src/misc/html.c`
`src/client/catalogHtml.c`
`src/client/extractHtml.c`
`src/client/serverHtml.c`
`src/client/misc.c`

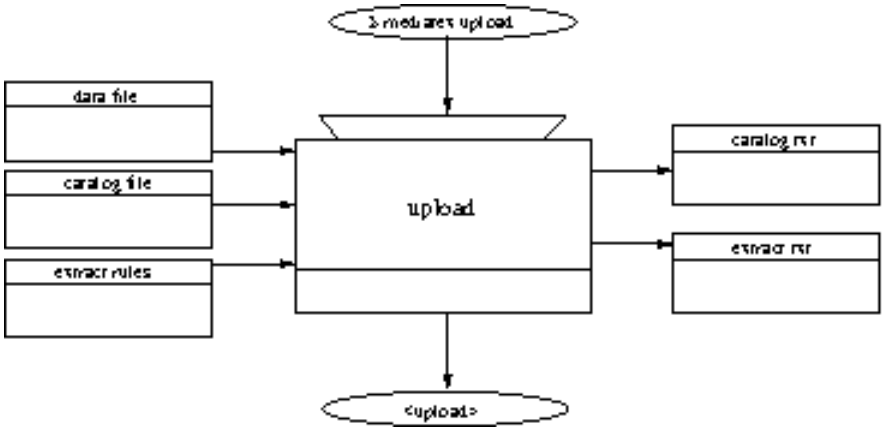
5.3.5 uploadClient

The **Upload** process provides a single API to upload new incoming archives. This query can take 4 parameters:

- descriptive meta-data to concatenate into the `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41) file.
- extraction rules to concatenate into the `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45) file, when the new incoming archive provided is a file container (TGZ...).
- a new incoming archive file
- a target path where to upload the file into the cache.

Checks are made to optimise the meta-data consistency so the provided meta-data files are first parsed into memory, using the same parsers as usual. Because we always assert that an archive description should be linked to an existing archive, we cannot upload here descriptions without a related archive or extraction rules. To do it, you will have to edit or concatenate yourself the `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41) file.

Process conceptual model:



- Events in

  `$ mediatex upload[+] [file` *`file`* `[as` *`target`*`]]* `[catalog` *`file`*`] [rules` *`file`*`] to coll` *`coll`*

  from PUBLISHER (see Section 1.1 [Who], page 3), **Syslog** (see Section 5.1.1 [Syslog], page 58), **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76) or **audit** (see Section 5.2.11 [audit], page 85): upload an incoming archive into the cache.

- Processing
  - parse the provided meta-data,
  - check archive is not already into the current extraction rules,
  - Add an entry to `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45) file, into the dedicated `INC` container and associated with current date,

    - ask daemon to upload the incoming archive files.
    - concatenate new meta-data into both `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41) and `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45) files,
    - ask daemon to reload,
- Events out

  `<upload>` to **uploadServer** (see Section 5.4.5 [uploadServer], page 118): ask SERVER to upload an incoming archive into its cache.
- Data in/out
    - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)
- Data out
    - `/etc/mediatex/mdtx-COLL/catalogXXX.txt` (see Section 4.4 [catalog.txt], page 41)

Examples:

- Adding new archives: see Section 2.5 [Scenario 4], page 14.

Code:

`src/client/upload.c`

5.3.6 motd

The **Motd** process display the list of supports the MEDIATEX system wants to check or extract for, and the archives that needs to be put on external supports to be safe.

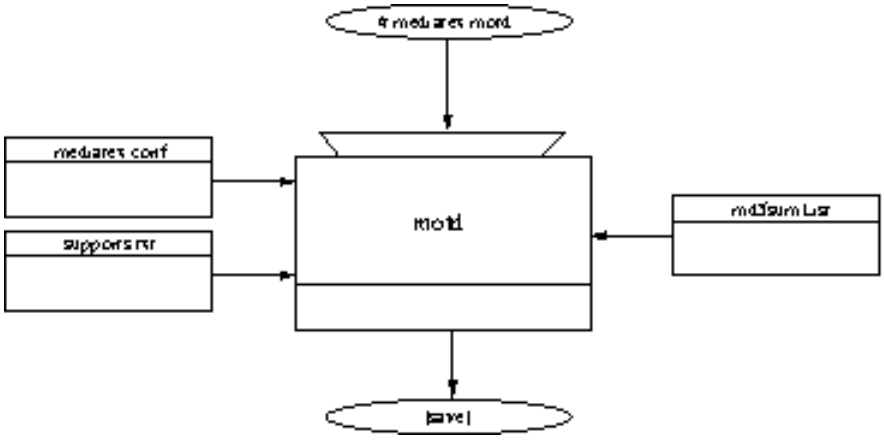
Process conceptual model:

Motd use 3 options defined into the `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31) file:

`checkTTL` The time interval we have to re-check supports before MEDIATEX considers them obsolete (In this case, scores are impacted now ; but you ever can provide the support after and recover your previous scores).

`fileTTL` Twice the time interval MEDIATEX will automatically re-check support's files.

`motdPolicy`
Set to "All" if all images must be retrieve localy. Default is 'Most' to only retrieve localy images having a bad score.



- Events in

  `$ mediatex motd`
  from **conrHourly-cronDaily** (see Section 5.2.3 [cronHourly-cronDaily], page 76)

- Processing

  Ask PUBLISHER for:

  - supports not seen for a while (and so having no score computed),
  - supports needed to extract local or remote remands,
  - cache content to put on external supports.

- Events out

  `[save]` to SERVER (see Section 5.4 [Server], page 107): ask SERVER (see Section 5.4 [Server], page 107) to dump its state into disk.

- Data in

`/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
`~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)
`/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
`~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)

Example:

```
****************************
Mediatex's message of the day
****************************
Please provide theses local supports:
- AAA: periodic check, coll1
- ZZZ: coll1, coll2
Please burn theses files:
- Collection coll1:
 - /var/cache/mediatex/mdtx1/cache/mdtx1-coll1/aaa (-1.00)
 - /var//cache/mediatex/mdtx1/cache/mdtx1-coll1/zzz (-1.00)
Please remove theses local supports:
- /HERE/misc/logo.png: not used
- SUPP11_logo.png: coll1
```

Notice:

- First stanza ask PUBLISHER to provide ($ mediatex check supp *supp* on *file*) supports either because it wasn't checked for too long time or because it is needed in order to extract some content for a collection.
- Second stanza ask PUBLISHER to fix some cache content into a new support ($ mediatex add supp *supp* on *path* or $ mediatex add file *path* followed by $ mediatex add supp *supp* to (all|coll *coll*)).
- Third stanza notify PUBLISHER that some supports may be removed because either it is no more used by any collection or because another safe support already include it.

See also:

- Extract an archive: see Section 2.4 [Scenario 3], page 14.
- Adding new archives: see Section 2.5 [Scenario 4], page 14.
- Add a second server: see Section 2.6 [Scenario 5], page 15.

Code:

