The File Management System: a highly configurable package for the implementation of automatic flows of files between local and remote machines.

Gianluca Fantino[1], Giancarlo Cerretto[2], Elena Cantoni[3], Fabrizio Pollastri[4]

The File Management System: a highly configurable package for the implementation of automatic flows of files between local and remote machines

T.R. 18/2021 July 2021

I.N.RI.M. TECHNICAL REPORT

Abstract

At the I.N.Ri.M. Time Laboratory, the need to perform an unmanned management of the huge amounts of produced data files, at precisely scheduled times, has become more and more important through the last years. In the present technical report we describe the context that led to the ideation and realization of the File Management System (FMS) package, as well as the detailed explanation of the FMS structure and logic, showing how it can be customizable for the files management needs of indeed any laboratory. Descriptions of each single script and variables definitions and usage also follow.

About the authors: 1 - Formerly at I.N.Ri.M., now at Tim Laboratories. 2 - I.N.Ri.M. researcher. 3 - I.N.Ri.M. researcher. 4 - Formerly at I.N.Ri.M., now retired.

Index

Introduction: the context

The I.N.Ri.M. Time Laboratory, devoted to the generation of the UTC(IT) Italian Standard Time, has the task to disseminate the UTC(IT) signal to users, granting 24/7 availability. From the year 2014 on, the I.N.Ri.M. Time Laboratory undertook a deep process of renewal of the timescale generation process as well as the relative data measurement and management techniques, to ensure this continuous availability at the best level.
The areas of intervention involved every aspect of the laboratory's functionalities, in terms of setting, instruments and software [1], at deep as well as higher level.

To grant the best ambient conditions for the equipment, damping signal instabilities due to temperature sensitivity, the rooms were provided with a new air conditioning system;
the physical dissemination of the UTC(IT) signal to nearby laboratories was realized by means of a new generation of Andrew cables;
to grant continuity for the electrical power supply, this included a new system, with three new Uninterruptible Power Supply (UPS) lines and a new electrical distribution panel with integrated alerting tools.
A long work was devoted to set up a dedicated protected LAN network fitting fast connection performances and high security standards.
But most of all, the automatic signal and data ***generation****,* ***measurement*** *and* ***management*** *processes were completely reviewed, machines renewed and put in redundancy mode, and software* ***completely rewritten****.*

The UTC(IT) Timescale automatic generation algorithm is described in [2]. The continuity of the UTC(IT) physical signal is mandatory, for example, among other tasks, to grant the reference to the Global Navigation Satellites System (GNSS) receivers fleet of the RadioNavigation nearby Laboratory. The receivers measurements (i.e. the time phase UTC(IT) - GPSTime), collected into specifically standardized files, must then be sent on hourly and daily basis to the international centers accounted for clock calculations (i.e. International GNSS Center, IGS) and Timescales comparisons (i.e. Bureau International des Poids et des Mesures, BIPM, as well as other Laboratories). Moreover, GNSS data files constitute the basis for all the calibration services that the Time Laboratory offers to a number of other external entities and enterprises.

About the data measurement system (DMS), a detailed description can be found in [3]. This system provides the time phase data (as well as clocks and environment parameters) in various formats according to the needs.
Each atomic clock, as well as its relative microstepper, participating to the Timescale generation must, in fact, be constantly measured with respect to the reference UTC(IT) signal itself and UTC(IT) – CLOCK time phase data must be continuously present and at disposal for two main reasons:

1) being a prompt reference for the automatic corrections that the UTC(IT) algorithm applies daily to the timescale.
2) Be sent daily (together with the environment data) to the BIPM for the participation to the International Atomic Timescale calculation.

In this context, it is clear how, near the **Timescale system** (providing the signal) and the **DMS system** (providing the data measurements files) a **file management system (FMS)**, accounting for data files 24/7 automatic manipulation, storage, sending and retrieving to and from a number of destinations and sources, is fundamental for the fulfillment of the Time Laboratory tasks.

It can be seen as if the Time Laboratory has three “hearts”, strongly connected and with perfectly coordinated procedures, able to operate in a totally unmanned mode: if one of these “hearts” fails, the Laboratory fails.

The FMS in its context

The Time Laboratory has to deal with a number of tasks that need the presence, in a particular machine, at a specific time, of a clearly identified set of data. A tool is needed that, with an easy, standard, repeatable yet highly customizable procedure, is able to manage data *files* in *space* and *time*, as well as manage their data *format* and *metadata*, assuring *unambiguous identification*, without *any measurement loss or unwanted modification*. There are no commercial tools allowing this level of flexibility, so the FMS design and realization was completely internal.

The step that brought to the FMS concept was the identification of the tasks, as said, to be performed on data, that could be associated with a *data file/files flow*. These tasks are: **data measurement, retrieving, formatting, storage, processing, monitoring, delivering**, and they give rise to the following flows:

1) Data files **storage** in the Laboratory central NAS archive. Files are **incoming** from:
   - *external repositories* (data types: external *products*, e.g. IGS satellites clocks and orbits calculations)
   - RadioNavigation Laboratory *receivers workstations* (data types: *GNSS measurements*)
   - Time Laboratory *processing servers* (data types: *algorithm results* or *monitoring plots*)
   - Time Laboratory data *measurement server* (data types: *UTC(IT)-SIGNAL measurements;* clock and environment *parameters*)

2) Data files **external delivery**. Files are **outgoing** from archive to:
   - *external repositories* (data types: *DMS measurements, environmental parameters, formatted GNSS data* to be communicated outside*)*

3) Data files **internal delivery**. Files going from archive to other internal machines for:
   - *data processing*. Measured data are analyzed with algorithms (e.g. Time Transfer Technique is performed on GNSS data) to obtain *internal products*.
   - *data monitoring*. Internal measurements, parameters and products are plotted and sent to the Laboratory web page on hourly and daily basis to get a continuous check of their trend to permit a near real time identification of anomalies.

The previous data files flows can be summarized in the following schema (Fig. 1).

