USER'S GUIDE

BIOMAJ v1.2

Authors: David Allouche (allouche@toulouse.inra.fr)**, Olivier Filangi** (olivier.filangi@irisa.fr)**

Website: http://biomaj.genouest.org/

This document was issued on: 8 Sep 2011

Acknowledgements:

The authors would like to thank:
-Christophe Caron, Olivier Collin and Hugues Leroy for their contributions.
-The Bretagne Regional Council and the *Réseau National des plates-formes de Bio-Informatique* (ReNaBi) for supporting this project.

Table of Contents

1Introduction....5
1.1Why BioMAJ?....5
1.2Background....5
1.3Presentation....5
2Setting up BioMAJ....7
2.1Product contents....7
2.1.1Available banks....7
2.1.2Available indexes / conversions....8
2.2Pre-requisites....8
2.2.1System and hardware....8
2.2.2Utility applications....9
2.2.2.1Software usually present on a standard Linux installation....9
2.2.2.2Software requiring special attention....9
2.3Installation....10
2.3.1Initialising environment variables....10
2.3.2Compilation....10
2.3.3Application global variables....10
2.3.4General configuration: global.properties file....11
2.3.5Database management....11
2.3.6Statefiles migration....12
2.3.7BiomajWatcher installation....12
2.3.8First run of BioMAJ....12
2.3.9Demonstrations....13
2.3.9.1Alu bank of the NCBI (with formatdb process)....13
2.3.9.2STS bank of the NCBI (with fastacmd process)....13
2.3.10BioMAJ directories....13
2.3.11Proxy Configuration....14
3Using the application....16
3.1Application behaviour....16
3.1.1Update cycle....16
3.1.2Data organisation....17
3.2Data maintenance....19
3.2.1Updating a bank....19
3.2.2Deleting one or several bank versions....24
3.2.3Changing the bank name....24
3.2.4Changing production directories....24
3.2.5Rebuilding a version....25
3.2.6Error correction....26
3.2.7Importing data....27
3.2.8Database cleanup....27
3.3Automating updates....27

3.4Warehouse monitoring....29
3.4.1E-mail alert: monitoring an update session....30
3.4.2Debugging and error identification during running....32
3.4.3Exploring in the command line: biomaj.sh --status....33
3.4.3.1Exploring the warehouse....33
3.4.3.2Exploring the status of a bank....34
3.4.4Html report....36
3.5Multitenancy....37
4Workflow creation....38
4.1General information....38
4.2Bank update session report....38
4.2.1Configuring the source: description / classification / location....38
4.2.1.1Description and classification....38
4.2.1.2Location of data....39
4.2.2Configuring the data Synchronisation stage....39
4.2.2.1Remote server and download protocol....40
4.2.2.1.1FTP protocol....40
4.2.2.1.2HTTP protocol....41
4.2.2.1.3Direct HTTP protocol....42
4.2.2.1.4Sftp protocol....43
4.2.2.1.5Amazon S3 protocol....43
4.2.2.1.6Rsync protocol....43
4.2.2.1.7Local protocol....43
4.2.2.2Building download filters (remote.files/local.files)....44
4.2.2.3Getting a version number....46
4.2.2.4Viewing data....47
4.2.3Configuring the pre, post & remove processing stages....49
4.2.3.1General operation....49
4.2.4Defining elements....50
4.2.4.1Block....50
4.2.4.2Meta-process....51
4.2.4.3Process....51
4.2.5Workflow example from scripts available in BioMAJ....54
4.2.6Deployment....56
4.3Information overload: global.properies....57
4.4Computed bank : bank dependencies....58
5Developing and integrating post or pre-processes....60
5.1.1Communication management: messaging....60
5.1.2Managing dependencies of produced files....61
5.1.3Context to script: information on the version number....62
5.1.4Return code....63
5.1.5Debugging....63
5.2Virtual bank concept....63
6Metadata and classification of sources and processes....64
6.1Definition....64
6.2BioMAJ implementation....64
6.3Potential uses....65
7F.A.Q....66
8Appendix....69
8.1Example of configuration files....69

8.1.1Global.properties.....69
8.2BioMAJ properties.....70

1 Introduction

1.1 Why BioMAJ?

In bio-computing, analyses are almost systematically reliant on databanks. Any bio-computing site therefore needs to manage these invaluable databanks that hold a huge amount of information, usually several terabytes, spread over various international sites and in a consistent format (there are still several different standards currently).

The type and number of these databanks are constantly changing and the frequency with which they are updated varies greatly. In most cases, before they can be used, they need to be reformatted several times, a process that can be very costly in terms of calculation time (several days for SRS indexes, for example). Monitoring of updates and format conversion results usually produce trace files (logging) that need to be analysed in the event of any errors (by reading a daily report e-mail and analysis of the trace files if required). Management of this data is cumbersome and fiddly. Data managers and users can often lose their way.

1.2 Background

The BioMAJ project came out of the work of three teams in 2005: INRIA Rennes and INRA Toulouse and Jouy-en-Josas. At the time, no free applications met users' requirements. The closest application was citrina, developed by Josh Goodman (from Washington University's gmod project). This was a promising prototype – nonetheless quite far from the application required – and it had not been updated since 2004.

In 2006, these teams (INRIA and INRA) developed a new engine called BioMAJ[1]. Based on citrina 0.51, nearly all the code was rewritten and the application's architecture and functions were completely rethought and considerably extended.

During 2007, the application was tested on the three sites involved in the project to make it more robust and suitable for large-scale bio-computing. It was deployed in production on the three sites involved.

1.3 Presentation

BioMAJ provides:

- A reliable workflow engine that can download remote data automatically and intelligently (error correction, synchronisation of local and remote data), apply formatting to this data and put it into production (make the data available for all users and/or applications).
- A group of predefined workflows for the main biological banks.
- An indexing scripts library (formatting for biological data) to be applied to workflows defined by the administrator or on the predefined workflows.

[1] **Bio**logie **M**ise **A** **J**our

Application's main functional features:

- Management of updating cycles for a data source ;
- Logging of workflows;
- Monitoring of the integrity of transferred data;
- Standardisation of data organisation (standard directory structure);
- Management of classification of data sources (depending on type, origin, application index, etc.) ;
- Management of the local data warehouse, consultation of available versions and management of source versions (addition, deletion of version and source);
- Administration of each databank with a set of commands ;
- Error correction, i.e. restarting an unfinished cycle;
- Rebuilding of a given release ;
- Description of a relatively complex post-processing workflow;
- Supervision of the application via e-mail alerts, positioned at key points in the workflow of each source. During use of crontab, this helps alert the administrator as to the correct or incorrect functioning of sessions ;
- Monitoring of the data warehouse, generation of statistics on make-up of the data warehouse. Monitoring of source development over time. Creation of web summary reports containing an overview of warehouse data and a development history source by source.

BioMAJ is an application designed to help maintain a data warehouse and is intended to assist the data administrator. BioMAJ can manage a great mass of similar data and locally consolidate all distributed sources available on the network via protocols such as ftp, http and rsync, and then automate more or less complex processing chains on this data. The application automates the updating cycle and facilitates supervision of the catalogue of managed databanks while providing a history of maintenance operations. The application helps maintain the main biological databanks provided by the international scientific community on a local platform. BioMAJ's field of application is nonetheless wider than this and could be extended to any area which manages massive distributed data.

BioMAJ provides flexibility in managing banks of sequences on a site while allowing for rapid implementation of new workflows by simply creating a bank description file.

This document presents the set up and installation procedures for BioMAJ on a Unix system. The various aspects of daily management are then covered. Advanced functions, including creation of workflows and post-processing, are also analysed.

2 Setting up BioMAJ

2.1 Product contents

- `bin/biomaj.sh` : BioMAJ executable
- `bin/env.sh` : environment variables to be defined for processing
- `conf/process` : processing scripts
- `conf/db_properties` : directory of properties files used by BioMAJ
- `conf/db_properties/samples` : sample properties files without post-processing.
- `conf/db_properties/samples-extended`: sample properties files with post-processing (indexing).
- `doc` : BioMAJ documentation
- `src` : sources
- `sbin` : additional utilities
- `workflows` : scripts for creating updates

2.1.1 Available banks

To work, the application needs a configuration file containing a description of the workflow associated with each data source. This file is the properties file of the bank.

Construction of a properties file is not difficult, however, it is a laborious task that requires good knowledge of the data source and the BioMAJ application. As validation of the properties file is not immediate, we are providing sample configuration files for several common banks. These can be found in the following directories:

`$BIOMAJ_ROOT/conf/db_properties/workflow_withprocess`

and

`BIOMAJ_ROOT/ conf/db_properties/workflow_withoutprocess.`

We suggest that you mutualise the properties files. If you develop them for specific banks and you want to share them, you can post them to the list: BioMAJ-users@lists.gforge.inria.fr.

After validation, these files will be included in future versions of the software.

In terms of validation, to be validated, a post-process or properties file must be successfully tested on at least two sites using BioMAJ. Minimal documentation should also be provided. This must include a script synopsis and a description of the installation procedure: pre-requisites, environment variables, configuration of applications using data from the script.

A few examples are given in the next section.

2.1.2 Available indexes / conversions

Some post-processes are available with the engine as standard:

- Fastacmd: `conversion of blast bank into fasta file`
  `$BIOMAJ_ROOT/doc/process/fastacmdTLSE.html`

- Formatdb: `indexing a fasta bank for a blast ncbi`
  `$BIOMAJ_ROOT/doc/process/formatdbTLSE.html`

- SRS : `SRS indexing`
  `$BIOMAJ_ROOT/doc/process/indexSrsTLSE.html`

Other process:

- SendMail: Script for sending an e-mail alert.
  `$BIOMAJ_ROOT/doc/process/sendMailTLSE.html`

Each of these scripts has individual configuration help in the *doc/process* directory. In most cases, you need to adapt the workflow to your own configuration.

2.2 Pre-requisites

2.2.1 System and hardware

BioMAJ was widely developed and tested on architectures with AMD or Intel X86 or X86-64 processors on Linux. However, given the software technology used (java and ant), it should be usable on a wide range of computers and systèmes d'exploitation, including Mac OS, and the most popular versions of Unix (BSD, Solaris, AIX, ...).

The main OS compatibility limits aren't due to the engine but to the post-process used. Indexing applications are not necessarily compatible with or available on all platforms.

Using BioMAJ on Windows is more difficult. It still needs further checking but with a few changes to the configuration, the application should be able to work in these environments using the cygwin emulator.