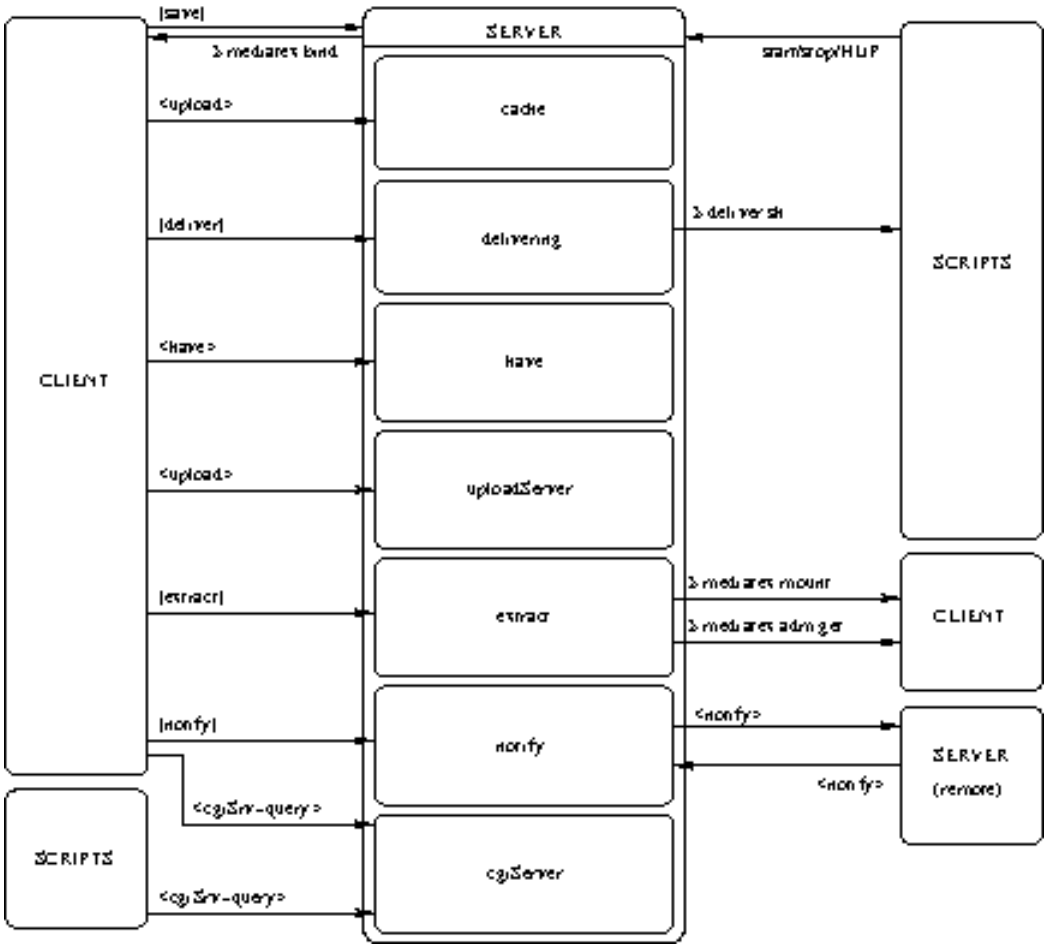
`src/client/motd.c`

5.4 Server

The SERVER activity manage local cache integrity and exchange files with other MEDIATEX's SERVERS. CLIENT (see Section 5.3 [Client], page 87) may ask it to updates the `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37) file that gives the local cache's status.

The SERVER activity is divided into 7 process:

Flood diagram:



- Events in

  ```
  # nohup mediatexd
  # kill -s TERM PID
  # kill -s HUP PID
  ```

  from **init.d** (see Section 5.2.1 [init.d], page 74): standard UNIX process management (HUP signal for reload...).

  `[save]` from **misc** (see Section 5.3.4 [misc], page 100): serialize the last cache status into a file.

`[extract]`
`[notify]` from CLIENT respectively to **extract** (see Section 5.4.2 [extract], page 113), **notify** (see Section 5.4.6 [notify], page 119): ask server to extract or notify locally available archive contents to other servers.

`[quick-scan]`
`[scan]` from CLIENT: ask server to check recorded local supplies are still available from the cache and match recorded size ([quick-scan]) or both recorded md5sum and size ([scan]). Next, scan the cache directory and remind new local-supplies founded, or remove files if they do not match an extraction rule.

`[trim]` from CLIENT: remove form cache all files that are safe and extractable from a containers available into the cache.

`[clean]` from CLIENT: remove form cache all files that are safe and extractable from supports locally available.

`[purge]` from CLIENT: remove form cache all files that are safe.

`[status]` from CLIENT: ask server print into the logs its memory status.

`<cgiSrv-query>`
from **cgiClient** (see Section 5.2.2 [cgiClient], page 75) to **cgiServer** (see Section 5.4.3 [cgiServer], page 115): receive a socket query asking for an archive.

`<have>` from **supp** (see Section 5.3.2 [supp], page 94) to **have** (see Section 5.4.4 [have], page 117): receive a socket query providing archives from a support.

`<upload>` from **misc** (see Section 5.3.4 [misc], page 100) to **cache** (see Section 5.4.1 [cache], page 111): receive a socket query providing an archive and asking to handle it into the cache.

`<notify>` from a remote SERVER to **notify** (see Section 5.4.6 [notify], page 119): receive a socket query providing remote cache content index.

- Processing
  - Serialise cache index into `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtx-COLL.md5], page 37), on [extract]query. Remote supplies and demands are now readable by CLIENT, or on next SERVER (see Section 5.4 [Server], page 107) start-up.
  - Re-index the cache when receiving HUP signal:
    - wait until all threads end,
    - lock concurent threads setup
    - serialise cache index into `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37),
    - free all memory used,
    - parse `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31) file configuration,
    - parse `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37),

- scan the cache directory
- unlock concurent threads setup

- Events out

`$ mediatex adm bind`
`$ mediatex adm unbind`
`$ mediatex adm get` *`path`* `as` *`coll`* `on` *`path`*
: to **misc** (see Section 5.3.4 [misc], page 100): tells client to make operations that needs privileges, like binding directories into the chroot jail or using collection account to retrieve archives remotely.

`$ mediatex adm mount` *`iso`* `on` *`path`*
`$ mediatex adm umount` *`path`*
: from **extract** (see Section 5.4.2 [extract], page 113) to **supp** (see Section 5.3.2 [supp], page 94): tel client to mount supports as it need root privileges.

`<notify>`
: from **notify** (see Section 5.4.6 [notify], page 119) to a remote SERVER: send a socket query providing local cache content index.

`$ /usr/share/mediatex/scripts/deliver.sh`
`$ /usr/share/mediatex/scripts/audit.sh`
: from **delivering** (see Section 5.4.7 [delivering], page 121) to **deliver** (see Section 5.2.10 [deliver], page 85) or **audit** (see Section 5.2.11 [audit], page 85): respectively notify archive availability to USER or audit an archive.

- Data in

`/etc/mediatex/mdtx.conf (see Section 4.1 [mdtx.conf], page 31)`
`~mdtx/git/mdtx/supports.txt (see Section 4.2 [supports.txt], page 35)`
`/etc/mediatex/mdtx-`*`COLL`*`/servers.txt (see Section 4.6 [servers.txt], page 49)`
`/etc/mediatex/mdtx-COLL/extractXXX.txt (see Section 4.5 [extract.txt], page 45)`

- Data in/out

`~mdtx/md5sums/mdtx-COLL.md5 (see Section 4.3 [mdtxCOLL.md5], page 37)`

Return codes on <*socket*> messages:

- status (`X--`)

`1-- ...`
: negative reply

`2-- ...`
: ok

`3-- ...`
: bad request

`4-- ...`
: internal error

- Generic messages (`-0-`)

```
301 message parser error
302 message without collection
303 message from server '%s' not registered into %s collection
304 message contains a record not related to author %s but %s
305 unknown message type: %s
400 internal error
401 fails to read message
402 fails to load %s collection's serverTree
403 fails to get localhost
404 fails to load %s collection's cache
```

- **uploadServer** (see Section 5.4.5 [uploadServer], page 118) messages (`-1-`)

```
210 ok
```

```
310 empty message
332 message do not provide a final supply %s
313 already exists %s:%lli
```

- **cgiServer** (see Section 5.4.3 [cgiServer], page 115) messages (`-2-`)

```
120 not found %s:%lli
220 ok %s/%s
221 ok
```

```
320 empty message
```

- **have** (see Section 5.4.4 [have], page 117) messages (`-3-`)

```
230 ok
```

```
330 empty message
331 message do not provide a final supply %s
```

- **notify** (see Section 5.4.6 [notify], page 119) messages (`-4-`)

```
240 ok
```

Code:

```
src/server/threads.c
src/mediatexd.c
```

5.4.1 cache

Cache process manage the local server cache's directory.

Notice: in order to not delete unsafe archives, **Cache** compute archive's scores using the same algorithm as **serv** (see Section 5.3.3 [serv], page 97) based on image's score provided by `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49).