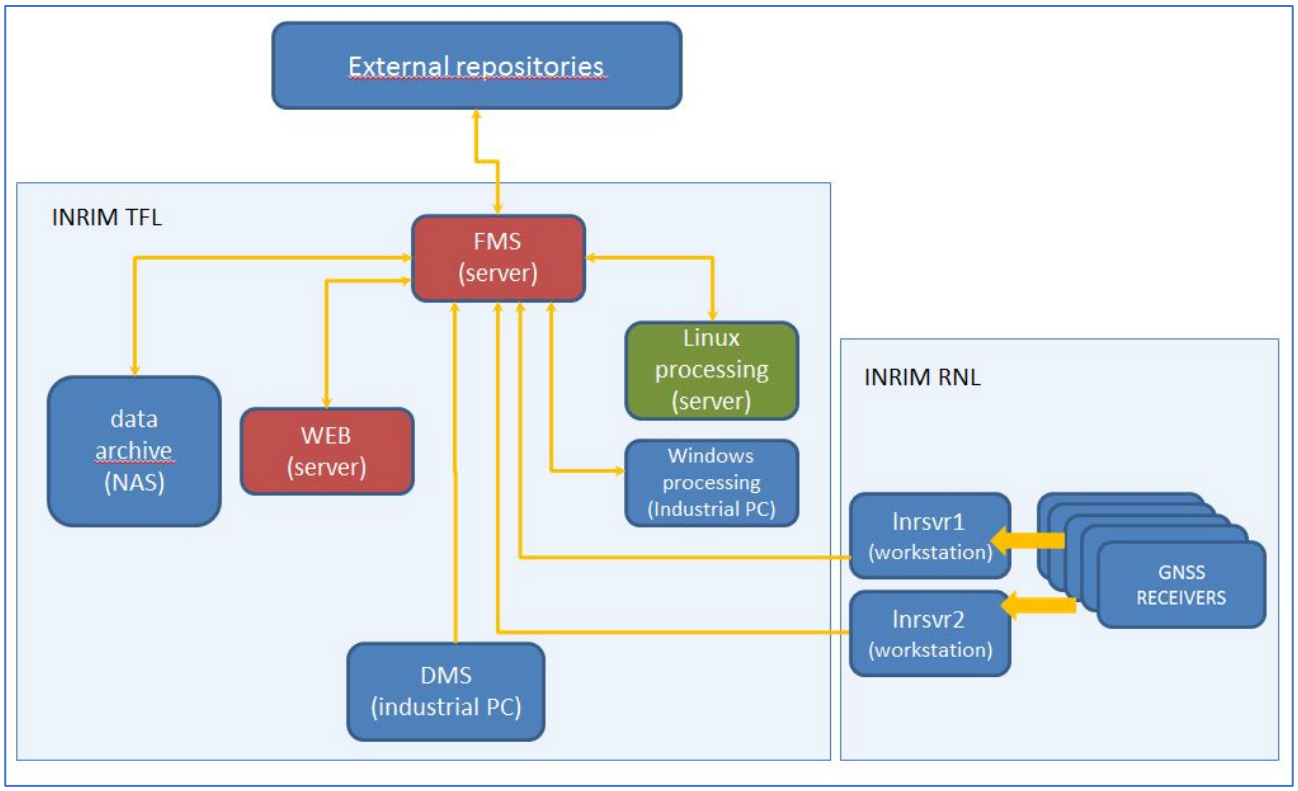

Fig. 1: schematic view of the FMS flows

The FMS scripts then deal well with the complex I.N.Ri.M. Time Laboratory scenario, nevertheless they are designed with a standard that permits a general application, to any reality dealing with the need of copying/moving at scheduled times a large number of files of any kind in any place. So it is possible to sketch the following flow chart to summarize FMS logic (Fig. 2).

It has to be also underlined that not only it is important to have the right files in the right place at the right time, but also the possibility of software and hardware failures has to be minimized. Concerning the software failures, there can be for a number of reasons: these are going to be explained in the next chapter, together with the solutions developed inside FMS to manage “recoveries” (check especially par. 5 and the “Retry” functionality algorithm). About hardware failures, they are prevented by the realization in the Laboratory of the complete hot redundancy of the scheme of Fig. 1 and in particular with the presence of a hot redundant FMS + NAS “cluster”, so that if the nominal platform breaks down, the backup is already aligned and implementable.

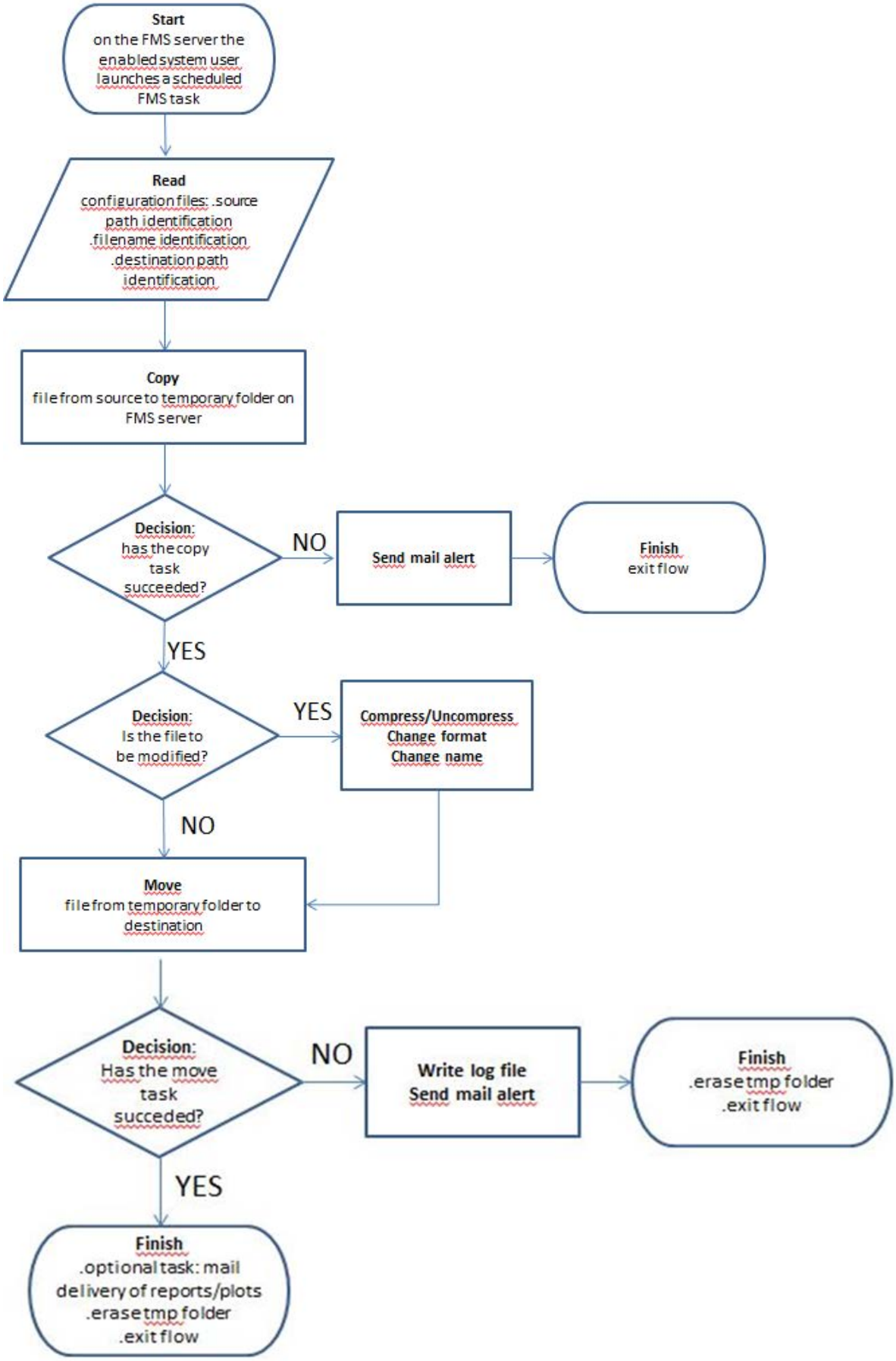

Fig. 2: the FMS base logic.