BioMAJ is a general-use application that can synchronise different types and size of data. When used in a biological context, the application has to carry out massive downloads and extremely resource-heavy post-processing. The hardware architecture hosting the application must therefore be of a suitable size. The application works on a simple laptop but for full operation (data mining and indexing), of workflows in distribution, **a multi-processor (bi or quadri) and 1-2 TB of disk space are strongly recommended.**

Note that ideally, a central storage space accessible via a cluster of machines can help BioMAJ provide much faster indexing. In this event, calculation delay using rsh, ssh, or any ordering such as Torque, Pbs, Sge, Lsf or Condor is required. All calculation machines must have the capacity to write to a shared storage space. To do this, several options are available: using NFS, a clustered file system, an NAS, a split file system, etc. **These solutions are relatively complex to implement but they are the price to be paid for optimum management of data on a large scale.**

2.2.2 Utility applications

This section contains the software required prior to installation of the application engine:

The list of pre-required software is subdivided into two sections:

2.2.2.1 Software usually present on a standard Linux installation

- GNU Tar 1.13 or higher (http://www.gnu.org/software/tar/tar.html)
- Gzip 1.3.3 or higher (http://www.gzip.org/)
- Bzip2 0.9.0 or higher (http://sources.redhat.com/bzip2/)

2.2.2.2 Software requiring special attention

- Java 1.6.x (http://java.sun.com/j2se/)
- Ant 1.7.0 or higher(http://ant.apache.org/)

This list does not include the pre-requisites for each post-process.

Each post-process called by the engine generally uses one or more utilities belonging to it.

e.g.: For the purposes of the demo, installation of the toolkit blast ncbi is required as well as the software in the above lists. Indexing uses blast and fasta formatting post-processes. Another example is that the SRS indexing post-process requires pre-installed SRS and an ad hoc configuration of the application (cf. doc/process/srsdb.i). More generally, each post-process includes dedicated documentation in the directory /doc/process/

2.3 Installation

Described below are the steps for a manual installation of Biomaj. If you installed BioMAJ via a distribution package (.deb or rpm), go directly to section 2.3.6.

2.3.1 Initialising environment variables

BioMAJ requires three environment variables to be set to run properly. These are defined in the file `bin/env.sh`.

```
export BIOMAJ_ROOT=
export JAVA_HOME=
export ANT_HOME=
```

- `BIOMAJ_ROOT`: The BioMAJ installation path
- `ANT_HOME`: ANT home directory
- `JAVA_HOME`: JAVA home directory

2.3.2 Compilation

To compile the application, you just need to enter the following commands in BIOMAJ_ROOT directory:

```
ant
```

In the event of successive compilations, we recommend you clean up intermediate files created by previous compilations by using the command (to be run in the BioMAJ root directory):

```
ant clean
```

2.3.3 Application global variables

To run post-processes, applications often require specific environment variables.

The BioMAJ engine uses the script `$BIOMAJ_ROOT/bin/env.sh` to position the required environment variables for running post-processes (indexing, e-mail, etc.).

This script is especially useful when the application administrator does not have sufficient rights to initialise variables at the overall level of the system or when they want to separate their usual working environment from the application environment.

Example of an initialisation script for an srs or blast environment:

```
#!/bin/sh
export BLASTDB=/bank/blastdb #initialisation of the variable
                             #BLASTDB
. /data/srs/srs/etc/prep_srs.sh #initialisation of environment
                                #required for SRS
```

2.3.4 General configuration: global.properties file

The file `$BIOMAJ_ROOT/conf/db_properties/global.properties` contains a number of internal variables for the BioMAJ application that are pre-initialised in this file (cf. writing new workflows section)

Before running BioMAJ (script $BIOMAJ_ROOT/bin/biomaj.sh), you should initialise the root of your local data repository.

➢In the file `$BIOMAJ_ROOT/conf/db_properties/global.properties`, initialisation of the variable data.dir is essential for running BioMAJ

2.3.5 Database management

Since version 1.1.0, logging system is no longer XML based but relies on a relational database.

Biomaj is shipped with an embedded database, HSQLDB, which requires very few configuration, but also supports MySQL.

The configuration is done via the `global.properies` file :

```
#----------------
#Database configuration
#----------------

#database.type=mysql
#database.driver=com.mysql.jdbc.Driver
#database.url=jdbc:mysql://{server adress}/{database name}
#database.login=
#database.password=

database.type=hsqldb
database.driver=org.hsqldb.jdbcDriver
database.url=jdbc:hsqldb:hsql://localhost/bmajdb
database.login=sa
database.password=
```

MySQL installation and configuration is left to the user. The schema of the database is available in `$BIOMAJ_ROOT/sql/mysql.sql`. As for HSQLDB, a few scripts have been written to manage the basic operations :

- Starting the server : `$BIOMAJ_ROOT/bin/start_dbserver.sh`

- Stopping the server : $BIOMAJ_ROOT/bin/stop_dbserver.sh
- (Re)Creating the database : `$BIOMAJ_ROOT/bin/create_dababase.sh`

Note that if you manage a significant amount of banks, MySQL is likely to offer better performances.

2.3.6 Statefiles migration

To migrate the data from an older version of Biomaj which relies on xml "statefiles" (<1.1.0), you can use this script :

```
$BIOMAJ_ROOT/bin/import_statefiles.sh
```

It can take either a list of files or a directory to search the statefiles in (see the command help for further information). Please note that depending on the amount of data you want to import and the RDBMS (HSQL or MySQL), the process can last from a few seconds to several hours.

2.3.7 BiomajWatcher installation

BiomajWatcher (BW) is a web interface (available since BioMAJ 1.1.0) designed to provide a quick overview of available banks and basic administration commands. Refer to BiomajWatcher specific documentation.

2.3.8 First run of BioMAJ

Two sample banks are provided for demonstration purposes. These are the alu and sts banks of the ncbi.

For each one, two workflow scenarios can be used:

- without post-process
- with post-process (blast indexing for alu and fasta conversion for sts)

If you want to use the demonstration workflows, take care to place these properties files in the directory `$BIOMAJ_ROOT/conf/db_properties/`

Use of a demonstration scenario with post-processing means you need to have available the executables contained in the blast package of the ncbi. This is available at: ftp://ftp.ncbi.nih.gov/blast/executables/LATEST.

Before use, you must also update the paths for the executables called in the file:
`$BIOMAJ_ROOT/conf/process/unix_command_system.cfg`

```
FASTACMD=/MY/CHEMIN/fastacmd
FORMATDB=/MY/CHEMIN/formatdb
```

For more information on configuration and running of available post-processes, see the directory:
`$BIOMAJ_ROOT/doc/process`

NB: The list of parameters that can be used in the properties files of your banks is available in the

appendix to this document.

2.3.9 Demonstrations

2.3.9.1 Alu bank of the NCBI (with formatdb process)

Configuration file: `$BIOMAJ_ROOT/conf/db_properties/alu.properties`

To not run blast indexing (formatdb), the following comment line should appear in the configuration file (use a # at the beginning of the line):

```
db.post.process=POST1
```

Run command: `$BIOMAJ_ROOT/bin/biomaj.sh --update alu --console`

2.3.9.2 STS bank of the NCBI (with fastacmd process)

Configuration file: `$BIOMAJ_ROOT/conf/db_properties/sts.properties`

To not run blast indexing (formatdb), the following comment line should appear in the configuration file (use a # at the beginning of the line):

```
db.post.process=POST1
```

Run command: `$BIOMAJ_ROOT/bin/biomaj.sh --update sts –console`

2.3.10 BioMAJ directories

The directories used by BioMAJ :

- log generated by the application
- workflows definition
- production state files (for old biomaj versions and migration purposes)
- post-processing directory
- html report directory
- directory that contains database and related stuff

A configuration file is located on the root directory : `$BIOMAJ_ROOT/general.conf`. This file is generated on the first execution of the application.

```
# Created by BioMAJ 0.9.3.2
# Date : 03/01/2008 14:36:30
# File : General configuration
[DIRECTORIES]
log.dir        =/home/biomaj/log
log-biomaj.dir =/home/biomaj/log/biomaj-runtime
statefiles.dir =/home/biomaj/statefile
workflows.dir  =/home/biomaj/conf/db_properties
```

```
process.dir    =/home/biomaj/conf/process
webrepport.dir =/home/biomaj/rapport

[APPLICATIONS]
uncompress.bin=tar,tar2,tar3,gunzip,bunzip,unzip

tar.bin=/bin/tar
tar.case=.tar.gz,.tgz,.tar.Z
tar.option.uncomp=-zxf
tar.option.output=-C
tar.option.test=-tf

gunzip.bin=/usr/bin/gunzip
gunzip.case=.gz,.Z
gunzip.option.uncomp=-f
gunzip.option.test=-t

tar2.bin=/bin/tar
tar2.case=.tar.bz2
tar2.option.uncomp=-jxf
tar2.option.output=-C
tar2.option.test=-tf

bunzip.bin=/usr/bin/bunzip2
bunzip.case=.bz2
bunzip.option.uncomp=-f
bunzip.option.output=-c
bunzip.option.test=-t

tar3.bin=/bin/tar
tar3.case=.tar
tar3.option.uncomp=-xf
tar3.option.output=-C
tar3.option.test=-tf

unzip.bin=/usr/bin/unzip
unzip.case=.zip
unzip.option.uncomp=-f
unzip.option.output=-d
unzip.option.test=-t

rsync.bin=/usr/bin/rsync
```

2.3.11 Proxy Configuration

Biomaj support proxying under some conditions.

For the HTTP protocol, a classic HTTP proxy is ok.

For the FTP protocol, your proxy must handle both SOCKS (4 or 5) and FTP protocol.

For both cases, you have to initialize the following properties in the file

`$BIOMAJ_ROOT/conf/db_properties/global.properties` :

- proxyHost
- proxyPort
- proxyUser (optional)
- proxyPassword (optional)

In addition, you have to specify some environment variables that will be used for the download in the file `$BIOMAJ_ROOT/bin/env.sh`.