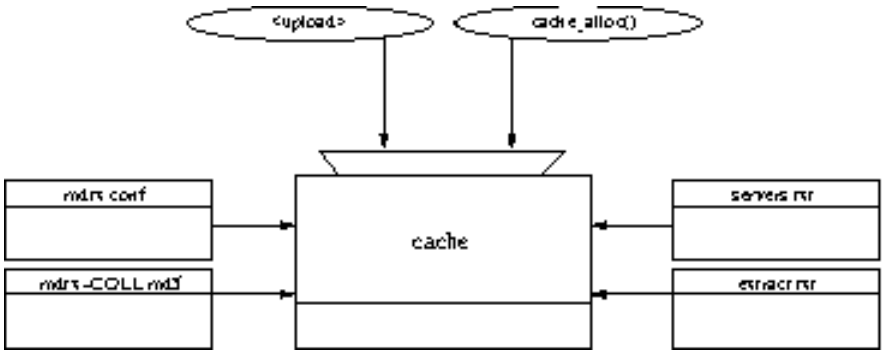
Cache use 3 options defined into the `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31) file:

`cacheSize`
: the maximum size that limit growing operations into the cache

`cacheTTL`
: the minimum time a file notified to USER (see Section 1.1 [Who], page 3) has to be keept into the cache

`queryTTL`
: the time to live of a USER query is kept into the `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37) file

Process conceptual model:



- Events in

  `cache-alloc()`
  : from **extract** (see Section 5.4.2 [extract], page 113): allocate place into the cache.

- Processing
  - Scan the cache directory `~mdtx/cache/mdtx-COLL` without following symlinks.
  - Manage on demand copy of local supports-files and locations into the cache.
  - Compute available and frozen (unsafe archive) spaces.
  - Remove safe archives when more place is needed for allocation.
  - Reserve space for newly allocated archives (during extract or upload)
- Data in
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)
  - `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)

- `~mdtx/git/mdtx/supports.txt` (see Section 4.2 [supports.txt], page 35)

Code:

`src/server/cache.c`

5.4.2 extract

Extract process extracts archives into the local server cache by applying the extraction rules. NAT servers also brings back available files wanted by other severs from its clients.

12 extraction rules are implemented:

- SCP, ISO, CAT: multi OS containers
- TGZ, TBZ, AFIO: GNU/Linux combined containers
- TAR, CPIO, GZ, BZ: GNU/Linux simple containers
- ZIP, RAR: Windows combined containers

`cpio` No compression ; GNU multi-volume archives are not supported.

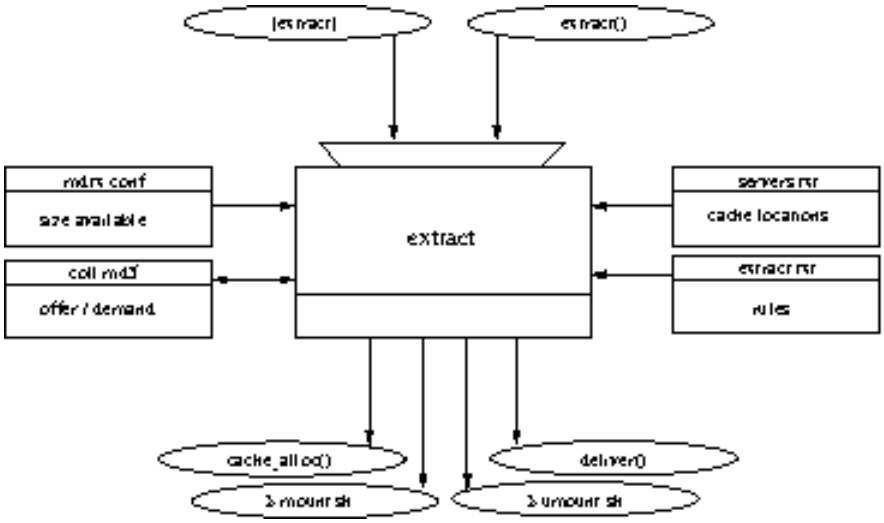
`tar` Multi-volume support is a GNU extension and the archives created in this mode should be read only using GNU tar.

```
$ tar -cM -L1000 -f multi1 -f multi2 -f multi3  data
```

`afio` Compressed archives that are fault tolerant: thanks to the individual files compression, if there is a single byte error you will only loose one or two files.

Notice: The native multi-volumes archives are only handle for RAR. Other containers must be cut using `split` and re-assembled with `cat`.

Process conceptual model:



- Events in

  `[extract]`

  from **misc** (see Section 5.3.4 [misc], page 100): ask to start extraction by setting a value into share memory and sending a signal.

  `extract()`

  from **cgiServer** (see Section 5.4.3 [cgiServer], page 115) or **have** (see Section 5.4.4 [have], page 117): internally ask to start extraction.

- Processing

    - When not call internally, first list all archives that will need to be extracted: all demands and unsafe archives that are not already extracted.
    - Next, when performing an archive extraction:
        - allocate space into the cache
        - extract archive into the temporary extraction directory
        - move the archive into the cache
        - adjust the archive time-to-live so as recursive extraction using this archive will not remove it by allocating new places.
        - send a notification to User (see Section 1.1 [Who], page 3) who was looking for the newly extracted archive.
- Events out

  `cache-alloc()`
  : to **cache** (see Section 5.4.1 [cache], page 111): allocate place into the cache.

  `deliver()`
  : to **delivering** (see Section 5.4.7 [delivering], page 121): deliver mails related to an extracted archive.

  `$ mediatex adm get` *`path`* `as` *`coll`* `on` *`path`*
  : to **misc** (see Section 5.3.4 [misc], page 100): retrieve a remote collection's file via SSH.

  `$ mediatex adm mount` *`iso`* `on` *`path`*
  `$ mediatex adm umount` *`path`*
  : to **supp** (see Section 5.3.2 [supp], page 94): manage mounting ISO devices.
- Input data
    - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
    - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
    - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)

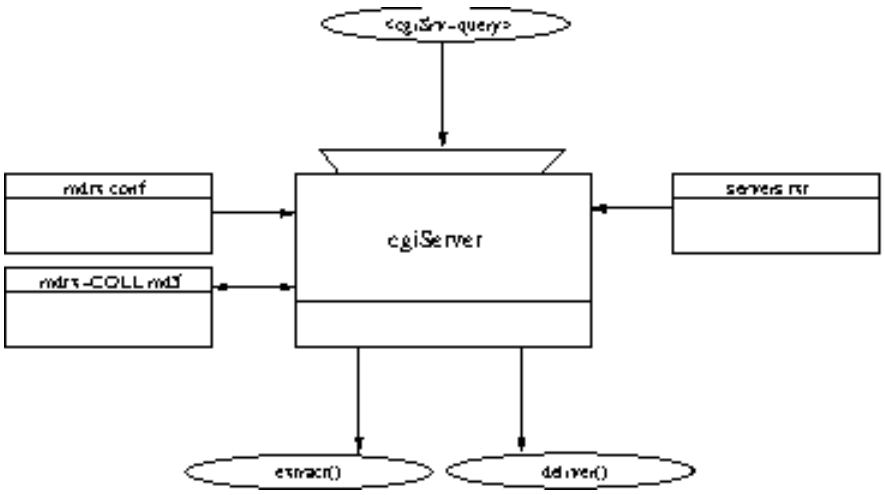
Code:

`src/server/extract.c`

5.4.3 cgiServer

This process handle queries from the cgi script.

Process conceptual model:



- Events in

  `<cgiSrv-query>`
  : from **cgiClient** (see Section 5.2.2 [cgiClient], page 75): receive a socket query asking for an archive.

- Processing

  This process handle 2 kind of queries:

  - When no email address is provided, try to extract archive and:
    - if archive is available on cache returns `220` and its path,
    - if not, returns `120`.
  - When a email address is provided:
    - returns `221` and reminds that a USER (see Section 1.1 [Who], page 3) is looking for the related archive and want to be notified when it becomes available using this email address.

- Events out

  `extract()`
  : to **extract** (see Section 5.4.2 [extract], page 113): internally ask to start extraction.

  `deliver()`
  : to **delivering** (see Section 5.4.7 [delivering], page 121): deliver mails related to an extracted archive.