The FMS usage explained

The File Management System (FMS) is a set of bash scripts, running on Linux platforms, that allows to keep data files coming from heterogeneous sources, organized in a highly customizable folder structure. It works by means of basic logical blocks called “flows” that can be put together to form complex workflows called “lists” with easy-to-fill configuration files and can be work either manually or via scheduled tasks. The files treated by FMS can be of any kind: data files, reports, plots. I.N.Ri.M. FMS is quoted in [1] and the software sources are available at INRIM repositories under request to the authors and come with a MIT license.

1. Folders structure

The FMS package consists of the following folder structure, to be placed in the host machine that will be devoted to the management of the flows processes. In the following examples the /opt folder will be used.

/FMS/bin → Folder containing the bash scripts. A description of the single scripts and their usage can be found in annex B of the present document.

/FMS/etc → contains the configuration files of the flows, in turn organized in subfolders as follows:

/FMS/etc/config → contains the following files:

- **Global** configuration files that define the values of the variables loaded by the main script with a **global** scope: this means that they can be used in the main script and also by the subscripts (see par. 4).
- **Default** configuration files, with the default values for the variables that are called in a single flow and in a list of flows respectively (see annex A).
- Some **specific** configuration files read by peculiar FMS scripts or functions (see par. 8).

/FMS/etc/login → Folder containing the credentials of all the target endpoints the FMS host needs to connect to, each in a separate file. Since at I.N.Ri.M the target data are publicly available these files are not encrypted but if necessary, it should be possible to extend FMS capabilities to implement this feature. *Tip*: They should **be named with some convention that recalls the host location** (e.g. “bipm”) and should have correct permission so that each file can be read only by the FMS service user. In this way given the origin and destination host filenames in the configuration file of a flow, the FMS reads the correspondent access credentials in the respective login files (see par. 3).

/FMS/etc/flows/list → It contains the **list files** of all the flows. Each list can contain one or more **paths to single flow configuration files**. For a detailed description of the flows folder see par. 2.

/FMS/log → Here, at each flow run, a folder is created, named with a unique number, identifying the flow (process identification number PID), in which log messages related uniquely to a particular flow are placed. If any error occurs, the relative log

messages can be sent by email to configurable recipients (see par. 7) in the form of attachments. At the end of the process the folder is deleted.

/FMS/tmp → Folder used as staging area where, at each flow run, a subfolder is created, named with a unique number identifying the flow (process identification number PID), in which the files to be handled are temporarily placed. Here, according to the flow configuration, the file can be modified before delivery. For a list of the possible modifications permitted, see par. 3. At the end of the process the folder is deleted.

2. Running a workflow

The main concept of the FMS is that a list of flows is the **argument** any time an FMS run is launched, in a linux shell, in the following way:

```
/opt/FMS/bin/main2 /opt/FMS/etc/flows/list/list_name
```

Tip: a convention should be established for the list names for easier identification. A possible one could be the following:

list name: <source>_<file type or station name>_<file rate>_2_<DESTINATION>

example: esa_gess_h_2_ARCHIVE
esa: files are being retrieved from some machine hosted at the European Space Agency
gess: the files belong to a GNSS receiver hosted inside a Galileo Experimental Sensor Station
h: the files are retrieved hourly and/or contain one hour of data
2_ARCHIVE: the files are sent to the local laboratory NAS.

In the following picture the appearance of the list in the example can be seen:

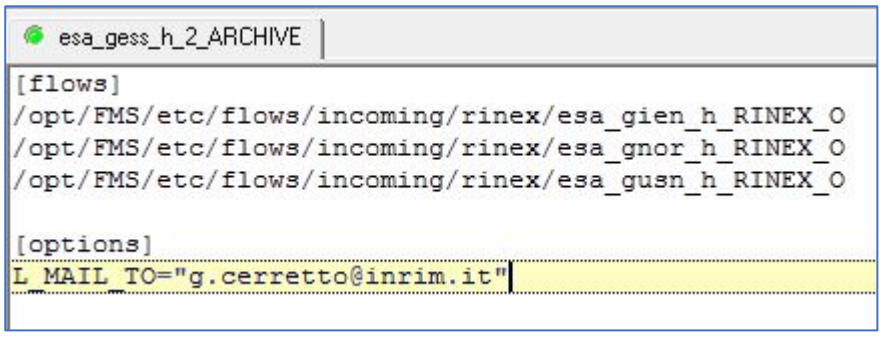

Fig.3: example of list file.

This list file is structured as a .ini file: in the `[flows]` section there are all the actions that need to be performed by FMS in that particular order. Each action or flow is defined by its file name absolute path, which contains the specific configuration to retrieve or send some kind of file.
The number of flows that can be grouped in a single list is potentially unlimited, being each process completely independent. Obviously, the CPU capacity of the server dedicated to

FMS must be taken into account to avoid overloads as long as the time needed to run the list grows with the number of flows defined in the list.

Under the `[options]` section it is possible to override the default values of all the variable related to an FMS run. For example, one can choose to send via mail the results of a certain run to a specific set of mail addresses or even to a single person. The complete list of variables handled by FMS as well as their default values can be found in `/opt/FMS/etc/config/default_list.config` and annex A of the present document.

2.1 “Incoming” and “outgoing” flows concept

Under the `/opt/FMS/etc/flows` folder, files can be organized by the user in such a way to simplify their identification and consultation. For the I.N.Ri.M. Time Laboratory server devoted to data flows, the following philosophy was chosen:

/FMS/etc/flows/incoming → contains the configuration files for data arriving **in the local archive** from external hosts.
/FMS/etc/flows/outgoing → contains the configuration files for all the data that, starting **from the local archive**, are sent to external hosts.

NOTE: a flow of files is considered “incoming” or “outgoing” **with respect to the local archive (NAS)**, not only when a machine is owned by an external entity, but also when the hosting machine is internal to the laboratory.