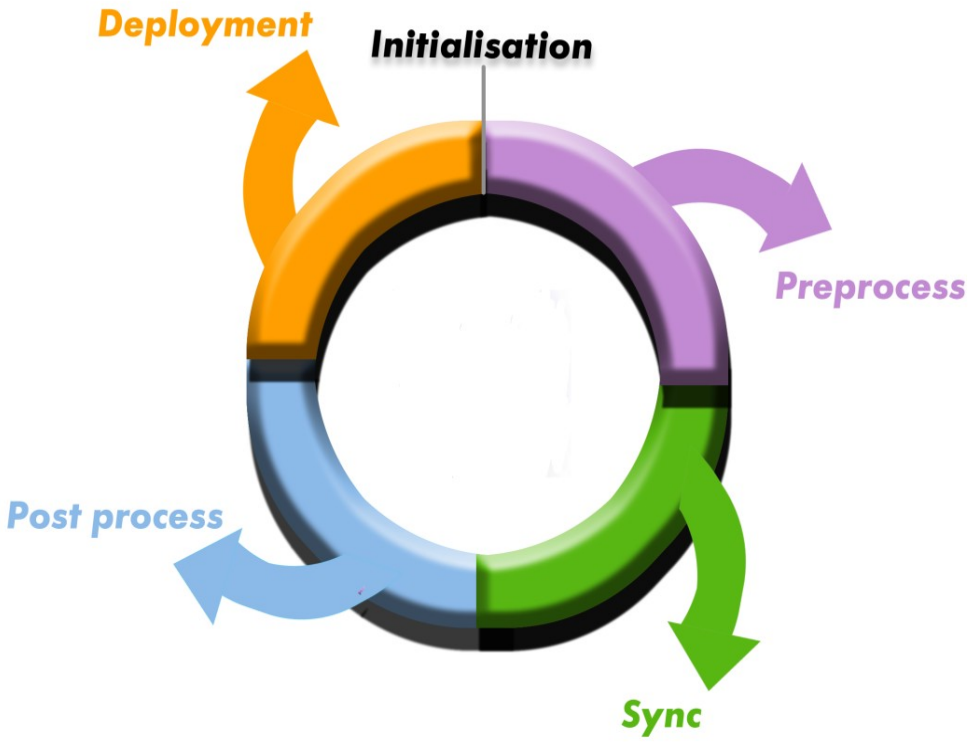
- `export http_proxy=http://login:mdp@[host]:[port]`
- `export ftp_proxy=http://login:mdp@[host]:[port]`

3 Using the application

3.1 Application behaviour

3.1.1 Update cycle

In this section, the cycle and stages that form it are summarised.



BioMAJ is designed to carry out update cycles for a data source. Each cycle has five stages.

1. Initialisation:

The engine loads the properties file containing the workflow description and looks at the current status of the bank by running through the associated status file. After determining the bank's status, the application opens the full cycle or, if necessary, tries to finish the previously incomplete cycle (in the event of an error correction).

A data source has an associated file that must be named *{bankname}*.properties (e.g. for the nr bank: nr.properties). Running the properties file describing the workflow (nr.properties) will generate or add to a status file showing the workflow's progress (run date of sub-tasks, log of amended files, etc.).

The properties file has to be placed in the directory ${workflows.dir} defined in ${BIOMAJ_ROOT}/general.conf.
The status file is generated in xml format in the directory ${statefiles.dir} defined in ${BIOMAJ_ROOT}/general.conf.

2. **Pre-processing**

This is a sub-workflow run before the data update. It has the same properties as the post-processing part explained below. Its purpose is simple: to start tasks, controls and alerts prior to the rest of the updating process.

3. Synchronisation
During this stage, the engine connects to the source and checks for new data compared to data already present locally. It determines the list of files to be downloaded and assigns a version name. It then carries out the download followed by extraction. Finally, it consolidates the data by producing a full version of the bank, adhering to the restrictions defined by the properties file.

4. Post-processing
During this stage, the engine runs the post-processing sub-workflow. The form can describe relatively complex workflows. It is a succession of task blocks that contain one or several sub-collections of meta-tasks that can themselves be made up of several processes. Each block is run in sequence. In a given block, the meta-tasks that make it up are run in parallel. In a meta-task, processes are run in sequence. If there is an error in a process, only the branch that it belongs to is stopped. This creates a Directed Acyclical Graph (DAG).

5. Deployment
If the previous stages have completed without error, the application puts the new version of the source online. Then it deletes the obsolete versions and the temporary files produced when running the post-processes.

All stages in a session are written up into the status file. If there is an error, during the following session, the application will try to continue the session from the first erroneous stage of the previous session to complete the cycle. One cycle is associated with a data source. One or more sessions may be necessary to complete a cycle.

3.1.2 Data organisation

When BioMAJ manages data sources, the application produces a folder structure by default for each one. If there is one missing, directories are created during the first update session. An example of a typical folder structure is shown in the below diagram.

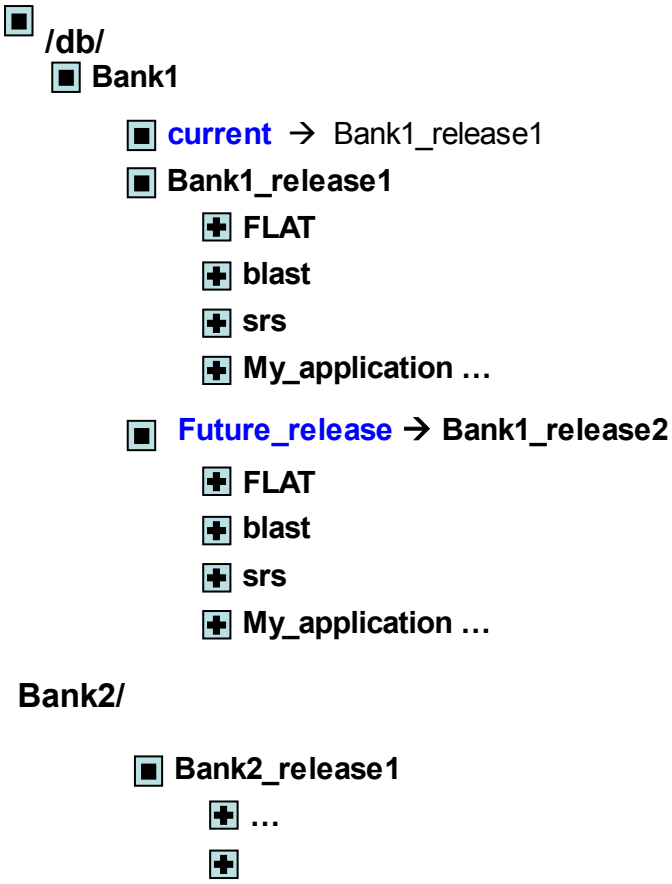


Illustration 1: Organisation of the production folder

The BioMAJ folder structure is based on a four-tier architecture:

➢`/db`: This is the path where all sources maintained by the application will be kept (name of the root bank production directory defined by the property ***data.dir***) .

➢`/db/Bank[n]`: all biological banks in production on the site. This second level is defined by the property ***dir.version*** (redefined for each workflow), if it is not defined, this property takes the value ***db.name*** (the source name) by default.

➢`/db/Bank[n]/Bank[n]_release[p]:` each downloaded version is placed in a version directory. The name of this directory is by default called: ***db.name_release***. The release number used depends on the property ***keep.old.version***.

➢`/db/Bank[n]/ Bank[n]_release_[p]/flat`: each version has the same organisation. Downloaded files have been moved into a flat directory after extraction. At the same level, we can find several directories, each containing the results of indexing post-processing applied to the version in question.

➢`/db/Bank[n]/current`: For each bank, a current link is automatically positioned by BioMAJ during deployment of a new version of the bank.

➢`/db/Bank[n]/future_release`: Similarly a future_release link is placed on the directory of a version under construction.

The two links « current » and « future_release » have an invariable name. They are therefore stable in time. We recommend that you use them when configuring applications that carry out indexing (future_release link) or use the data (current link).

 **However, users who carry out very lengthy processing should use absolute paths of the bank versions. If data is then updated, they will not be affected by changes in position of the « current » link during calculation.**

3.2 Data maintenance

3.2.1 Updating a bank

For the first update of a bank, you have two options: use a configuration file (workflow description) already defined by the directory:

```
$BIOMAJ_ROOT/conf/db_properties/workflow_withprocess
```

or

```
$BIOMAJ_ROOT/conf/db_properties/workflow_withoutprocess
```

or create a new workflow (see 4).

To show that the BioMAJ engine can handle a new workflow, a properties file must always feature in the directory:

```
$BIOMAJ_ROOT/conf/db_properties/
```

To run the workflow in console mode:

```
$BIOMAJ_ROOT/bin/biomaj.sh --update [bank name] –console
```

 **Console mode is practical for viewing a workflow running. We recommend you use it occasionally to check the behaviour of a properties file.**

Example with alu bank:

```
$BIOMAJ_ROOT/bin/biomaj.sh --update alu --console
```

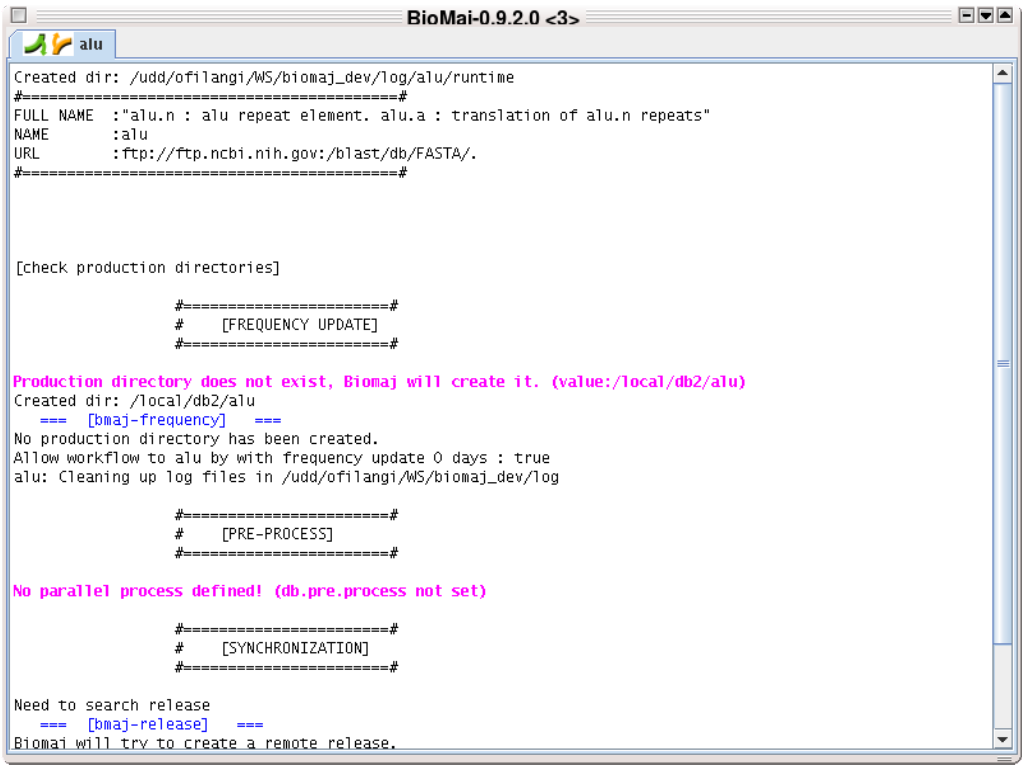


Illustration 2: Console d'une première mise en ligne de la banque alu

BioMAJ creates temporary and production directories if they do not already exist. In this example, note that there is no pre-processing.