- Input data
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)

Example:

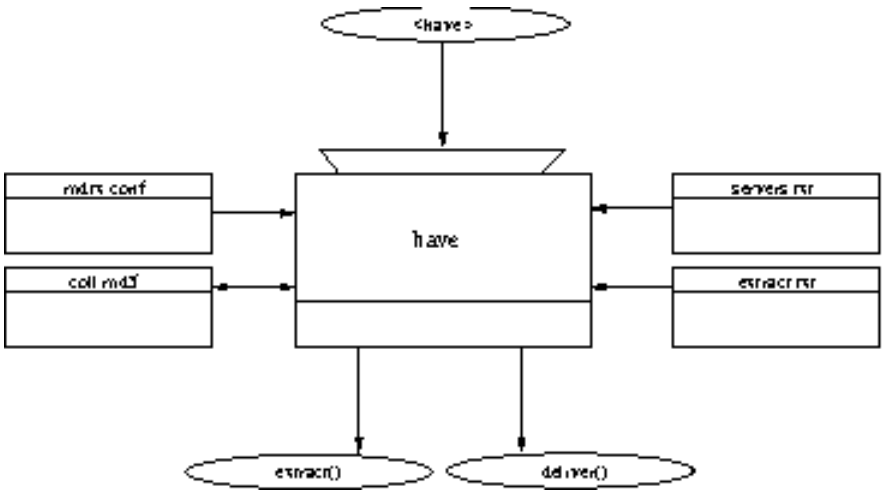
```
$ telnet localhost 6560
Headers
Collection hello
        Server          fbc144f6da753a9a4288e2915df14edd
Type CGI
DoCypher FALSE
Body
#                date                                host ...
D 2013-12-10,10:24:24 fbc144f6da753a9a4288e2915df14edd ...

...                              hash                size extra
... 365acad0657b0113990fe669972a65de               15106 test@test.org
^]quit

# tail -f /var/log/mediatex.log
[info mdtxd.c:384] socketJob
...
[notice mdtxd.c:400] hello: DISK message from \
                    fbc144f6da753a9a4288e2915df14edd server (localhost)
```

Code:

src/server/cgiSrv.c

5.4.4 have

Have process is call when PUBLISHER (see Section 1.1 [Who], page 3) provides a support.
Process conceptual model:



- Events in

  `<have>` from **supp** (see Section 5.3.2 [supp], page 94): receives socket query providing archives from a support.

- Processing
  - add the support as a remote supply into the cache index,
  - recursively extract demands from the support,
  - remove the support's remote supply from the cache index,
  - returns `230`.

- Events out

  `extract()`
  to **extract** (see Section 5.4.2 [extract], page 113): internally ask to start extraction.

  `deliver()`
  to **delivering** (see Section 5.4.7 [delivering], page 121): deliver mails related to an extracted archive.

- Input data
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)
  - `/etc/mediatex/mdtx-COLL/extractXXX.txt` (see Section 4.5 [extract.txt], page 45)

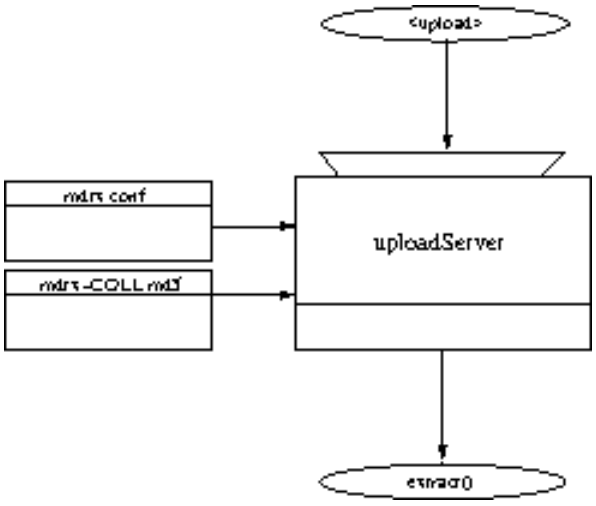
Code:

`src/server/have.c`

5.4.5 uploadServer

uploadServer process manage finalise the upload process into the server's cache.

Process conceptual model:



- Events in

  `<upload>` from **uploadClient** (see Section 5.3.5 [uploadClient], page 103): receives a socket query providing an archive and asking to handle it into the cache.

- Processing
  - if archive is already indexed into the cache returns `313`,
  - add the archive to upload as a remote supply into the cache index,
  - extract demands from the archive, and consequently copy it into the cache directory,
  - remove the upload archive's remote supply from the cache index,
  - returns `210`.
- Events out

  `extract()`
  to **extract** (see Section 5.4.2 [extract], page 113): internally ask to start extraction.
- Data in
  - in: `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - in: `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)

Code:

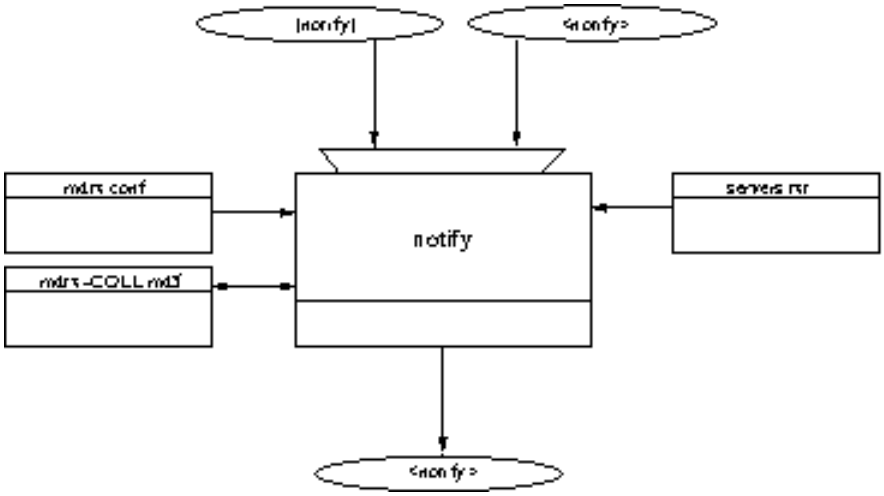
`src/server/upload.c`

5.4.6 notify

Notify process exchanges information on cache supplies and demands with other MEDIATEX servers.

Notice: messages only contains records related to the original sender servers. When receiving a message, server forgives and replace all records previously sends by the related server. Consequently, empty message may be sent too.

Process conceptual model:



- Events in

  `[notify]` from **misc** (see Section 5.3.4 [misc], page 100): ask notify by setting a value into share memory and sending a signal.

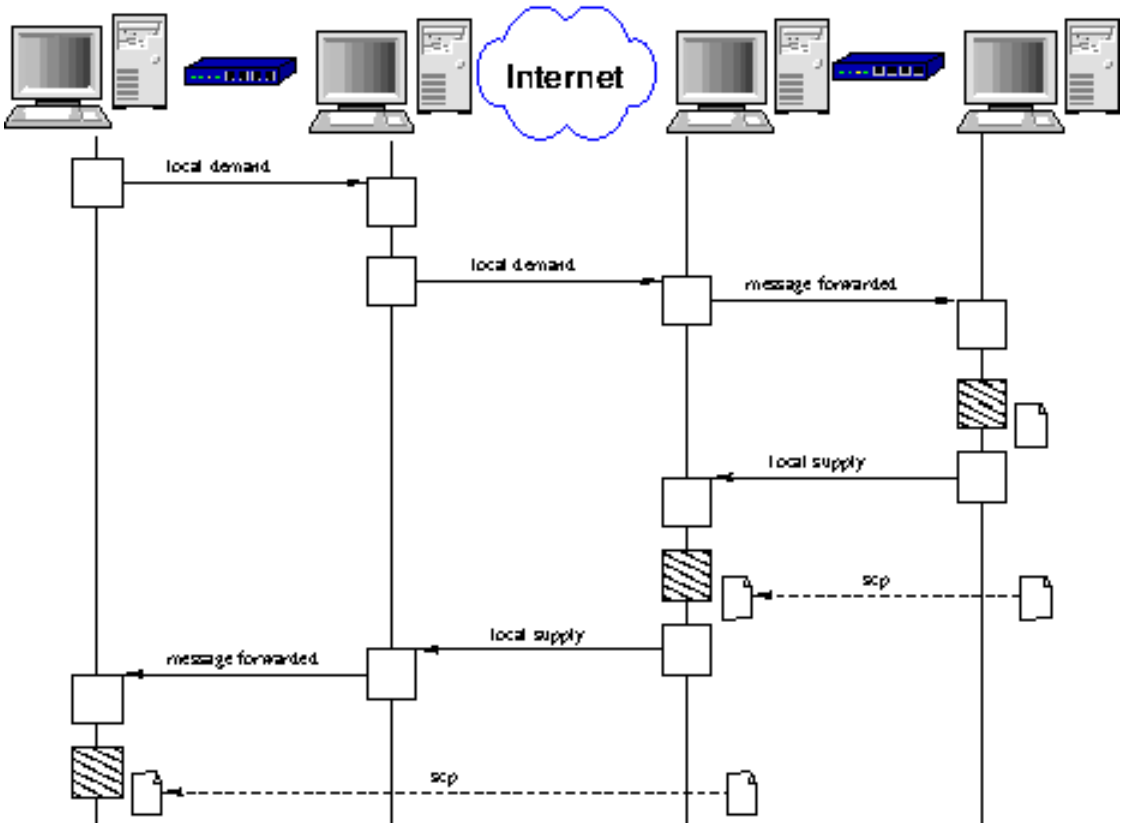
  `<notify>` from a remote **notify** process: receive a socket query providing remote cache content index.