The “incoming” and “outgoing” folders can be organized dividing them in **subfolders**, named according to the different kind of files to manage (corresponding to different kind of data, following the local and/or general conventions: see Fig. 2). For example, the data flows referring to files in RINEX (Receiver Independent Exchange Format) format can be kept in
/FMS/etc/flows/incoming/rinex

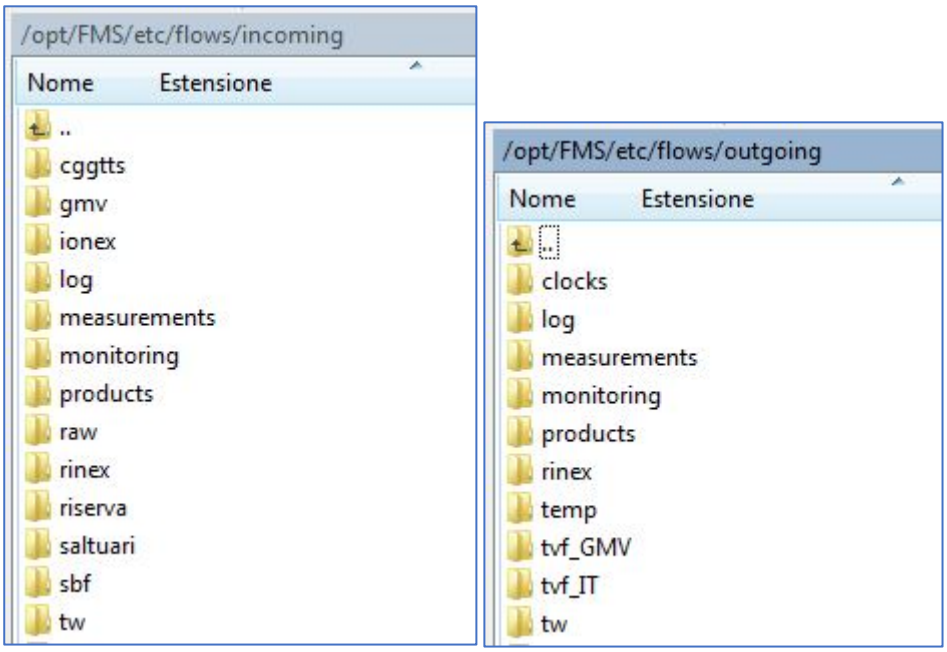

Fig. 4: the “incoming” and “outgoing” folders structure.

3. Single flow file configuration

As in the previous example, let's consider the configuration file ~/FMS/flows/incoming/rinex/esa_gien_h_RINEX_O (Figure 3).
Possibly, a naming convention should be chosen and constantly followed also for the single flow configuration files, for easier reference. In the Time Laboratory the following rule is kept since a long time, but users can choose their favorite:

flow name: <source>_<station name>_<flow rate>_<file format>_<file rate>.

In the example, the "source" is a host at the European Space Agency (ESA), the "station name" is GIEN, the flow has an hourly rate ("h"), the file is in RINEX format, with observation data inside ("O").

In a flow configuration file, variables names starting with "C_*VARNAME*" are handled. The complete list, together with their defaults, is found in `/opt/FMS/etc/config/default.config` and annex A of the present document.

esa_gien_h_RINEX_O

```
C_SOURCE_TYPE=SFTP
C_DEST_TYPE=FTP

C_SOURCE=dsf_esa_3
C_DEST=archive

STATION=GIEN
station=gien

C_OFFSET_DAY=0
C_OFFSET_HOUR=1
C_OFFSET_UTC=1

C_UNZIP=0

C_PATH_SOURCE=/dsf/fr/yyear/mmonth/dday/h%hour%
C_FILENAME_SOURCE=${station}doy%hour_alpha%.yyo.gz

C_PATH_DEST=${G_ROOT_GNSS}/GESS/${STATION}/RINEX/OBS/HOURLY/3.XX/year/month/day
C_FILENAME_DEST=${station}doy%hour_alpha%.yyo.gz
C_COPY_TO_DELIVERY=0
```

Figure 5: example of single flow configuration file.

According to the variable type the assigned value can be 0/1 for Booleans and alphanumeric for Strings. Then, a specific script of the package called by the main script, can interpret the variables to execute a certain action.
Let's comment the configuration variables assignments of Figure 3:
Taking a look to the main variables, we can define the protocol to be used to connect both to source host `C_SOURCE_TYPE` and destination `C_DEST_TYPE`, in this case respectively SFTP (Secure File Transfer Protocol and FTP (File Transfer Protocol).
`C_SOURCE` and `C_DEST` are the names of the files that sit in the `/opt/FMS/etc/login` directory so that using this reference the main script has all the information to contact the

servers. In our example, source host is “dsf_esa_3” an ESA server while the destination is the laboratory NAS “archive”. In particular these files are read by the `/opt/FMS/bin/file_transfer` script which expects that in the login folder such names exist, otherwise an error is thrown. A typical login file content is depicted below:

```
HOST=somehost
USER=someuser
PASSWD=somepassword
```

The variable `C_DEST` points to a login folder file name, here “archive”, keeping the credentials of the local laboratory NAS. The **file_transfer** script reads the source protocol type and the source credentials as well as the destination protocol type and credentials and copies the file from the remote host (here via SFTP) to the local NAS (via FTP). See par. 8 for the script description as well as the list of the managed interface types.

Since flow files are sourced by FMS, custom variables can be declared inside this kind of configuration files and can be used to simplify and make more human readable the file itself. In this case the local variable `STATION=GIEN` is defined and reused multiple times below, concatenated with other strings. This method allows also for a simpler reuse of similar configuration files where managed data types are the same and only some part changes, i.e. the station name.

In most of the managed cases at I.N.Ri.M., the target data files are archived and named according to the timestamps of the data set they contain, so that knowing exactly this time tag is necessary to point to the right file.
Using FMS in automatic mode by means of scheduled tasks using crontab, one can refer to those timestamp using the execution time of the script as reference. For this reason, the variables group `C_OFFSET_*` is very important, constituting **one of the central features of the FMS**: they can be used together or individually to define the time offset, with respect to the execution time of the machine (in days, hours, or with respect to UTC), which the target file data refers to. The `/opt/FMS/bin/Replace` has the task **to compute the timestamp from the time offset and express it in the format needed to access the target file**, both in terms of file path and of file name. The **Replace** script represents time through the definition of a list of **keywords**, that can be invoked when needed to represent: year, month, day, DOY (Day Of Year), MJD (Modified Julian Date), hour, minute, GPS Week and Day of the execution time and/or the target time (see par. 8 for the script description, see annex A for the list of the keywords).