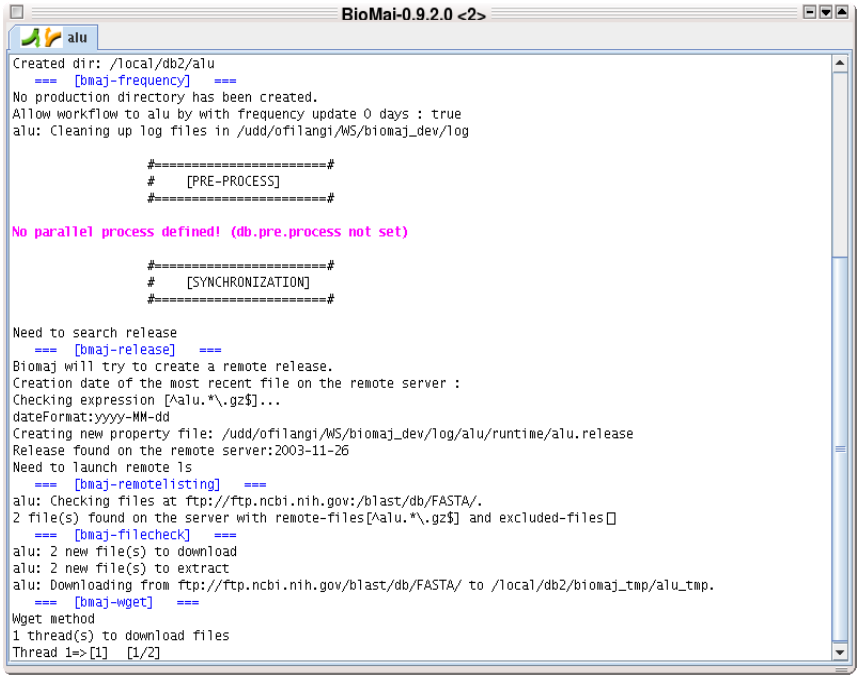


Illustration 3: Console - phase de synchronisation pour la banque alu

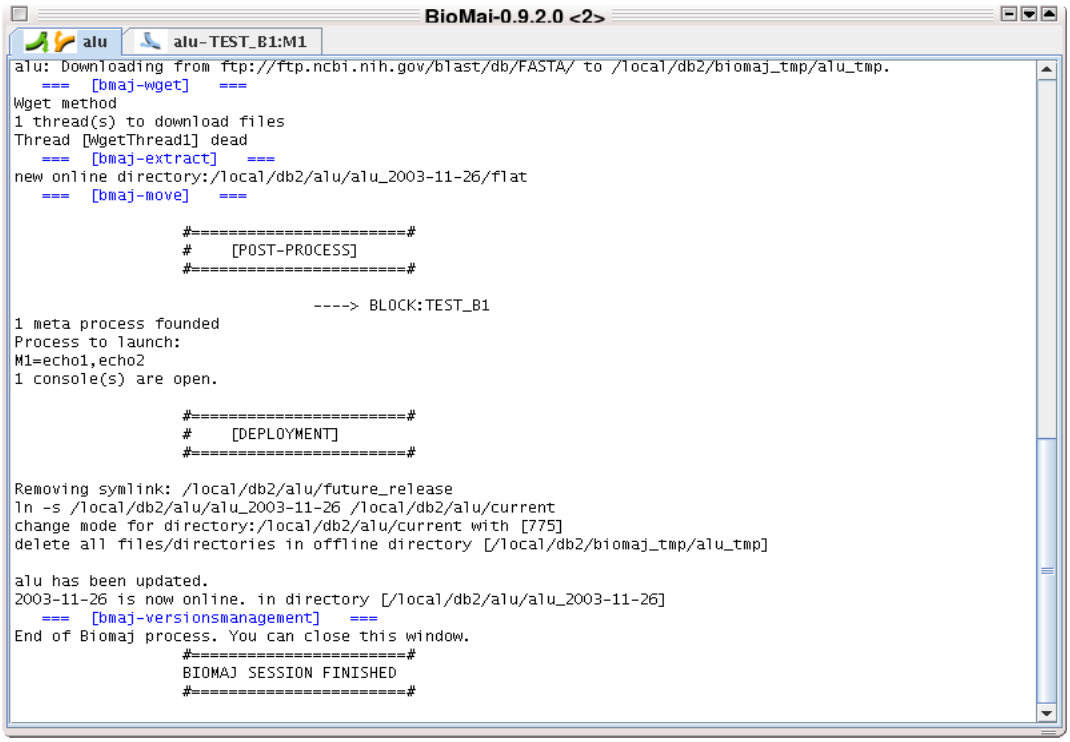


Illustration 4: Console – fin de phase de synchronisation et de déploiement pour la banque alu

As can be seen in the illustrations above (cf illustration 2 and 3), the synchronisation phase is subdivided into seven sub-tasks: release ; remotelisting ; filecheck ; download; extract ; versionsmanagement ; move

-To begin with, the task **release** gets the version number (or by default the application gives it the date of the most recent file for the version) ;
-Then the task **remotelisting** in BioMAJ sets the elements (files and directories) that make up the remote version ;
-They are then compared to the local files during the **filecheck** task. In this example, no file exists locally ;
-They are picked up during the download stage ;
-The downloaded files are extracted during the **extract** task;
-Then the directory of the new version is created (**versionsmanagement** task) .
-Finally, the files are moved to the future production directory during the **move** task (in the example: /local/db2/alu/alu_2003-11-26/flat) .

Following this stage, a post-processing stage is run in a new tab (cf illustration 4).
By clicking on this tab, you can see the post-processing log (here a simple echo). Each **bmaj-execute** task represents an executed process.

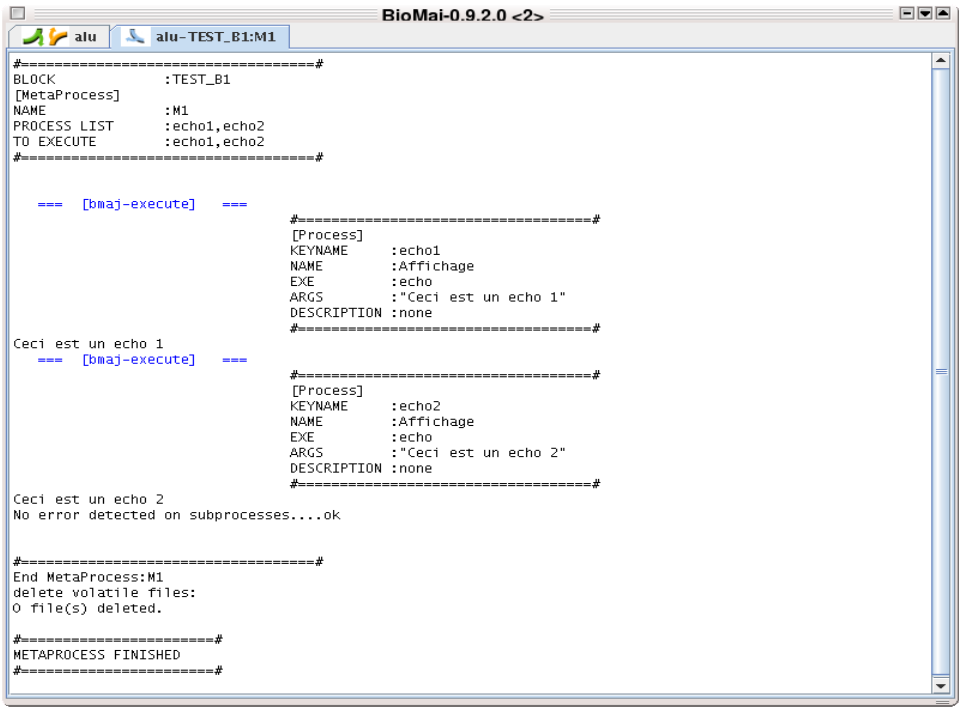


Illustration 5: Phase de post-processus

When the post-processing stage is finished, you can follow the final progress of the workflow (deployment task) by clicking on the first alu tab. The deployment task creates a symbolic current link on the production direction that has just been created, modifies the access rights to make this directory accessible to users and deletes the contents of the temporary alu directory (files created but not used in production).

3.2.2 Deleting one or several bank versions

BioMAJ keeps a number of versions defined by the use of the parameter **keep.old.version**. By default, this parameter is set to zero (cf. global.properties). At most, the engine will keep one version in production and one under construction. The parameter **keep.old.version** can be loaded in the properties file of any bank. It will only be taken into account for the source in question.

It is quite usual for the data warehouse to be at saturation limit during operation. It is therefore necessary to clean up quite quickly.

To remove a bank that depends on management by BioMAJ, the -- remove command is:

```
$BIOMAJ_ROOT/bin/biomaj.sh --remove [bankname] --keep-dir-prod [true|false]
```

This option:

- Asks which version you want to delete
- Then deletes the log directory `$BIOMAJ_ROOT/log/[bankname]`
- It keeps the production directory data if the option `--keep-dir-prod` is used
- It launches remove process for each version deleted (see 4.2.4).

3.2.3 Changing the bank name

If we want to keep the history of the sessions, production directories and change the name of a bank, use the option --change-dbname

```
$BIOMAJ_ROOT/bin/biomaj.sh --change-dbname [name] [newName]
```

Propertie db.name don't need to be redefined.

3.2.4 Changing production directories

There are two ways to change the directory production of a bank.
Changing the property data.dir in file global.properties (All banks will have their production in repertory production amended)
Changing the property version.dir file ownership of the bank ([name]. Properties)
Executing the command:

```
$BIOMAJ_ROOT/bin/biomaj.sh –move-production-directories [bankname]
```

If the properties version.dir or data.dir are modified, you can no longer perform updates. You must run the command-move-production-directories.

3.2.5 Rebuilding a version

You can go back a step and rebuild the last version of a databank.

e.g.: `biomaj.sh –rebuild Mybank`

This command is interpreted in two different ways:

-If no version of the source is available locally:

i) The application changes the current bank (version N) status from online to updating. To do this, the symbolic « current » link is erased and a « future_release » link is repositioned in the version directory.

ii) Data other than raw data is deleted

iii) An update cycle is re-run. If the data has not changed on the remote server, this can then re-run post-processing on the raw data and re-deploy.