- Processing
  - The policy is, sender servers tells each others:
    - final demands they have to extract,
    - local supplies (presently into theire cache) that may be useful to others or are unsafe.
    - Receiver server returns `240`.
  - However:
    - servers do not try to send messages to unreachable NAT's client servers,
    - NAT client servers do not send messages to servers that cannot reach them.
  - So, NAT gateway servers do:
    - forward messages unchanged to its NAT's client servers,
    - relay (doing masquerade) final demands from its NAT's client servers.
- Events out

  `<notify>` to a remote **notify** process: send a socket query providing local cache content index.

- Input data
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)
  - `~mdtx/md5sums/mdtx-COLL.md5` (see Section 4.3 [mdtxCOLL.md5], page 37)
  - `/etc/mediatex/mdtx-`*`COLL`*`/servers.txt` (see Section 4.6 [servers.txt], page 49)

Example: Servers communication between private networks



Code:

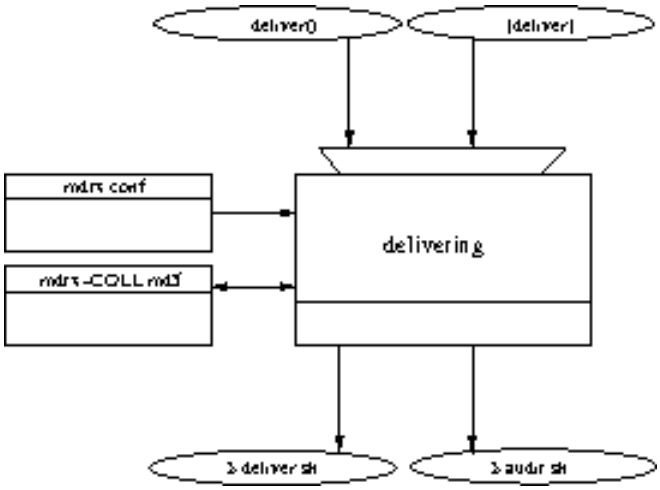
`src/server/notify.c`

5.4.7 delivering

Delivering process call a script in order to warm-up an User (see Section 1.1 [Who], page 3) that an archive he ask for is now available.

However, when this process is hit for an audit, it will call a dedicated script that will summarise results.

Process conceptual model:



- Events in

  `deliver()`
  : from **extract** (see Section 5.4.2 [extract], page 113): deliver an extracted archive.

  `[deliver]`
  : from **misc** (see Section 5.3.4 [misc], page 100): (no more use)

- Processing
  - On normal context, adjust the archive time-to-live to the type of related demands.
  - On audit context, check the archive's md5sum and size.
  - For a final demand, switch to the appropriate delivering script (mail to User or audit).
  - Remove the final demand (email) from the cache index.
- Events out

  `$ /usr/share/mediatex/scripts/deliver.sh`
  : to **deliver** (see Section 5.2.10 [deliver], page 85): tel User (see Section 1.1 [Who], page 3) an archive he was looking for is now available.

  `$ /usr/share/mediatex/scripts/audit.sh`
  : to **audit** (see Section 5.2.11 [audit], page 85): audit an archive.

- Input data
  - `/etc/mediatex/mdtx.conf` (see Section 4.1 [mdtx.conf], page 31)

Code:

`src/server/deliver.c`

6 Examples

MEDIATEX should help building a whole OAIS system.

- It manage redundancy, consistency and unitary access.
- It divides the application complexity as the "Administration" and "Preservation planing" functionality are already taken into account for many features.

`exemples` directory also provides 2 sub-project :

- jukebox: a ISO builder that automate most of the PUBLISHER (see Section 1.1 [Who], page 3)'s job.
- sso: a single-sign-on solution for mediatex.

6.1 Archiving icons

Testing MEDIATEX (see Chapter 5 [Conceptual model], page 53) with about 50,000 archives.

Download the Open Icons Library and build the related catalog and extraction rules files. This last about 15 minutes.

```
$ /usr/share/mediatex/examples/example.sh
```

If not already done, please configure the `mdtx-hello` collection: see Section 2.3 [Scenario 2], page 13.

Grow the cache size by modifying `/etc/mediatex/mdtx.conf`

```
# default cache parameters
cacheSize  1 Go
```

Empty the catalog and extraction meta-data files:

```
$ cat /dev/null > ~mdtx-hello/git/catalog000.txt
$ cat /dev/null > ~mdtx-hello/git/extract000.txt
$ mediatex upgrade+
```

Uploading the icons and build the HTML catalog:

```
$ mediatex upload+ file open_icon_library-standard.tar.gz \
   catalog icons.cat rules icons.ext to coll hello
```

Archive are now injected. You can browses the icons catalog at this URL: `https://localhost/~mdtx-hello`, using `mdtx` login and the password you provide before. Now MEDIATEX (see Chapter 5 [Conceptual model], page 53) should be able to help you conserving or migrating theses archives.

The audit process show bellow is not requiered but only provides a way to double check all goes wright (MEDIATEX (see Chapter 5 [Conceptual model], page 53) already do that by providing a score for collections).

It take time to extract archives from this big TAR container. To speed up the audit process that simulate all archive retrieval, you can first manually extract all archives:

```
# /etc/init.d/mediatexd stop
$ mediatex su
$ cd ~mdtx/cache/mdtx-hello
$ tar -zxf 2016-06/OpenIconsLibrary.tar.gz
# /etc/init.d/mediatexd start
```

The server will take too long time to scan the cache directory when (re-)starting. (This should become optional in further versions). This last about 4 minutes.

Audit the collection:

```
$ mediatex audit coll hello for email_address
$ mediatex srv extract
# tail -f /var/log/mediatex
```

You should be notified by mail that the audit report is uploaded to the collection. The audit process is not optimized and probably will not. This last about 4 hours.

6.2 Configuration behind a firewall

This configuration is needed when you cannot install mediatex on the router of your private network (the bellow configuration cannot serves collections hosted on the true gateway as `ssh/config` file cannot handle 2 ports for the same hostname).

Mediatex must be configured as if it was hosted on the gateway.