Going on with the description of the configuration variables in the example of Figure n. 3, we find `C_UNZIP=0`, that means the file does not need to be uncompressed. This is already the default value for this variable (set 1 to decompress the source file), and in this case is useless. A redundant declaration in a configuration file is not a problem for the program, except in a special case that will be explained further (see par. 5).

`C_PATH_SOURCE` defines the file path **at the source**. We can see this path is expressed through the keywords mentioned above.
`C_FILENAME_SOURCE` defines the file name **at the source**, with the same method. `C_PATH_DEST`, in the same way, defines the file path **at the destination**. `C_FILENAME_DEST` defines the file name **at the destination**.

`C_COPY_TO_DELIVERY` here is set to “0”. If set to “1” the script saves a copy of the delivered file in the `/archive/deliveries` folder of the NAS, as a check of the completed delivery and as a remind. This is used only for outgoing files as an acknowledgement of the file actually been sent. When someone must accomplish deliveries on a campaign basis with external laboratories, this option can be very useful. When someone is just exchanging files between internal entities continuously for institutional purposes this function is clearly redundant, and it is better not to use it to save space on the NAS.

The complete list of configurable options can be found in annex A, as said. The bash scripts structure makes it very easy for the user to add new options to upgrade the system. An example is shown in annex A.

3.1 Tip

Please note that for each flow that appears in a list corresponds the action of copying or moving one single well determined file, with a specific name. That name usually contains a time stamp that is computed as the time difference between a defined offset with respect to the execution time. **Stars (*)** inside the source file name **CAN** be used to move similar files within a single run. But **CAUTION must be paid** in that case, **do not use the destination file name variable** if you don’t want undesired overwrites, but only the destination path. This allows for multiple files to be copied/moved in a destination folder **without renaming**.

4. Global configurations

The list of the global variables with their default assignments is found in `/home/user/FMS/etc/config/GlobalVariables.config` and annex A of this document. Most of them define aliases for the different archive paths to ease the flows files compiling, as in the example in Fig. 3 where `G_ROOT_GNSS` represents the path to the GNSS data in the NAS. Other global variables are related to the **RETRY functionality of the FMS**, that is described in the next paragraph and constitutes the **other central feature of the FMS,** together with the possibility to easily manage timetags as said before in par. 3.
Finally, it is important to remind that in a flow configuration file, these global variables must be called within the brackets of the symbol ${}.

5. The RETRY functionality

The reasons of a flow failure can be different:

- file is missing at the source
- file name to point to is wrong
- source machine is not responding
- target machine is not responding
- source or target paths are wrong
- communication protocol is wrong

Failed flows are notified by e-mail. The cause of the failure can be investigated through the flow log file, sent in attachment. The FMS email functionality is described in par. 7.

FMS natively has a feature that tries many times a flow that is failed for any reason. This functionality is enabled by default in the list variables configuration file in this way: `L_ENABLE_RETRY=1`.
In the global variables configuration file, `G_RETRY_TIME` defines a basic time interval (in minutes) from which the next flow times are defined (see algorithm explained below), e.g. `G_RETRY_TIME=10`.
If the file's retrieve fails again, the flow is launched repeatedly inside time intervals **exponentially deferred** to avoid the overloading of the FMS hosting machine (see main script, from line 64 to 114).
The maximum number of trials is configurable through the variable `G_MAX_RETRIES` in the global variables configuration file.

The exponential algorithm works as follows:

1. For each execution, the `n_retry` variable (which first value is 1) is updated and the maximum interval in minutes for the next trial is defined as:

$$interval = G\_RETRY\_TIME * 2^{n\_retry}.$$

2. A random number is then extracted with the bash `$RANDOM` function between 0 and 32767.
3. The `$RANDOM` %(modulo) `$interval` operation then gives the **next retry time** in minutes between 0 and `$interval`.
4. The next retry time is turned to a `$next_retry` date variable in the format `%Y%m%d%H%M` using the `$(date)` bash function and the name of a retry list to be used in case of error is then defined as follows:

```
retry_list=${G_RETRY_PATH}/${list_name}.${n_retry}_${origin_date}_${next_retry}.
```

If a flow fails because of one of the reasons listed above and the retry functionality is enabled, the path of the flow's configuration file is saved in **`/opt/FMS/etc/flows/limbo`.**
In crontab, the **/opt/FMS/bin/chkRetry** "check retry" script is scheduled (e.g. every five minutes), so that, when launched, it makes the following check:
if in the limbo folder retry list exists that is identified with a next retry date lower than the current date, then the main script is launched with that retry list as argument (see line 21 to 37 of the script).

The option `L_ENABLE_RETRY=0` in a list disables the retry functionality **for that list**.
Please be aware that the retry feature has a recognized, still unfixed bug: pay attention NOT to write in a list a repetition of the default condition `L_ENABLE_RETRY=1`: this can throw the system in a loop, driving the server to a crash!

6. Scheduled and manual runs

A flows list can be scheduled via crontab or, if needed, can be launched manually. Manual runs are useful for example if one needs to run FMS to copy/move files covering a certain time span. It can be run from terminal in the following way:

```
/opt/FMS/bin/manual /opt/FMS/etc/flows/list/list_name \
                      <from date> \
                       <to date> \
                       <file rate>
```

This is the format for the three arguments:

`<from date>` → YYYYMMDDHHMM, e.g.: 202004010000 (April 1st 2020, at midnight)

`<to date>` → YYYYMMDDHHMM, e.g.: 202004302359 (April 30th 2020, at 23:59)

`<file rate>` → "daily", "hourly", "weekly", "monthly".

In this way, FMS will be executed for the selected list N times starting with reference execution time `<from date>` at steps of `<file rate>` where N is the number of steps to go from starting PIT (Point In Time) to destination PIT with steps large as the file rate.

7 Mail alerting & reporting system

The FMS sends by default the following **emails subjects to configurable recipients** (see "`G_MAIL_TO`" variable in the global variables configuration files, to define recipients that are going to receive every alert, and "`L_MAIL_TO`" variable in the list variables configuration file, to define recipients that are going to receive emails just from that specific list):

Error log: it alerts that a flow has finished correctly. The log file is attached and it is useful to troubleshoot the error origin. Moreover, if the error originated from the source machine, the log file is named "source.log", otherwise is named "dest.log".

Success log: when, during a retry cycle a flow succeeds, an email is sent to the relative list recipients.

Max retries reached: it warns an entire retry cycle has ended, reaching the maximum number of trials without success.

FMS error log: it notifies that one of the flows in the launched list is not valid, maybe it is not existing, or the flow name is incorrect. It notifies the user to verify and try again.

Notification log: $list_name (after $n_retry attempts): for a flow that was expected to contextually deliver a report as email attachment, if the delivery finally succeeds after a certain number of attempts.