-If at least two versions of the source are available locally:

During stage i) above, BioMAJ will also re-present version N-1 online during processing of version N. The current link is effectively repositioned on version N-1 and the future_release link on version N.

3.2.6 Error correction

BioMAJ includes different types of error correction, a generally transparent procedure. The command used is the same as that for an update, i.e.:

- `biomaj.sh -d MyBank`

This will start a new session that will attempt to complete the last non-closed cycle for the bank or banks in question. The way the application behaves depends on the error context.

- If an error is produced during the synchronisation stage (network outage, remote server crash, etc.), during the new session, after rebuilding the list of files included in the version, the application will scan the data held locally and try to continue the cycle.

- If an occasionally error happens during a file download, several attempts are made before the download is terminated with errors. The session will be fully stopped after all files in the list are processed.

 **After each download, an integrity check is carried out that includes checking the file's local attributes (name, size and date) against those on the original remote server.**

- If an error occurs during the post-processing stage, when it is re-tried, BioMAJ starts again at the first process with error status.

In the event of post-processing, errors noted are often due to system errors. It is important to note that if the auly process does not return an error (different from zero for shell scripts) during erratic interruption, BioMAJ will not detect it and will continue the session until it is put online. However, to help detect this type of problem, if the new version is smaller than the previous one, the application will transmit a warning.

Data is associated with a version and this version is dependent on the remote server. If a version is incomplete locally when the remote server is updating data, the local version will not be completed. In this event, it is necessary to open a new update cycle as the current cycle cannot be terminated.

To resolve this matter, the option –N or –new has been developed:

- `biomaj.sh --update <dbname> --new`

This command creates a new cycle.

This command must only be used as a last resort after analysis of the situation (reading log files, checking the remote server, etc.) if you are sure that the current cycle will not be completed.

3.2.7 Importing data

The management history of a source is stored in the database. If for any reason the database was to be erased or unusable, all of the history would be lost. To reintegrate this data and regain control of the bank, a command that re-imports the data has been developed.

An example of using this command is below:

```
biomaj.sh --import nr
```

Use of this file implies that the data is coherent with the bank properties file and that the production directory data « conforms » with BioMAJ directory structure organisation:

The property **dir.version** must be placed in the production directory of the bank to be imported. This directory must contain a version directory (*dbname_version*) and a current link pointing to this directory. The version directory must contain a flat directory containing all source data from the server.

`conf/db_properties/nr.properties` (cf. Organising Data)

It is also important to note that:

- Only the data contained in the flat directory (raw data) of the current version will be imported.
- If there are several versions, older versions than the latest one will not be imported.

3.2.8 Database cleanup

Every update cycle is stored in the database, even those who do not result in the update of the databank. As the database grows larger, the removal of these cycles might be useful to save some space. The option –clean-database was developed for that purpose.

The command:

- `biomaj.sh --clean-database <dbname>`

will delete all the sessions and update cycles that ended without error and did not produce new production directories.

3.3 Automating updates

Since version 1.1.0, scheduled updates are directly managed via BmajWatcher.

Still, if you don't want to use the web administration interface, the crontab option is available.

Automatic updates can be carried out using a *crontab*. You first need to know the frequency of source updates you want to automate maintenance for.

Cron is a Unix utility that allows the user to run commands at given intervals.

An example of *crontab* is available in the directory: `$BIOMAJ_ROOT/misc/`

You need to use *crontab* under the BIOMAJ user identity, i.e. the login the application is meant to work under.

User profile files (e.g. .cshrc) are not read before cron runs commands. This can cause strange behaviour in scripts that is sometimes difficult for the layman to understand.

To avoid this problem, before use, you must initialise environment variables required for BioMAJ operation.

A summary of required environment variables follows:

```
#define the shell used when opening a cron session
SHELL=/bin/sh
# position of the log storage directory for cron commands:
HOME=$BIOMAJ_ROOT/log/cron
# position of the BIOMAJ_ROOT variable:
BIOMAJ_ROOT=/MY/RACINE/BIOMAJ
# definition of the java environment:
JAVA_HOME=/usr/local/java/jdk1.6.0
CLASSPATH=/usr/local/java/jdk1.6.0/dt.jar:.
# definition of the ant environment:
ANT_HOME=/usr/ant/apache-ant-1.6.5
```

-To use the crontab, enter the following command:

- `crontab ./BioMAJ_crontab`

-To edit the contents of a crontab, use the command:

- `crontab -e`

This command opens the crontab file `BioMAJ_crontab` in a text editor

For each bank, it is very important to position the local update date depending on how the remote site works. Otherwise, you risk not finishing an update cycle. The reason is simple: the files you want to use will be deleted while you're downloading.

If you place the BioMAJ session at the same time as a remote data update by mistake, BioMAJ does have a monitoring function that will protect you from building a chimeric version. A chimera is a corrupt version made up partly of files from the old version and partly of those from the new source version.

-To add an update:

Just add it to the crontab file. Each line represents a command to be run. The lines have the following format:

```
time        command
```

The commands are simply those that the user would enter if they were running them themselves.

A time is split into five fields:

field	Available values

Minute	0-59
Hour	0-23
Day of the month	1-31
Month	1-12
Day of the week	0-7 (0 and 7 for Sunday) for Linux. For other Unix versions, see the manual for the meaning of these figures.

Each field is separated by a space and several elements of the same type are separated by commas (no space). An asterisk (*) means: "all". E.g. in Linux:

```
0,10,20,30,40,50 * 1 * * /usr/local/BioMAJ/bin/biomaj.sh –d
genbanknews
```

This line means:
« Run /usr/local/BioMAJ/bin/biomaj.sh –d genbanknews every 0, 10, 20, etc. minutes of every hour on the first day of each month ».

Usually, you only give the elements that are pertinent and use an asterisk (*) for the others. You complete a crontab syntax using the command:

- `man -s 5 crontab`

NB: an elegant alternative to using variables is to run a specific script before running the biomaj.sh. executable, e.g.

```
0,10,20,30,40,50 * 1 * 7 $HOME/.bashrc ;
/usr/local/BioMAJ/bin/biomaj.sh –d genbanknews
```

3.4 Warehouse monitoring

BioMAJ has permanent features for monitoring operation. What do we mean by monitoring? This can cover several aspects at different levels:

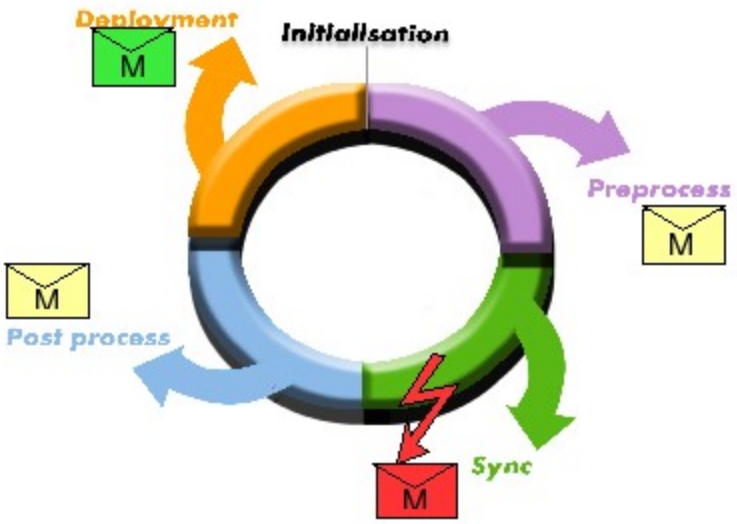
-Knowing the warehouse’s contents;

-Following the progress of a bank;

-Monitoring the progress of an update cycle.

As we will see in this section, three information access methods are available. Depending on the type of information, it can be obtained by e-mail, in console mode or using your browser via an html link.

3.4.1 E-mail alert: monitoring an update session



At the end of each BioMAJ session, a report is sent by e-mail to the data administrator. The person receiving the report's contact details are given by default in the file `global.properties`.

The relevant properties are: **mail.smtp.host**, **mail.admin** and **mail.from**

As their name suggests, these are respectively the address of the mail server, the address of the administrator and the address of the sender.

The e-mail summarises all the information relating to the updated version and the various stages of cycle stages, warnings and errors.

Its subject field has been structured so it can be easily filtered by your e-mail client.

This subject is a variable character string made up of the following fields:

```
BioMAJ message:BANK[$dbname]-STATUS[$status]-UDPATE[TRUE|FALSE]-$Release
```

The subject is therefore linked to the bank and the session results. A session without errors will have a *STATUS[TRUE]* field whereas a session with errors will have a *STATUS[FALSE]* field in its title. If the session production is a version, the string: *UPDATE[TRUE]* followed by the version name will be added to the e-mail subject line.

Here are two examples of e-mail reports:

Date: Thu, 27 Sep 2007 18:35:34 +0200 (CEST)
From: biomjtest@toulouse.inra.fr
To dataman@toulouse.inra.fr
Subject: Biomaj message: BANK [human_genomic] - STATUS [TRUE]- UPDATE [TRUE] RELEASE:2007-10-01

```
Start:05-10-2007 13:04:10
End  :05-10-2007 14:02:04
```

```
****************** INFO RELEASE ***************
Number of session                 :1

Production directory :/db/ncbi/blast/human_genomic/human_genomic_2007-10-01
Release                          :2007-10-01
Download                       :1,995G
Bandwidth (Mo/s)       :3.1467621
Num files downloaded :3
Release                          :4,474G
------------
Processes:
Metaproc:[POST1] log:
[/db/biomaj/Bin/log/human_genomic/20071005130410/postprocess.POST1.log]
 --> fastacmd(Create Fasta File) --> sendMail(mail)

*********************************************
SESSION:
--------
LOG:/db/biomaj/Bin/log/human_genomic/20071005130410/mirror.log
----------- GLOBAL ERROR  -----------------------
---------------------------------------

----------- GLOBAL WARNING --------------------

----------- ERROR ON SUB-TASK -----------------------
            *** preprocess ***
            *** release ***
            *** check ***
            *** download ***
            *** extract ***
            *** addLocalFiles ***
            *** makeRelease ***
            *** postprocess ***
WARNING:  POST1::fastacmd_hg : the  environment  variable  BLASTDB  is not
set. Default value is /db/blastdb.
            *** deployment ***
---------------------------------------
```

Illustration 6: Example of an e-mail alert produced during a correct BioMAJ session with a version update.