`/etc/mediatex/mdtx.conf`:

```
host     GW
mdtxPort 6562
sshPort  2222
wwwPort  4443
```

`iptables` rules for GATEWAY:

```
-t nat -A PREROUTING -p tcp --dport 2222 -j DNAT --to $HOST:22
-t nat -A PREROUTING -p tcp --dport 4443 -j DNAT --to $HOST:443
-t nat -A PREROUTING -p tcp --dport 6562 -j DNAT --to $HOST:6562
-A FORWARD -i $WWW_NET -o $LOCAL_NET -p tcp --dport 22 -j ACCEPT
-A FORWARD -i $WWW_NET -o $LOCAL_NET -p tcp --dport 443 -j ACCEPT
-A FORWARD -i $WWW_NET -o $LOCAL_NET -p tcp --dport 6562 -j ACCEPT
```

Locally if your server own collections, you will have to skip the gateway. (GW must match the gateway's IP address on the WWW_NET interface)

`iptables` rules for HOST:

```
-t nat -A OUTPUT -p tcp -d $GW --dport 2222 -j DNAT --to $HOST:22
-t nat -A OUTPUT -p tcp -d $GW --dport 4443 -j DNAT --to $HOST:443
-t nat -A OUTPUT -p tcp -d $GW --dport 6561 -j DNAT --to $HOST:6561
```

`~mdtx-coll/public_html/.htaccess`

```
# force https
RewriteEngine On
RewriteCond %{SERVER_PORT} !^4443$
RewriteCond %{SERVER_PORT} !^443$
RewriteRule .* https://%{HTTP_HOST}%{REQUEST_URI} [QSA,R=301,L]
```

6.3 Search the meta-data history

GIT allows to quickly search text into the meta-data whole history:

```
$ git log -S 'panthere'
commit <COMMIT_NUMBER>
$ git diff <COMMIT_NUMBER>
```

6.4 Configure HTTP

Default is to use HTTPS for the catalogue's URLs. But HTTP can be used instead : `/etc/mediatex/mdtx-COLL/servers.txt` (see Section 4.6 [servers.txt], page 49)

```
https      no
```

However, both HTTPS and HTTP remain reachable. You can force HTTPS access if needed: `/etc/mediatex/mdtx-COLL/apache2/home.htaccess`

```
# uncomment to force https (no more http available)
SSLOptions +StrictRequire
SSLRequireSSL
```

Default is to protect contents by passwords. You can disable them if no more wanted: `/etc/mediatex/mdtx-COLL/apache2/home.htaccess`

```
# uncomment to disable authentication
SetEnvIf Request_Protocol "^H" NO_AUTH
```

Default is to disallow `www-data` user to lunch MEDIATEXqueries. However allowing him make possible to upload files via a dedicated cgi script shown on "cache" section of the HTML catalogue.

```
# mediatex adm add user www-data
$ mediatex make
```

This extra feature remains protected by password even if authentication is disabled as shown before. The final query lauched by the `www-data` user is configurable into the `/etc/mediatex/mdtx-COLL/put.sh` script.

```
QUERY="mediatex upload+"
```

6.5 Configure DNS

- The following bind9 configuration provides severals IP addresses for a unique domain name.

  `/etc/bind/db.domain-name`

  ```
  ; round-robin
  rr1    IN A IP-1-1
  rr1    IN A IP-1-2
  ...
  rr1    IN A IP-1-N

  dn1 IN CNAME rr1.domain-name.
  ```

  This domaine name is used by MEDIATEX WEB interface. For instance, clicking on the logo will resolve this domain name.

  `~mdtx-coll/git/servers.txt`

  ```
  dnsHost    dn1.domain-name
  ```

- This second bind9 configuration dupplicate the first one in ordr to provide several pool of servers.

  /etc/bind/db.*domain-name*

  ```
  dn1 IN CNAME rr1.domain-name.
  dn2 IN CNAME rr2.domain-name.
  ...
  dnN IN CNAME rrN.domain-name.

  www     IN      CNAME   ns.domain-name.
  ```

  This may be usesefull if a mediatex instance is not enought to handle all collections. The Apache configuration will distribute a uniq main domain (www.*domain-name*) name to several MEDIATEX (see Chapter 5 [Conceptual model], page 53) instances, by maping collection's name to target domain names:

  -
    ```
    $ cat >map.txt <<EOF
    coll1 dn1
    coll2 dn1
    ...
    collX dn2
    ...
    collY dnN
    EOF
    # httxt2dbm -i map.txt -o map.map
    # chown www-data. map.*
    ```

  - /etc/apache2/sites-enabled/000-default.conf

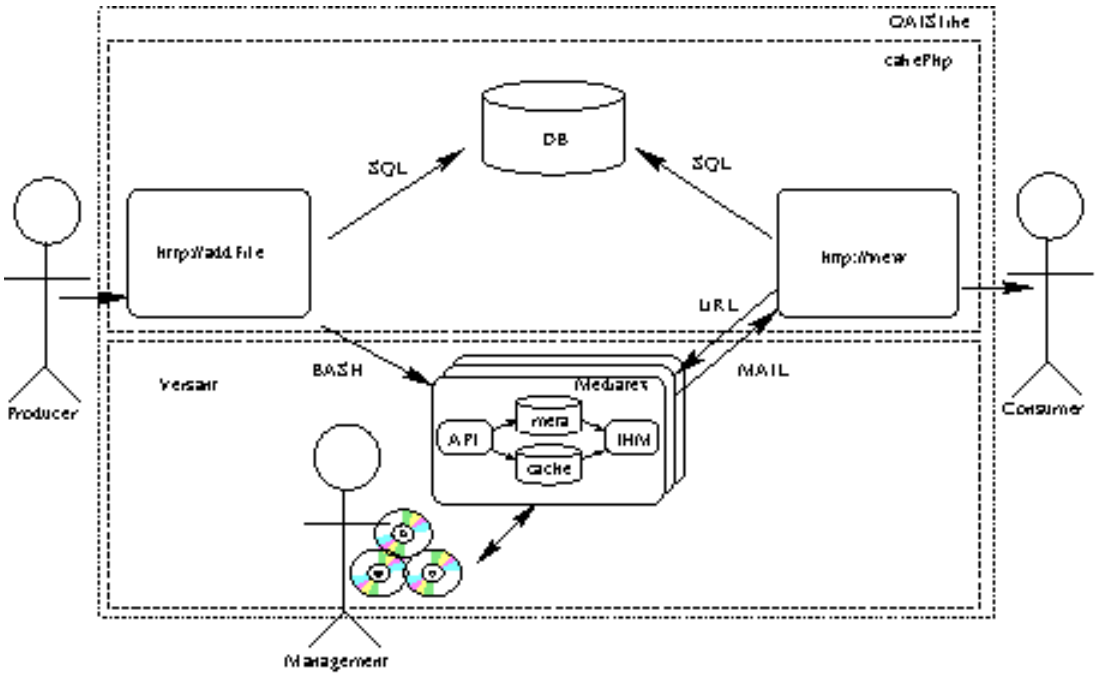
    ```
    RewriteEngine On
    RewriteCond %{HTTP_HOST}   ^www\.(.+)$ [NC]
    RewriteMap dn "dbm:/etc/apache2/map.map"
    RewriteRule "/~mdtx-([^/]*)(.*)" "http://${dn:$1}.%1/~mdtx-$1$2"
    ```

    So URLs like http://www.*domain-name*/~mdtx-*collX*/... will be maped to http://dn2.*domain-name*/~mdtx-*collX*/....

6.6 Minimal Php inerface

A minimal CAKEPHP application is provided with the sources (`/usr/share/mediatex/useCases/cake.tgz`), showing how to use the MEDIATEX system as a storage layer.

Notice: Not tested for a while



There are only 4 interactions from the CAKE's models and view:

`cake/models/user.php`

Add a user

```
function beforeSave() {
  $cmd = "/var/www/cake/ingest.sh user ".
    $this->data['User']['username']." ".
    $this->data['User']['passwd'];
  system($cmd, $retval);
  return ($retval == 0);
}
```

`cake/models/node.php`

Add a directory or a file

```
function beforeSave() {
  $father = $this->find('first',
     array('conditions' => array('Node.id' =>
      $this->data['Node']['node_id'])));

  switch($this->data['Node']['type']) {
  case 1:
    $cmd = "/var/www/cake/ingest.sh dir ".
```

```
                    $this->data['Node']['name']." ".
                    $father['Node']['name'];
                  break;

                case 2:
              ...
                  $cmd = "/var/www/cake/ingest.sh file ".
                    $this->data['Node']['username']." ".
                    $this->data['Node']['name']." ".
                    $father['Node']['name']." ".
                    $path;
                  break;
              ...
                }

                system($cmd, $retval);
                return ($retval == 0);
              }
```

`cake/views/nodes/view.ctp`

CAKE view points on the MEDIATEX URL.