WARNING on return value of mail client: if attachments could not be sent due to client or server issue.

WARNING at FMS hosting machine reboot.

FMS mail alerting functionality employs SWAKS SMTP tool, calling it from the main script code. So SWAKS needs to be installed and tested before FMS is used. The command to be tested and that is used in the script has the following structure:

```
swaks --to $G_MAIL_TO --h-From: "$SENDER" --server "$SERVER" --port $PORT --auth
LOGIN --auth-user $MAIL_USER --auth-password $MAIL_PASSWD -tls --header "Subject:
FMS error log" --body $error_log
```

The `$SENDER`, `$SERVER`, `$PORT`, `$MAIL_USER`, `$MAIL_PASSWD` variables must be

configured in `/home/user/FMS/etc/config/MailSwaks.config` and they, in order, refer to: the email address to be set as sender of the FMS emails, the SMTP server endpoint of your organization, its port and the service email account used by FMS or any account of your organization, that will be used to authenticate to the SMTP server for automatic email sending.

If a **notification email** must be associated to a certain flow (e.g. if a report or a plot has to be sent to one or more recipients while saved into NAS) the dedicated list variables must be configured (see annex A).

8 Web interface for high level identification of flows (FMS Flows Reporting Tool)

An interactive localhost/fms/fmstof.html web page lives on the FMS hosting machine, giving access to the **FMS flows reporting tool** ((c) F. Pollastri). This tool gives the possibility to perform a methodical search through the FMS data flows configuration files, easing their identification also for a non-operator/developer user (rather than entering the low level configuration files folders and perform a manual search in alphabetical order through the *hundreds of configuration files* FMS manages during typical periods of full commitment…). A picture of the web page is shown below (Figure 5).

A browse can be done visualizing the main flow features of the configuration files (the same features described along the present report), combining them together as desired through the selection/de-selection of the correspondent grey buttons (see Fig. 5). A search by *keywords* can be done in parallel in the `Search` window on the right. More than one keyword can be inserted, blank space separated.

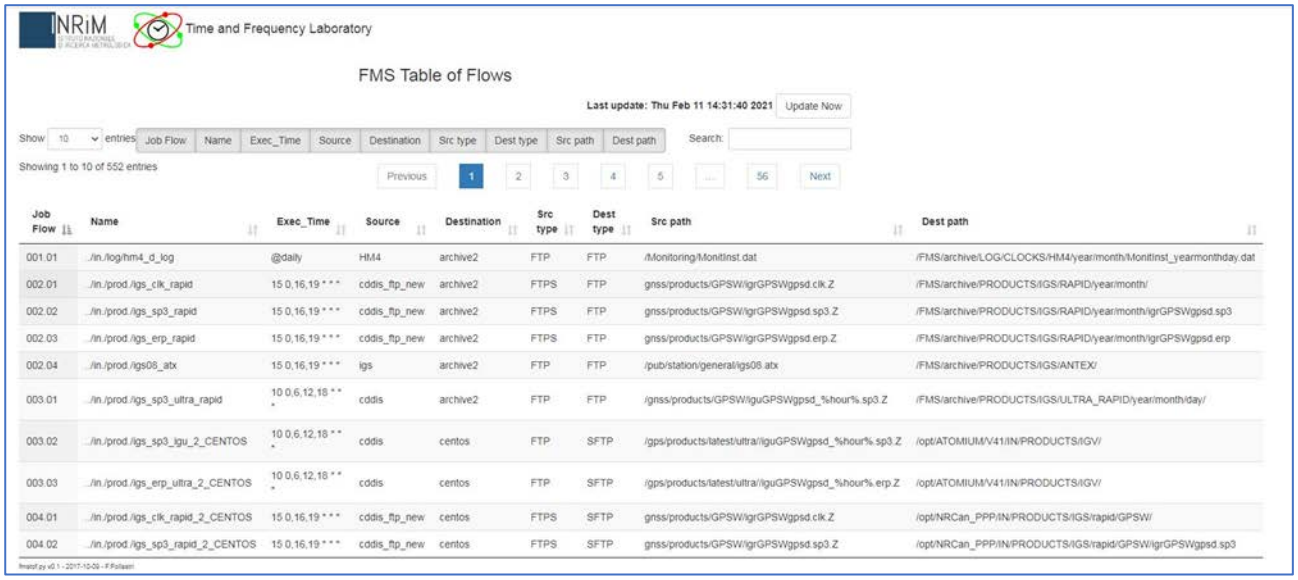

Figure 6: the FMS Flows Reporting Tool main page

- To enable the web page on the hosting machine, first of all the *Apache 2 web server* (https://httpd.apache.org/) must be installed and enabled.
- Then the `fmstof.html` file must be placed (referring to linux platforms) in the `/var/www/html` folder.
- The web page functionalities then point to the `fmstof.bash` and `fmstof.py` *cgi python scripts*, to be placed in `/usr/lib/cgi-bin`.

- The script must have the *execution privileges* for the user and group www-data.

The fmstof package comes together with the FMS package. Please refer to the author for the license agreement (all rights are reserved).

9 Scripts summary (in alphabetical order) and usage

chkRetry
type: bash script
action: checks and launches flows to be retried in the /home/user/FMS/etc/flows/limbo folder
called scripts: main2
usage:
/home/user/FMS/bin/chkRetry

CRX2RNX
type: executable
action: change file format from Hatanaka to RINEX. It must be downloaded from https://terras.gsi.go.jp/ja/crx2rnx.html
Warning: this is a third parties software that FMS can manage but it is not part of FMS itself. For the Hatanaka software licence please consult https://terras.gsi.go.jp/ja/crx2rnx/LICENSE.txt
called by: main2
usage:
cat ${tmp_path}/$filename_source | ${bin_path}/CRX2RNX - > ${tmp_path}/${filename}.out

file_transfer
type: bash script
action: manages file transfer **protocol** according to flow configurations
called by: main2
handled protocols: ftp, sftp, tftp, ncftp, http, smbclient, local copy
called scripts according to $C_SOURCE_TYPE: ftpPut, ftpGet, sftpPut, sftpGet, ftpsPut, ftpsGet, pftpPut, pftpGet, httpGet, smbPut, smbGet
usage (copy file from source to tmp folder for next file manipulation):
$bin_path/file_transfer "$C_SOURCE_TYPE" "${etc_path}/login/${C_SOURCE}" "$
{path_source}" "${filename_source}" "${tmp_path}" "${source_log}"
"$C_DELETE_FROM_SOURCE"
usage (copy file to final destination):
$bin_path/file_transfer "$C_DEST_TYPE" "${etc_path}/login/${C_DEST}" "${tmp_path}" "$
{filename_dest}" "${path_dest}" "${dest_log}" ""