Date: Thu, 27 Sep 2007 18:35:34 +0200 (CEST)
From: biomjtest@toulouse.inra.fr
To: dataman@toulouse.inra.fr
Subject: *BioMAJ message: BANK [alu] - STATUS [FALSE]*

```
Start :27-09-2007 18:16:56
End   :27-09-2007 18:17:31
SESSION:
--------
LOG:/home/allouche/BioMAJweb/BioMAJ_dev/log/alu/20070927181656/mirror.log
----------- GLOBAL ERROR  ----------------------
ERROR : BioMAJ stopped.
See the logs to obtain more information.
---------------------------------------
----------- GLOBAL WARNING -------------------
----------- ERROR ON SUB-TASK ----------------------
            *** preprocess ***
            *** release ***
            *** check ***
            *** download ***
            *** extract ***
WARNING:  extract: no match to be decompressed !
            *** addLocalFiles ***
            *** makeRelease ***
            *** postprocess ***

ERROR:    POST1::cptest : /bin/cp: cannot stat
`/bank/ncbi/blast/alu/future_release/flat': No such file or directory
ERROR:    POST1::cptest : /bin/cp: cannot create regular file
`/home/allouche/bank/ncbi/blast/alu/futur_release/testfasta/test.fa': No such
file or directory
ERROR:    POST1::cptest : "Process cptest (with executable cptest.exe) generate
an error."
read   log  files to obtain  more information.
---------------------------------------
```

Illustration 7: Example of an e-mail alert produced during a problematic BioMAJ session.

3.4.2 Debugging and error identification during running

Session logs can be found by:

- Seeing the html report (see 3.4.4)
- Viewing files:
  - `$BIOMAJ_ROOT/log/[bankname]/[datesession]/mirror.log`: initialisation, synchronisation, deployment
  - `$BIOMAJ_ROOT/log/[bankname]/[datesession]/[preprocess|postprocess]-[groupe-process].log`: log of an defined process log for post-processing (see 4.2.3)

 You can get continuous information on workflow status by running this command in a shell:

> tail -f log/[bank]/[date]/mirror.log

3.4.3 Exploring in the command line: biomaj.sh --status

3.4.3.1 Exploring the warehouse

- The command: *biomaj.sh --status* lists the sources managed by the application.

If the list is relatively long, this command has an option for filtering to extract a sub-group from the entire catalogue.

Several filters are available:

--dbtype <dbtype>	Sources with the given dbtype.
--online	Sources not currently updating.
--updating	Sources having a running update cycle.

The –status command can be used as follows:

- `biomaj.sh -S --dbtype genome`

```
DbType          DbName                        Last release       Date Session Status
-------------------------------------------------------------------------------------
genome          Anopheles_gambiae            2007-06-01         15-06-2007   online
                Apis_mellifera                23-01-2007          21-03-2007   online
                Arabidopsis_thaliana         2007-04-24          28-04-2007   online
                Bacteria                     2007-08-10         22-08-2007   online
                Bos_taurus                   2007-04-25         11-05-2007   online
                Caenorhabditis_elegans        16-02-2006          21-03-2007   online
                Canis_familiaris              23-01-2007          21-03-2007   online
                D_rerio                       01-03-2007          21-03-2007   online
                Drosophila_melanogaster       23-01-2007          21-03-2007   online
                Fungi                         2007-07-12          20-08-2007   online
                Gallus_gallus                 23-01-2007          21-03-2007   online
                H_sapiens                     2007-04-17          20-04-2007   online
                M_musculus                    2007-07-05          21-08-2007   online
                Macaca_mulatta                23-01-2007          23-03-2007   online
                Monodelphis_domestica         06-03-2007          23-03-2007   online
                Pan_troglodytes               02-03-2007          23-03-2007   online
                Plasmodium_falciparum         2007-07-24          16-08-2007   online
                R_norvegicus                  23-01-2007          21-03-2007   online
                Saccharomyces_cerevisiae      2007-08-13          20-08-2007   online
                Schizosaccharomyces_pombe     05-03-2002          23-03-2007   online
                Strongylocentrotus_purpuratus 23-01-2007          02-04-2007   online
                Tribolium_castaneum           25-01-2007          23-03-2007   online
```

Illustration 8: Result of status command

In the above example, dbtype filtering displays all sources maintained in the local catalogue classified as « genomes ».

The results table has five fields:

- **Dbtype**: source classification field
- **Dbname**: generic name given to the source
- **Last release**: name of the latest source version
- **Session Date**: update date
- **Status**: version status (online or updating: if the last cycle is open or closed)

 Note that it is possible to combine filters:

- `biomaj.sh -S --updating –dbtype=genome`

will give a list of genomes being updated.

3.4.3.2 Exploring the status of a bank

We have just seen that warehouse contents can be displayed with the command:

- `$BIOMAJ_ROOT/bin/biomaj.sh --status`

Detailed information on each bank maintained by BioMAJ can be obtained with the command:

- `$BIOMAJ_ROOT/bin/biomaj.sh --–status [bankname]`

When a bank name is given, the option *--status* displays four blocks of data: the first contains the configuration attributes (user-defined properties file), the second includes all data associated with the current version, the third relates to the version being updated. The fourth block lists the

bank versions in production.

 The blocks « *current release* », « *future release* » and « *list of production directories* » are optional. Their presence depends on the contents of the local data warehouse.

An example of a report is given in the following section, relating to a bank with a version in production (*current*) and a version being updated (*future release*):

```
#----------------
#          Properties          used          For          the          last          session
#----------------
Keyname                                                                    :estMM8_HG18_CANFAM2
Description                                        :"EST  :  Dog  (Canis  familiaris),  Human
(Homo               Sapiens),               Mouse               (Mus                musculus)"
#----------------

First    utilisation                                                  :22-09-2007    00:00:13
Url                              :ftp://hgdownload.cse.ucsc.edu//goldenPath
Remote     regular     expression                                :canFam2/bigZips/est.fa.gz
hg18/bigZips/est.fa.gz                                                  mm8/bigZips/est.fa.gz
Remote                       excluded                      regular                      expression:
Local          regular          expression                                                    :.*
Version     directory                                                  :/db/estMM8_HG18_CANFAM2
Offline    directory                                    :/db/biomaj_tmp/estMM8_HG18_CANFAM2_tmp
Log files on state file          :true
```

```
#----------------
#                                    Current                                    Release
#----------------

Release                                                                        :2007-08-31
Number      of      session                                                           :13
Last    session                                                        :03-09-2007    11:45:00
Duration                                                                    :69:55:45:00
Production         directory
:/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-08-31
Num        files        downloaded                                                     :3
Bandwidth      (Mo/s)                                                       :0.20482694
Download     size                                                              :2,270G
Bank     size                                                               :15,803G

View     Last     log                                                  :tail     -f
/db/biomaj/Bin/log/estMM8_HG18_CANFAM2/20070903112705/mirror.log

--------------
Unparseable                                  date:                                    ""
<process  args="'fasta/*.fa  flat/*/*/*.fa'  '-p  F  -o  T  -n  est_dmh  -t   est_canfam_mm8_hg18'"
biomaj_error="false"                  desc="Index                  blast"
elapsedTime="00:31:23"  exe="formatdb.bash"  keyname="formatdb"  name="formatdb"  start="03-09-2007
10:45"                  type="index"                       value="-1"                       >
Unparseable date: ""

<process  args="'fasta/*.fa  flat/*/*/*.fa'  '-p  F  -o  T  -n  est_dmh  -t   est_canfam_mm8_hg18'"
biomaj_error="false"                  desc="Index                  blast"
elapsedTime="00:00:00"  exe="formatdb.bash"  keyname="formatdb"  name="formatdb"  start="03-09-2007
10:32"                  type="index"                       value="-1"                       >


Processes                                    by                                    metaproc:
Metaproc:[P1]
log:[/db/biomaj/Bin/log/estMM8_HG18_CANFAM2/20070903112705/postprocess.P1.log]
-->concat(concatenation      des      fichiers      est      au      format      fasta)
-->formatdb(Index blast)
```

```
#----------------
#                                     Futur                                     Release
#----------------
Release                                                                        :2007-09-23
Number      of      session                                                           :14
```

```
Production         directory
:/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-09-23
Num        files        downloaded                                                    :3
Bandwidth      (Mo/s)                                                          :0.07028251
Download      size                                                              :2,270G
Bank      size                                                                      :0K

View     Last     log                                                    :tail     -f
/db/biomaj/Bin/log/estMM8_HG18_CANFAM2/20071001161020/mirror.log
--------------
Processes                                    by                                metaproc:
```

```
#                        List                          production                        directories
20-06-2007                                          15:51:00
/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-06-16                                  (6,512G)
03-09-2007                                          11:45:00
/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-08-31                                  (15,803G)
```

Illustration 9: Result of the status command for a databank (biomaj.sh –S alu)

As you can see in the above example:

-a block of information is linked to the configuration and contains:

- The name of the bank
- Its description (corresponding to **db.fullname**)
- The first session date
- The download url
- Files to be downloaded (regular expression)
- Files to be moved in production (regular expression)

-Then two blocks associated to current and future Releases containing:

- The release
- The last session date
- The last release's online time
- The production directory
- The number of downloaded files
- The bandwidth
- The size of downloaded files
- The size of files in the production directory
- The number of files in production
- The number of meta-processes applied to the workflow
- The name of the session logfile.

-The last block contains:

- A list of versions available locally

3.4.4 Html report

See BmajWatcher documentation.

3.5 Multitenancy

Multitenancy is introduced in BioMAJ since version 1.2 through the web administration interface. Several users can have access to different content depending on their privileges. The main change introduced by that system is that each bank belongs to a user :

- A bank is associated to a user in the database
- The banks properties are saved under a directory named <user_login> in workflows.dir

So far the workflows.dir looked like :

```
<workflows.dir>
      bankA.properties
      bankB.properties
      bankC.properties
      bankD.properties
```

Now :

```
<workflows.dir>
      user1
            bankA.properties
      user2
            bankB.properties
      user3
            bankC.properties
            bankD.properties
```

These changes do not affect the behaviour of BioMAJ. It is only the view offered by BW that is impacted. It means that if you run an update via the console, these notions of bank ownership are ignored.

Further information on multitenancy are available on the BW documentation.

4 Workflow creation

4.1 General information

BioMAJ allows you to download a source, maintain data and run processes on this data (warnings, indexing, reformatting, processes for logs, analysis, etc.). The workflow that follows this processing can be configured from a single file called a **properties file.** BioMAJ includes a set of pre-defined properties files for a wide range of known biological banks. This chapter covers the contents and structure of properties files to help you create new ones.

Properties files should be located in the directory `$BIOMAJ_ROOT/conf/db_properties` to be interpreted by BioMAJ.

4.2 Bank update session report

A properties file contains the attributes for the various stages in an update cycle. Its contents shows the update cycle.