```
              ...
              echo $html->link($fils['name'],
                               'https://HOSTNAME/~mdtx-cake/cgi/get.cgi'.
                               $fils['url']);
              ...
```

The `cake/ingest.sh` manage theses 3 ingestion's actions

```
function addUser()
{
  CYPHER=$(printf "$LOGIN:$COLLECTION:$PASSWD" | md5sum | cut -d' ' -f1)
  LINE="$LOGIN:$COLLECTION:$CYPHER"
  echo $LINE >> $PWD_FILE
  sed $GRP_FILE -i -e "s/^\(.*\)$/\1 $LOGIN/"

  cat >>$CAT_FILE <<EOF
Human    "$LOGIN" ""
EOF
}

function addDir()
{
  cat >>$CAT_FILE <<EOF
top Category "$NAME": "$FATHER"
EOF
  mediatex upgrade coll $COLL
}
```

```
function addFile()
{
  HASH=$(md5sum $TEMP | cut -d' ' -f1)
  SIZE=$(ls -l $TEMP | awk '{print $5}')

  mediatex upload $TEMP to coll $COLL

  cat >>$CAT_FILE <<EOF
Document "$NAME": "$FATHER"
  with "uploader" = "$LOGIN" ""
  $HASH:$SIZE
EOF

  mediatex upgrade coll $COLL
}
```

To be fully operational, this implementation lakes a process to recover the database from MEDIATEX. This is the purpose of the following section.

6.7 Library API

One other possibility is to link with the MEDIATEX's library, using its API to write your own parser, and the mediatex parsers write your own serialiser.

The `/usr/share/mediatex/examples/seda-0.2.tgz` project provide such examples. It import meta-data from an XML file and export them into a SQL script.

Bellow show how the main functions linked with mediatex are:

```
#include "mediatex.h"
#include "client/commonHtml.h"
#include "client/catalogHtml.h"

#include <locale.h>

/*=================================================================
 * Function   : usage
 * Description: Print the usage.
 * Synopsis   : static void usage(char* programName)
 * Input      : programName = the name of the program
 * Output     : N/A
 ================================================================*/
static void
usage(char* programName)
{
  mdtxUsage(programName);
  fprintf(stderr, "\n\t\t-i label");
  mdtxOptions();
  fprintf(stderr, "  ---\n"
  "  -i, --label\t\tcollection's label\n");
```

```
  return;
}

/*====================================================================
 * Function   : main
 * Description: Example using libmediatex
 * Synopsis   : ./text2sql
 * Input      : stdin
 * Output     : stdout
 ====================================================================*/
int
main(int argc, char** argv)
{
  Collection* coll = 0;
  char* label = 0;
  // ---
  int rc = 0;
  int cOption = EOF;
  char* programName = *argv;
  char* options = MDTX_SHORT_OPTIONS"i:";
  struct option longOptions[] = {
    {"label", required_argument, 0, 'i'},
    MDTX_LONG_OPTIONS,
    {0, 0, 0, 0}
  };

  setlocale (LC_ALL, "");
  // so as printf do not write comma in float
  setlocale(LC_NUMERIC, "C");

  // import mdtx environment
  getEnv(&env);

  // parse the command line
  while((cOption = getopt_long(argc, argv, options, longOptions, NULL))
        != EOF) {
    switch(cOption) {

    case 'i':
      label = optarg;
      break;

      GET_MDTX_OPTIONS; // generic options
    }
    if (rc) goto optError;
  }
```

```
  // export mdtx environment
  if (!setEnv(programName, &env)) goto optError;

  /*****************************************************************/
  logEmit(LOG_INFO, "** txt2sql: %s **", label);

  if (!(coll = mdtxGetCollection(label))) goto error;
  if (!loadCollection(coll, SERV|CTLG|EXTR)) goto error;
  // DO YOUR EXPORT HERE
  if (!releaseCollection(coll, SERV|CTLG|EXTR)) goto error;
  /*****************************************************************/

  rc = TRUE;
 error:
  freeConfiguration();
  ENDINGS;
  rc=!rc;
 optError:
  exit(rc);
}
```

7 Administration

MEDIATEX should ease managing server changes.

Some usual cases of typology modifications will follows.

They should induce following connectivity errors:

- Test SSH and GIT connectivity:

```
$ mediatex adm update
[error message]
```

- Test daemon socket's connectivity:

```
$ mediatex srv notify
# tail -f /var/log/mediatex.log
[error message]
```

7.1 GIT conflicts

GIT conflict may happen if 2 PUBLISHER (see Section 1.1 [Who], page 3)s are editing the same meta-data.

We initialise 2 mediatex servers on the same host (we need 2 different ports):

```
# mediatex -c serv1 adm init
# mediatex -c serv2 adm init
# sed --follow-symlinks -i /etc/mediatex/serv1.conf -e "s/6561/6001/"
# sed --follow-symlinks -i /etc/mediatex/serv2.conf -e "s/6561/6002/"
# /etc/init.d/mediatexd-serv1 restart
# /etc/init.d/mediatexd-serv2 restart
```

We initialise a collection shared by the 2 servers:

```
# mediatex -c serv1 adm add coll hello
# mediatex -c serv2 adm add coll serv1-hello@localhost
# mediatex -c serv1 add key ~serv2-hello/.ssh/id_rsa.pub to coll hello
# mediatex -c serv2 adm add coll serv1-hello@localhost
```

Now we edit the same stanza:

```
# vi ~serv1-hello/git/catalog000.txt
<<<
  "another thing?" = "no"
---
  "another thing?" = "yes"
>>>

# vi ~serv2-hello/git/catalog000.txt
<<<
  "another thing?" = "no"
---
  "another thing?" = "maybe"
>>>
```

A conflict will occurs when second server upgrade:

```
# mediatex -c serv1 upgrade
# mediatex -c serv2 upgrade
scanner read unexpected caractere '<'
catalog parser fails on line 19
```

You have to edit that file again to resolve the conflict:

```
<<<<<<< HEAD
  "another thing?" = "maybe"
=======
  "another thing?" = "yes"
>>>>>>> 1d155a18d058093d47cd8649b5cd4d241de0c86f
```

Now upgrade do no more complains:

```
# mediatex -c serv2 upgrade
```

If you need specific permissions you can use "$ mediatex su [coll *coll*]" to become the collection system user.

```
# mediatex -c serv2 su coll hello
$ cd ~/git
$ git status
```

Next, please refer to GIT manual (`man 1 git`).

Cleanup:

```
# mediatex -c serv1 adm purge
# mediatex -c serv2 adm purge
# rm -fr /var/lib/mediatex/serv1
# rm -fr /var/lib/mediatex/serv2
```

7.2 Address's change

In this section we change the IP on `/etc/network/interfaces`.

- If we change the IP address a server is currently, we lost the socket connection from servers (including ouself). The correction consists to update the `host` parameter with the new IP and to upgrade this host. First, using the `-a` parameter in order to skip the update process.

  ```
  $ vi /etc/mediatex/MDTX.conf
  <<<
  host       OLD_IP
  ---
  host       NEW_IP
  >>>
  $ mediatex upgrade -a
  $ mediatex upgrade
  ```

  Other hosts will get the new address as soon as when they will upgrade too.

- If we change IP on master server, we also lost GIT queries from all servers. The correction consists in the same as upper, plus re-subscribing to the collection on every slaves.

  ```
  slave$ mediatex adm add coll MDTX-COLL@NEW_IP
  ```

Notice: By using a hostname (defined into `/etc/hosts` for instance) instead of an IP address for the `host` parameter into `/etc/mediatex/mdtx.conf`, changing the IP has no incidence on MEDIATEX's configuration.

7.3 Socket key's change

If we change the `collKey` value into `/etc/mediatex/mdtx-COLL/servers.txt` on one host, we will loose the socket connection to him and later to others until every hosts have upgraded. No mater we are on master or slaves.

```
$ mediatex upgrade
```

7.4 Collection user key's change

In this section we change the collection's SSH keys pair.

```
# su MDTX-COLL
$ ssh-keygen -t rsa
The key fingerprint is:
NEW_FINGERPRINT
```

Since ssh 7.4 the fingerprint is displayed into sha256 sum. mediatex is still using the md5 sum.

```
$ ssh-keygen -l -Emd5 -f ~MDTX-COLL/.ssh/id_rsa.pub \
   | awk '{print $2'} | cut -d: -f2- | sed -e 's/://g'
NEW_FINGERPRINT
```

This public key is shared and used by servers to identify themselves under the collection context, and to perform remote copies.

- Changing the collection's keys on a slave server make the GIT connection no more available. The new keys need to be send via another media, in the same way we first connect a new server to the collection:

  ```
  (slave)# scp ~/MDTX-COLL/.ssh/id_rsa.pub MASTER:PATH
  (master)$ mediatex add key PATH to coll COLL
  (slave)$ mediatex upgrade
  ```

  Next you should edit `/etc/mediatex/mdtx-COLL/servers.txt` file so as to remove the oldest server stanza (matching *OLD_FINGERPRINT*).

- On the master host you need to update the master fingerprint in two places into the `/etc/mediatex/mdtx-COLL/servers.txt` file.