FlowsFromList.awk
type: awk script
action: management of flow list parsing
called by: main2
usage:
awk -f ${bin_path}/FlowsFromList.awk $flow_list_raw > ${tmp_path}/flow_list.tmp

ftpGet
type: bash script

action: implements lftp file retrieval with File Transfer Protocol
called by: file_transfer
usage:
```
$bin_path/ftpGet "$USER" "$PASSWD" "$HOST" "${source_path}/${source_filename}" "$
{dest_path}/" "${log_filename}" "$delete_from_source"
```

ftpPut
type: bash script
action: implements lftp file transfer to remote destination with File Transfer Protocol
called by: file_transfer
usage:
```
$bin_path/ftpPut "$USER" "$PASSWD" "$HOST" "${source_path}/${source_filename}" "$
{dest_path}/" "${log_filename}"
```

ftpsGet
type: bash script
action: implements file retrival with FTP over SSL/TLS (ncftpget)
called by: file_transfer
usage:
```
$bin_path/ftpsGet "$USER" "$PASSWD" "$HOST" "${source_path}/${source_filename}" "$
{dest_path}/" "${log_filename}" "$delete_from_source"
```

ftpsPut
type: bash script
action: implements file transfer to remote destination with FTP over SSL/TLS (ncftpput)
called by: file_transfer
usage:
```
$bin_path/ftpsPut "$USER" "$PASSWD" "$HOST" "${source_path}/${source_filename}" "$
{dest_path}/" "${log_filename}"
```

gpsw
type: bash script
action: converts date to gps week + gps day variables
called by: Replace
usage:
```
var=$($current_dir/gpsw $yyyy $doy)
```
example on command line:
```
>$ gpsw 2014 70
>$ 17832
(where gpsw=1783 and gpsd=2)
```

GregToMjd
type: bash script
action: converts Gregorian Date to Modified Julian Date
called by: Replace
usage:
```
mjddd=$(${current_dir}/GregToMjd $yyyy ${mm} $dd $hh ${MM} "00")
```

httpGet
type: bash script
action: implements wget on http://"${host}""${filepattern}" source
called by: file_transfer

usage:
$bin_path/httpGet "$HOST" "${source_path}/${source_filename}" "${dest_path}/" "${log_filename}"

main2
type: bash script
actions: FMS main script. Loops over flows of given list, implementing file retrieve, transformation (see par. 3: uncompress, gzip, unix2dos, hatanaka/de-hatanaka, file name change) and final transfer, according to each flow configuration, associating email notification and alerting (using SWAKS see par. 7)
called scripts: FlowsFromList.awk, OptionsFromList.awk, Replace, file_transfer, CRX2RNX, RNX2CRX
usage on command line (par. 2):
/home/user/FMS/bin/main2 /home/user/FMS/etc/flows/list/list_name

manual
type: bash script
action: manned launch of list between given datetimes
called scripts: main2
usage on command line (par. 6):
/home/user/FMS/bin/manual /home/user/FMS/etc/flows/list/list_name <from date> <to date> <file rate>

OptionsFromList.awk
type: awk script
action: parsing of list options
called by: main2
usage:
awk -f ${bin_path}/OptionsFromList.awk $flow_list_raw > ${tmp_path}/flow_list_options.tmp

pftpGet
type: bash script
action: implements lftp file retrieval via File Transfer Protocol with PORT specification
called by: file_transfer
usage:
$bin_path/pftpGet "$USER" "$PASSWD" "$HOST" "$PORT" "${source_path}/${source_filename}" "${dest_path}/" "${log_filename}" "$delete_from_source"

pftpPut
type: bash script
action: implements lftp file transfer to remote host via File Transfer Protocol with PORT specification
called by: file_transfer
usage:
$bin_path/pftpPut "$USER" "$PASSWD" "$HOST" "$PORT" "${source_path}/${source_filename}" "${dest_path}/" "${log_filename}"

Replace
type: bash script
action: defines and manages **keywords** (see par. 3 and Annex A), at a given time offset w.r.t. execution time, for the construction of key variables in the main script, such as file

name at the source, path to file at the source, file name at destination, path for file destination
called by: main2
example usage:
filename_dest="$(${bin_path}/Replace "$C_FILENAME_DEST" "$offset_day" "$offset_hour" "$C_OFFSET_UTC" "$basetime")"

RNX2CRX
type: executable
action: change file format from RIINEX to Hatanaka. It must be downloaded from
`https://terras.gsi.go.jp/ja/crx2rnx.html`
Warning: this is a third parties software that FMS can manage but it is not part of FMS itself. For the Hatanaka software licence please consult
`https://terras.gsi.go.jp/ja/crx2rnx/LICENSE.txt`
called by: main2
usage:
${bin_path}/RNX2CRX ${tmp_path}/$filename_source
extention “d” must be then put to modified rinex for convention
(example: filename_source="${filename_source:0:-1}"d)

sftpGet
type: bash script
action: implements lftp for file retrieval from remote host with Secure File Transfer Protocol
called by: file_transfer
usage:
$bin_path/sftpGet "$USER" "$PASSWD" "$HOST" "${source_path}/${source_filename}" "${dest_path}/" "${log_filename}" "$delete_from_source"

sftpPut
type: bash script
action: implements lftp for file transfer to remote host with Secure File Transfer Protocol
called by: file_transfer
usage:
$bin_path/sftpPut "$USER" "$PASSWD" "$HOST" "${source_path}/${source_filename}" "${dest_path}/" "${log_filename}"

smbGet
type: bash script
action: implements smbclient for file retrieval from remote host
called by: file_transfer
usage:
$bin_path/smbGet "$USER" "$PASSWD" "$HOST" "${source_path}" "${source_filename}" "${dest_path}/" "${log_filename}" "$delete_from_source"

smbPut
type: bash script
action: implements smbclient for file transfer to remote host
called by: file_transfer
usage:
$bin_path/smbPut "$USER" "$PASSWD" "$HOST" "${source_path}" "${source_filename}" "${dest_path}/" "/dev/null"

startup.sh
type: bash script
action: sends email warning operator that the FMS host system has just been rebooted
usage in crontab:
```
@reboot    /opt/FMS/bin/startup.sh
```

Conclusions

In the present technical report, the File Management System (FMS) package of the I.N.Ri.M. Time Laboratory was presented. The importance of having a powerful tool for data files management, developed internally and independently from any already customized application, was underlined. The scripts were explained in detail, enlightening the three main features that make FMS a really powerful tool: **the file date identification algorithm**, performed with respect to time of script execution, **the mail alerting functionality**, and **the retry functionality**. Furthermore, the FMS Table of Flows **web interface** was also presented.