As we have seen, the update cycle is made up of five stages:

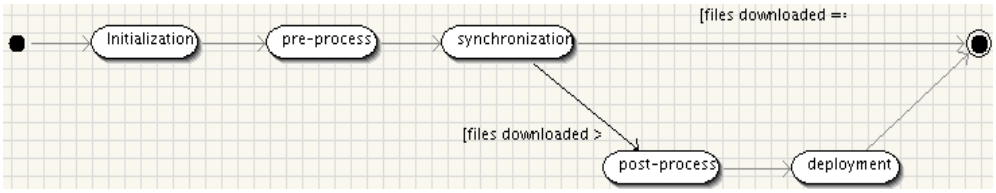


Illustration 10: Stages in the online cycle shown sequentially

(1) Initialisation: creates directories in the case of a first session and checks the consistency of defined properties for the workflow.

(2) Pre-process: runs pre-processes defined by the administrator in a workflow.

(3) Synchronisation: gathers the bank version number and downloads new files

(4) Post-process: runs a sub-workflow of all post-processes defined by the administrator.

(5) Deployment: places the bank version online and deletes obsolete versions.

Stages 4 and 5 are run if at least one file has been downloaded or deleted from the previous version. If the version is identical to the remote data, the update cycle does not produce a new version.

Another stage is run when a bank version is deleted during deployment. It is a remove process defined in a workflow by the administrator.

4.2.1 Configuring the source: description / classification / location

4.2.1.1 Description and classification

The first configuration elements in a bank's properties describe it:

e.g. alu.properties

```
db.fullname="alu.n: alu repeat element. alu.a: translation of
alu.n repeats"
db.name=alu
db.type=nucleic_protein
db.formats=fasta,ncbi
```

- **db.name**: short bank name.
- **db.fullname**: full bank name.
- **db.type**: type of classification.
- **db.formats**: list of raw data formats that make it up.

4.2.1.2 Location of data

The following properties define the bank information linked to its location in the local warehouse.

data.dir=/bank/test
dir.version=ncbi/blast/alu
offline.dir.name=BioMAJ/ncbi/blast/alu_tmp

- **data.dir** : root directory of the repository. Often defined globally in the *global.properties* file for all sources
- **dir.version**: root directory for managing bank versions (**db.name** by default).
- **offline.dir.name:** temporary directory used locally for downloading and extracting files (**BioMAJ_tmp/db.name** by default).

4.2.2 Configuring the data Synchronisation stage

This stage is split into sub-tasks:

1)Gather the bank version number.
2)Create a list of files relating to a version (interrogate remote server).
3)Create a list of files to be downloaded and a list of files to gather from an old version.
4)Download and copy files listed in stage 3 in a temporary directory.
5)Extract compressed data from the download.
6)Create a production directory.
7)Move temporary files (forming a new version) to the new production directory.

These sub-tasks use the properties mentioned above for naming and location and new variables:

- Server address (**server**)
- Protocol (**protocol**)
- Root directory of the bank on the server (**remote.dir**)
- Regular expressions for remote files to be downloaded (**remote.files**)
- Regular expressions for remote files to exclude from the download: (**remote.excluded.files**)
- Regular expressions for local files for applying processing (**local.files**)
- File containing the bank version number (**release.file**)
- Regular expression for the version number (**release.regexp**)

4.2.2.1 Remote server and download protocol

Connecting to the remote server involves defining its properties: **protocol, server and remote.dir.**

You can configure synchronisation of the source with three download protocols: *ftp, http, rsync* and a local mode. The variable r**emote.dir** is the remote directory from which the download is taking place.

4.2.2.1.1 FTP protocol

Here is an example for configuring the download of FASTA files from the ALU bank to the NCBI (ftp://ftp.ncbi.nih.gov/blast/db/FASTA/):

```
#ftp mode
protocol=ftp

#ftp server
     server=ftp.ncbi.nih.gov

#download root directory
remote.dir=/blast/db/FASTA/
```

In some cases, if the remote server is slow and/or overloaded, you can adapt the connection by adding to the properties relating to the detection of a connection loss and the number of retry attempts if communication with the server is lost.

```
#Time out during synchronisation stage (~20 minutes)
 ftp.timeout=1000000
#Number of reconnections to the server during synchronisation
ftp.automatic.reconnect=5
```

The property **ftp.timeout** can equal -1, in which case no timeout will be defined. The time unit is milliseconds.

If the value of **ftp.timeout** is too small, the connection will not complete. An error "java.net.SocketTimeoutException: Read timed out" will appear.

Properties associated with the download task:

```
#Number of download threads
files.num.threads=3
```

These properties are also valid for the http protocol!

The property **file.num.threads** defines the number of parallel downloads authorised by the application.

4.2.2.1.2 HTTP protocol

HTTP protocol is special because it is fairly hard to preview html pages and therefore the attributes of the file and remote server. BioMAJ allows you to redefine a regular expression for parsing a directory (**http.parse.dir.line**) or file (**http.parse.dir.line**) and select groups for gathering attributes (**http.group.dir.name, http.group.dir.date, http.group.file.name, http.group.file.date, http.group.file.size**).

```
#http mode
protocol=http

#http server
server=astral.berkeley.edu

#root download directory
remote.dir=/pdbstyle-1.71
```

```
#regular expression for parsing a directory
http.parse.dir.line=<img[\\s]+src=\"/icons/folder.gif\"[\\s]+alt=\"\\
[DIR\\]\".*href=\"([\\S]+)/\".*([\\d]{2}-[\\w\\d]{2,5}-[\\d]{4}\\s[\\d]{2}:
[\\d]{2})

#regular expression for parsing a file
http.parse.file.line=<a[\\s]+href=\"([\\S]+)\".*([\\d]{2}-[\\w\\d]{2,5}-[\\d]
{4}\\s[\\d]{2}:[\\d]{2})[\\s]+ ([\\d\\.]+[MKG]{0,1})

#directory name matches group 1 contained by http.parse.dir.line
http.group.dir.name=1

#directory date matches group 2 contained by http.parse.dir.line
http.group.dir.date=2

#file name matches group 1 contained by http.parse.file.line
http.group.file.name=1

#file date matches group 2 contained by http.parse.file.line
http.group.file.date=2

#file size matches group 3 contained by http.parse.file.line
http.group.file.size=3
```

This example shows parsing of an html page from the following url: http://astral.berkeley.edu/pdbstyle-1.71.

It includes the following link that matches directory 0g dated 31-Oct-2006 07:52:

```
<img src="/icons/folder.gif" alt="[DIR]"> <a href="0g/">0g/</a>           31-Oct-
2006 07:52-
```

The property **http.parse.dir.line** contains the regular expression for parsing this line. The value **http.group.dir.name** (1) gets the first bracketed group from **http.parse.dir.line** (([\\S]+/) is equal to « Og » in our example) and the value of **http.group.dir.date** (2) gets the second bracketed group from **http.parse.dir.line** (([\\d]{2}-[\\w\\d]{2,5}-[\\d]{4}\\s[\\d]{2}:[\\d]{2}) equals « 31-Oct-2006 07:52 »).

In the same way, **http.parse.file.line** recognises the following html link:

```
<img src="/icons/unknown.gif" alt="[ ]"><a href="d10gsa1.ent">d10gsa1.ent</a>28-Oct-2006 00:00  82K
```

The main restrictions for using http with BioMAJ are:

- If a directory exists, the name and creation date of this directory must be present.
- The directory name and creation date must be on the same line.
- The name, date and file size must be present.
- The name, date and file size must be on the same line.
- A date must be recognised by one of these three regular expressions:
  - [\\d]{2}-[\\d]{2}-[\\d]{4}\\s[\\d]{2}:[\\d]{2} (*e.g.: 02-04-2006 02:00*)
  - [\\d]{2}-[\\w]{3}-[\\d]{4}\\s[\\d]{2}:[\\d]{2} (*e.g.: 02-Apr-2006 02:00*)
  - [\\d]{2}-[\\d]{2}-[\\d]{4}\\s[\\d]{2}:[\\d]{2}:[\\d]{2} (*e.g.: 02-04:2006 02:00:00*)
- A size must be recognised by the following regular expression:
  - [\\d]+ ((\\.|,)[d++])? (G|M,K)? (ex: 82K ; 82 ; 1,2M ; 1.2M)

You can also redefine some properties for downloading: **file.num.threads** (see ftp protocol above).

4.2.2.1.3 Direct HTTP protocol

That protocol is a variation of the HTTP protocol. It allows one to download the content of a dynamic web page. Hence, with this protocol you can only download one « file ».
Some parameters have been added to build the URL BioMAJ will retrieve the content from.

Example to retrieve the content of : www.biomart.org/biomart/martservice?query=some_xml

```
protocol=directhttp

server=www.biomart.org
remote.dir=biomaj/martservice

url.method=POST
url.parameters=query
query.value=some_xml

target.name=output.txt
```

The parameter `target.name` is the name of the file the downloaded content will be saved under. If not defined, the default value will be `data`.

The last modification date and the size of the page content are handled differently from the other protocols. These values are retrieved from the page http header.
The release number is built from the « Last-Modified » parameter.

The size is retrieved from the « Content-Length » parameter.

4.2.2.1.4 Sftp protocol

Sftp uses mostly the same parameters as ftp. `username` and `password` parameters are required for the authentication.

```
protocol=sftp
server=test.server.com
port=22
username=test
password=test
remote.dir=/test/banks
files.num.threads=2
```

4.2.2.1.5 Amazon S3 protocol

You can download files from the cloud with BioMAJ. It works for any S3 based storage service. It has been tested against the amazon servers but also an EucalyptusWalrus server.
The parameters used are SFTP ones :

```
protocol=s3
server=my.walrusserver.com/services/Walrus
port=8773
username=id
password=key
remote.dir=/bucket1
remote.file=.*
```

To access the root, specify `remote.dir` as '/'. You will then be able to specify any bucket in `remote.files`.

Additional properties for S3 configuration are available in $BIOMAJ_ROOT/jets3t.properties.

4.2.2.1.6 Rsync protocol

Example of enzyme bank download on bio-mirror with rsync.

```
#rsync mode
protocol=rsync

#rsync server
server=bio-mirror.net

#root download directory
remote.dir=/biomirror/enzyme/
```

Operation is similar to other protocols. For the time being, you cannot specify options for the rsync utility.

Rsync protocol is only used as a downloading protocol. Management of the difference between the mirror and the production directory is carried out by the data synchronisation stage in BioMAJ.