  ```
  $ vi /etc/mediatex/mdtx-COLL/servers.txt

  <<<
  master OLD_FINGERPRINT
  ...
  Server OLD_FINGERPRINT
  ---
  master NEW_FINGERPRINT
  ...
  Server NEW_FINGERPRINT
  ```

```
>>>
$ mediatex upgrade
```

Both remote copies and socket's connections will fails until the target server automatically upgrade (via CRON) and then get the new public key of the source server.

Local keys are updated into `/etc/mediatex/mdtx.conf` by `$ mediatex upgrade`, but not by `$ mediatex upgrade coll` *COLL*.

7.5 Host key's change

In this section we change the host SSH keys pair.

```
# ssh-keygen -f /etc/ssh/ssh_host_rsa_key
The key fingerprint is:
NEWFINGERPRINT
```

This public host key is shared and matched by remote servers when they connect using ssh to the current host.

- On slave servers, upgrading is sufficient to propagate the new host's public key.
- However, upgrade is not sufficient on the master host and every slaves will have to re-subscribe in order to validate its new fingerprint manually:

```
slave$ mediatex adm add coll MDTX-COLL@MASTER
The authenticity of host 'MASTER' can't be established.
Are you sure you want to continue connecting (yes/no)?
```

Notice: Master host's fingerprint is written into its `/etc/mediatex/mdtx.conf` file or may be obtains using:

```
# ssh-keygen -lf /etc/ssh/ssh_host_rsa_key
```

7.6 Server's change

In this section we move the GIT's bare collection's repository from master host to one other. The same procedure may also be used to backup and restore a collection on the current master host.

First we backup the collection on master host:

```
# tar -C /var/lib/mediatex/MDTX MDTX-COLL -zcf backup.tgz
```

Next we restore it on the host we choose as new master and we re-add the collection so as to force a fresh checkout:

```
# rm -fr /var/lib/mediatex/MDTX/MDTX-COLL
# tar -C /var/lib/mediatex/MDTX/ -zxf backup.tgz
# mediatex adm add coll COLL
```

- If the tarball is extracted on the same host (so as to restore an old state of the collection), it is enough.
- If we move the git bare repository to another host, all hosts will have to re-subscribe :

```
# mediatex adm add coll MDTX-COLL@NEW-MASTER
```

All in all, when "upgrade" fails we can ever re-subscribe to re-synchronise collection.

7.7 GIT recovering

For example, here we recover the `/etc/mediatex/mdtx.conf` file:

```
$ mediatex su
$ cd ~mdtx/git/mdtx
$ git log
$ git revert --no-commit 0766c053..HEAD
$ vi mdtx.conf
$ git commit -a
```

8 Reporting bugs

To report bugs or suggest enhancements for Mediatex, please send electronic mail to nroche@narval.fr.eu.org.

For bug reports, please include enough information for the maintainers to reproduce the problem. Generally speaking, that means:

- The version numbers of Mediatex (which you can find by running '`mediatex --version`') and any other program(s) or manual(s) involved.
- Hardware and operating system names and versions.
- The contents of any input files necessary to reproduce the bug.
- The expected behaviour and/or output.
- A description of the problem and samples of any erroneous output.
- Options you gave to `configure` other than specifying installation directories.
- Anything else that you think would be helpful.

When in doubt whether something is needed or not, include it. It's better to include too much than to leave out something important.

Patches are welcome; if possible, please make them with '`diff -c`'. Please follow the existing coding style.

Tips to debug

- run scripts alone : change directory to `/usr/share/mediatex/scripts` or set the `srcdir` variable. For instance :

```
srcdir=/usr/share/mediatex/scripts MDTX_MDTXUSER=serv1 \
/usr/share/mediatex/scripts/audit.sh hello \
  audit_20160204-002128_395972467d4e582287143a094dd14c10:nroche \
  022a34b2f9b893fba5774237e1aa80ea 24075 0
```

- configure `rsyslog` so as not to drop messages

```
$ cat /etc/rsyslog.conf
$ModLoad imuxsock
$SystemLogRateLimitInterval 0
$SystemLogRateLimitBurst 0
```

- working on source tree

```
# apt-get install libtool-bin
$ libtool --mode=execute valgrind ./BINARY ARGS
$ libtool --mode=execute gdb BINARY
```

- drop setuid bit from mediatex client:

```
# chmod -s /usr/bin/mediatex
# valgrind mediatex ls coll
```

- `gdb` using emacs:

```
# xhost local:mdtx
# su mdtx
$ export DISPLAY=:0.0
```

```
$ emacs23
> M-x gdb
> Run gdb (like this): gdb --annotate=3 mediatexd
(gdb) set args -sinfo -ffile
(gdb) b extractRecord
(gdb) r
```

- `gdb` attachin daemon:

```
$ cat /var/run/mediatex/mdtx.pid
PID
$ gdb
> attach PID
```

- `gdb` using core dump:

```
# echo 1 > /proc/sys/fs/suid_dumpable
# echo '/var/tmp/core.%p' > /proc/sys/kernel/core_pattern
# ulimit -c unlimited               // (into /etc/init.d/mediatexd)

# /etc/init.d/mediatexd start
# kill -SIGABRT PID           // (to simulate coredump)
# su mdtx
$ gdb mediatexd
> core-file /var/tmp/core
> bt
> frame #Number
> set args ...
```

- `valgrind` to detect memory leaks

```
# su mdtx
$ valgrind \
       --leak-check=full --show-reachable=yes --track-origins=yes
       mediatexd
...
# mediatex srv notify
# kill -SIGTERM PID
```

- `gprof`: need to compile using the static library `src/Makefile.am`

```
<<<
mediatex_LDADD = client/libclient.a libmediatex.la
---
mediatex_LDADD = client/libclient.a libmediatex.a
<<<
```

```
$ make CFLAGS="-g -p"
# make install
# mediatex upload
$ ls gmon.out
$ gprof /usr/bin/mediatex > gprof.txt
```

- `kcachegrind`: profiling that include system calls

```
# apt-get install kcachegrind
```

```
# chmod u-s /usr/bin/mediatex
# valgrind --tool=callgrind mediatex upload
$ kcachegrind callgrind.out.nnn
```

- reload the info page into emacs

```
in a new buffer use the M-x org-mode :
info:/home/user/git/mediatex/doc/mediatex.info
info:/usr/share/info/mediatex.info
place cursor on a line and type C-c C-o
```

- investigate memory leaks on setuid binary (valgring refusing)

```
$ QUERY -A -sdebug:alloc 2>&1 | sed 's/:.*]/:/' \
   | cut -d" " -f2,3 | grep malloc | sort | uniq -c
$ QUERY -A -sdebug:alloc 2>&1 | sed 's/:.*]/:/' \
   | cut -d" " -f2,3 | grep free   | sort | uniq -c
```

  Otherwise, you need to drop setuid bit on binary and to run it as root, as seen previously.

- check memory in use

```
$ ps -e -ovsz -orss,args= | sort -b -k1,1n | grep mediatex
```

- `massif` to detect memory hogs

```
$ pmap PID

# su mdtx
$ cd /tmp
$ valgrind --tool=massif mediatexd mediatexd -sdebug
...
# mediatex srv notify
# kill -SIGTERM PID
$ ms_print massif.out.PID | less
```

- `gcov` to display code coverage

```
$ find -type f \( -name "*.gcda" -o -name "*.gcno" \) -delete
$ make clean
$ make CFLAGS="-fprofile-arcs -ftest-coverage ..."
$ make CFLAGS="-fprofile-arcs -ftest-coverage ..." check

# apt-get install gcovr
$ cd src
$ gcovr -r . -e "parser/.*\.c"
TOTAL      14908   11262    75%

$ rm cov.*
$ gcovr -r . -e "parser/.*\.c" --html --html-details -o cov.html
```

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept index

R

S

T

U

X