Annex A

KEYWORDS

%yearnow%	4-digits year at the time of flow run
%yynow%	2-digits year at the time of flow run
%monthnow%	2-digits month at the time of flow run
%daynow%	2-digits day of month at the time of flow run
%doynow%	3-digits Day Of Year at the time of flow run
MJDDD	Modified Julian Date (MJD) calculated from the offset defined in the flow with respect to execution time.
MJ.DDD	Modified Julian Date (MJD) with point digit, calculated from the offset defined in the flow with respect to execution time.
Day	2-digits day of month calculated from the offset defined in the flow with respect to execution time.
month	2-digits month calculated from the offset defined in the flow with respect to execution time.
year	4-digits year calculated from the offset defined in the flow with respect to execution time.
yy	2-digits year calculated from the offset defined in the flow with respect to execution time.
doy	3-digits Day Of Year calculated from the offset defined in the flow with respect to execution time.
%hour%	2-digits hour calculated from the offset defined in the flow with respect to execution time.
minute	2-digits minute calculated from the offset defined in the flow with respect to execution time.
%hour_alpha%	Hour expressed with lowercase letter (from a to x means from 00:00 to 23:00), calculated from the offset defined in the flow with respect to execution time.

%hour_ALPHA%	Hour expressed with uppercase letter (from A to X means from 00:00 to 23:00), calculated from the offset defined in the flow with respect to execution time.
GPSW	GPS Week calculated from the offset defined in the flow with respect to execution time.
gpsd	GPS Week Day calculated from the offset defined in the flow with respect to execution time.

GLOBAL VARIABLES (with default values)

G_ROOT = /FMS

G_ARCHIVE_ROOT = "${G_ROOT}/archive"

G_TIMETRANSFER_ROOT = "${G_ARCHIVE_ROOT}/TIME_TRANSFER"

G_ROOT_GNSS = "${G_TIMETRANSFER_ROOT}/GNSS"

G_ROOT_TW = "${G_TIMETRANSFER_ROOT}/TWSTFT"

G_LOG_ROOT = "${G_ARCHIVE_ROOT}/LOG"

G_MONITORING_ROOT = "${G_ARCHIVE_ROOT}/MONITORING"

G_CERTIFICATES_ROOT = "${G_ARCHIVE_ROOT}/CERTIFICATES"

G_PRODUCT_ROOT = "${G_ARCHIVE_ROOT}/PRODUCTS"

G_DELIVERY_TYPE = FTP

G_DELIVERY_ROOT = ${G_ROOT}/deliveries

G_NOT_DELIVERY_TYPE = FTP

G_NOT_DELIVERY_ROOT = ${G_ROOT}/not_delivered

G_RETRY_FOLDER = limbo

G_RETRY_PATH = /home/user/FMS/etc/flows/${G_RETRY_FOLDER}

G_RETRY_TIME = 10 (e.g. Here put integer representing # of minutes of next retry)

G_MAX_RETRIES = 11 (e.g. Here put integer representing # of max retries)

G_MAIL_TO = "someuser@someinstitution.something”
(e.g. Here put email of main user/operator)

LIST VARIABLES (with default values)

L_MAIL_TO = "" (to configure recipients just for that specific list)

L_INFO_REPORT_ENABLE = 0 (if put to 1, then the notification service is active and the other following options are verified)

L_INFO_REPORT_ATTACHMENTS = 1 (by default, the presence of an attachment is foreseen. 0 for no attachments)

L_INFO_REPORT_MESSAGEBODY = "" (configurable message)

L_INFO_REPORT_SUBJECT = "" (configurable subject)

L_ENABLE_RETRY = 1 (if needed to disable retry at list level, set to 0)

L_ATTEMPTS_BEFORE_ERROR = 1 (# of attempts before getting an error message)

SINGLE FLOW CONFIGURATION VARIABLES (with default values)

C_ACTIVE = 1 (put 0 to inactivate a specific flow inside a list)

C_SOURCE_TYPE = "" (explained in par. 3: to be always defined)

C_DEST_TYPE = "" (explained in par. 3: to be always defined)

C_SOURCE = "" (explained in par. 3: to be always defined)

C_DEST = "" (explained in par. 3: to be always defined)

C_OFFSET_DAY = 0 (put integer to define offset to target file w.r.t. time of execution)

C_OFFSET_HOUR = 0 (put integer to define offset to target file w.r.t. time of execution)

C_OFFSET_UTC = 0 (put integer to define offset to target file w.r.t. time of execution in UTC. It can be neglected if the hosting machine time is already in UTC.)

C_PATH_SOURCE = "" (explained in par. 3: to be always defined)

C_FILENAME_SOURCE = "" (explained in par. 3: to be always defined)

C_DELETE_FROM_SOURCE = 0 (file is copied; if 1, then file is moved)

C_UNZIP = 0 (no action if 0; if 1, file is unzipped)

C_DEHATANAKA = 0 (no action if 0; if 1, file is changed from Hatanaka http://sopac.ucsd.edu/hatanaka.shtml format to ASCII)

C_HATANAKA = 0 (no action if 0; if 1, file is change from ASCII to Hatanaka format)

C_UNIX2DOS = 0 (no action if 0; if 1, file is change from UNIX to DOS format)

C_ZIP = 0 (no action if 0; if 1, file is zipped)

C_ZIP_EXTENSION = “.gz” (default zipped file extention. The main script uses **gzip**)

C_ZIP_NEWNAME = "" (no change name for zipped file by default. If it must be changed, put new name here)

C_PATH_DEST = "" (explained in par. 3, must always be present)

C_FILENAME_DEST = "" (explained in par. 3, must always be present)

C_COPY_TO_DELIVERY = 0 (0 means no copy to “deliveries” folder, default condition. 1 means ok copy to “deliveries” folder)

C_COPY_TO_NOT_DELIVERED = 0 (0 means no copy to “not delivered” folder, default condition. 1 means ok copy to “deliveries” folder)

C_ENABLE_RETRY = 1 (explained in par. 5)

References

[1] G. Cerretto et al., "INRIM Time and Frequency Laboratory: an update on the status and on the ongoing enhancement activities", Proc. of the Precise Time and Time Interval Systems and Applications (PTTI), Boston, MA, USA, December 1st – 4th 2014.

[2] V. Formichella et al., “The First Months of Fully Automated Generation of the Italian Time Scale UTC(IT)”, Proc. of the Precise Time and Time Interval Systems and Applications (PTTI), Virtual Sessions, January 25th – 27th 2021.

[3] F. Pollastri et al., ”The new Data Measurement System - DMS of the INRIM Time Laboratory”, I.N.Ri.M. Techincal Report 3/2021.