4.2.2.1.7 Local protocol

```
# local mode (copy files on the local machine)
protocol=local
# one value is valid: localhost for this mode
server=localhost
#directory containing the data
remote.dir=/local/
```

This protocol can be very useful for generating a bank version from the data already present on the server that is hosting BioMAJ.

This protocol can help define process workflows for local data and supervise and monitor processes using BioMAJ functions.

4.2.2.2 Building download filters (remote.files/local.files)

Tasks **2)** (creating a list of files on a remote version) and **3)** (creating a list of files to be downloaded and getting a remote version) from the synchronisation stage use regular expressions in java (for more information, see SBIOMAJ_ROOT/doc/regexp.pdf).

These expressions are used to define the flow of files for later tasks (**4), 5)** and **6)**). These two flows relate to files to be transferred from the remote server (**remote.files** property) and to the files to be transferred from the temporary directory to the future bank version directory (**local.files** property).

Using regular expressions, you can tell BioMAJ the files that need processing. You can thus just download a few files on a server and exclude others.

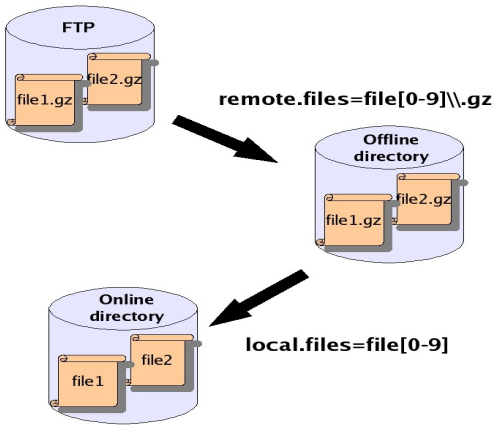


Illustration 11: Definition of remote.files and local.files properties

Example defining the bank PFam:

```
remote.files=^Pfam.*\\.gz$    ^swisspfam\\.gz$     ^version.*$
^pfamseq\\.gz$

local.files=^Pfam.*$ ^swisspfam.*$ ^version.*$ ^pfamseq.*$
```

In the above example:

The regular expression remote.files selects files on the remote server with the name: Pfam.*.gz , swissfam.*.gz , version.*, pfamseq.gz

Then the regular expression local.files selects files with the name: Pfam.*, swissfam.*, version.*, pfamseq.* to create a list of files to be replaced.

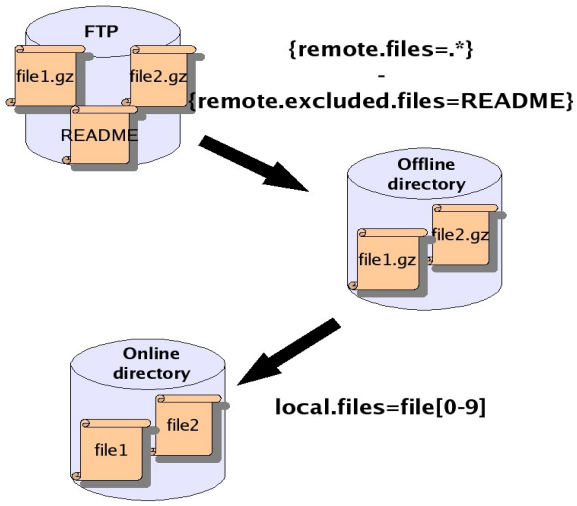


Illustration 12: Using the property remote.excluded.files

It is sometimes easier to say that you want to download everything except certain files. There is a third property, therefore, (**remote.excluded.files**) that will exclude unwanted files from the property **remote.files.**

Here is another example taken from the workflow « Genome Anopheles gambiae (NCBI) » (*Anopheles_gambiae.properties*). This example shows how to obtain specific sub-directories:

```
#files to be downloaded from server
remote.files=^CHR_[\\w]+/.*$ ^maps/mapview/.*$ ^SNP/.*$
#files to be excluded from the transfer
remote.excluded.files=.*\\.asn.*
#files to be transferred to the bank version directory
local.files=[\\w]+$ [\\w+]/[\\w]+$ [\\w+]/[\\w+]/[\\w]+$
```

The regular expression above selects directories beginning with CHR and the files they contain as well as the contents of the directory maps/mapview . Note that *.asn files are not downloaded (cf. **remote.excluded.files**).

One example of a production directory result is:

${dir.version}/current/flat/:

- CHR_2/
  - NC_009071.gbk, NC_009071.faa,...
- CHR_3/
  - NC_009072.gbk,NC_009072.faa,...
- CHR_MT/
  - NC_002084.gbk,NC_002084.faa
- CHR_X/
  - NC_004818.gbk,NC_004818.faa

- maps/
  - mapview/
    - cyto.md,cytosat.md,cytoscf.md,...
- SNP/
  - mosquito_snp_20020625,mosquito_snp_readme.txt

(*.asn files are not downloaded)

4.2.2.3 Getting a version number

For each cycle, BioMAJ tries to get or create a version number at the start of the file synchronisation stage.

There are three ways of getting this number:

- By default, the version number matches the date of the most recent file from the list of files to be downloaded.
- The version number is contained in a file (**release.file**, **release.regexp** and **release.file.compressed** must be defined).
- The version number is contained in the file name (**release.regexp** must be defined).

Some examples:

```
#GenbankRelease: number in the file GB_Release_Number
release.file=GB_Release_Number
release.regexp=[\\d]+
release.file.compressed=false
```

The result of this search gets numbers matching the version number of *genbank* from the file *GB_Release_Number*.

Another example with the enzyme bank:

The file containing the version number is enzuser.txt

Regular expression is a date in the following format:24-Jul-2007

```
#Enzyme: date in the file enzuser.txt
release.file=enzuser.txt
release.regexp=[0-9]{2}-[\\w]{2,5}-20[0-9]{2}
release.file.compressed=false
```

You can also get the version number that is not necessarily on the server holding the data. The file containing the enzyme bank version is found via a protocol and a different location.

```
    #ftp                protocol                for                download
protocol=ftp
server=ca.expasy.org
remote.dir=/databases/enzyme/


    #you can use another protocol than that used for download, e.g. with rsync,
etc.
release.file=rsync://bio-mirror.net/biomirror/enzyme/enzuser.txt.Z
release.regexp=[0-9]{2}-[\\w]{2,5}-20[0-9]{2}
release.file.compressed=true


    #files       to       download       from       the       ftp       server
remote.files=.*\\.txt$                  .*\\.dat$                  .*\\.get$
remote.excluded.files=
```

Since version 0.9.3.0, it is possible to recover a part of the regular expression through the parenthesis:

Example with UniProt :

```
release.file=reldate.txt
#On récupère seulement le groupe parenthése
release.regexp=UniProt\\sKnowledgebase\\sRelease\\s+([\\d]+\\.*[\\d]*)
```

In a large number of the instances shown in this chapter, an understanding of regular expressions in java is indispensable. For more information on this syntax, see `$BIOMAJ_ROOT/doc/regexp.pdf`

4.2.2.4 Viewing data

Data consolidation is intended to gather all data locally in the production directory.

The download stage requires a list of files from the version on the remote server to be generated. This list is created during stage 2 and will define a list of files to be downloaded and a list of files to be gathered locally (stage 3).

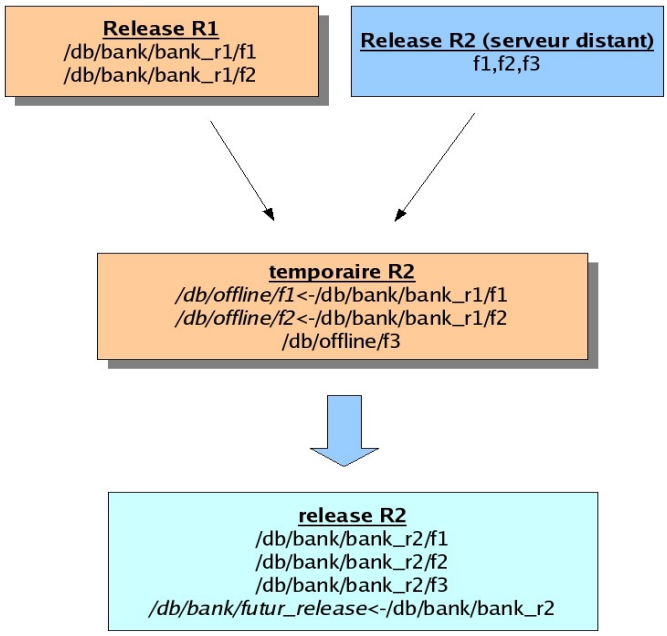


Illustration 13: Synchronisation stage

Several properties are involved in the process

remote remote.files= regular expression
remote.excluded.files= regular expression

local.files= regular expression ➔ filter files to be moved from the directory offline.dir.name to the release flat directory.
No.extract=true| false

offline.dir.name = temporary work directory

do.link.copy=true|false

When files are downloaded, these files are extracted to a temporary directory (**offline.dir.name*)***. The property **no.extract** (true|false) tells BioMAJ to go to the extraction stage.

Files defined using **remote.files and remote.excluded.files,** which are in common with the new and old version, are copied or symbolically linked in the temporary work directory during the task **versionsmanagement** .

(The property **do.link.copy=true** is for choosing between the link and the copy).

Then the task **versionsmanagement** creates in the production directory (**dir.version)**, the directory for the new version. The name of the new version is determined by the properties definition shown in 4.2.2.3. Finally, **versionsmanagement** positions the link *future_release* in the newly created directory.

To finish the task, **move** ends data consolidation. It moves all or some of the files from the temporary directory to the directory future_release/flat. The list of files to be moved is determined by the regular expression given in the property **local.files.**

At this stage, the raw data from the new release is available locally for carrying out the next stage. The post-process can then begin.

When there are a large number of source files (several thousand if these files are not tar, zip or rar archives or similar), activating the property log.files =false will considerably reduce the size of the xml log file and thus optimise application operation.

4.2.3 Configuring the pre, post & remove processing stages

4.2.3.1 General operation

Raw data is available in data.dir/dir.version/future_release/flat.

BioMAJ defines relatively complex post-processing topological workflows. These definitions use three types of element: blocks, meta-processes and processes. Pre-processing and remove processing use two of these elements: meta-processes and processes. Blocks support for pre and remove processing might be implemented in a future version.

The workflow is a series of blocks that are carried out in order. A block contains one or more meta-processes. Meta-processes in a block are run in parallel.

A meta-process contains a list of processes (jobs, processes or warnings, etc.) that are carried out in order. This structure defines a complex workflow represented by a directed acyclic graph or DAG. Sequential and parallel alternation is useful for process synchronisation (« meetings » between processes) and running processes at the same time to make the most of current machine performance (multiprocessors and clusters).

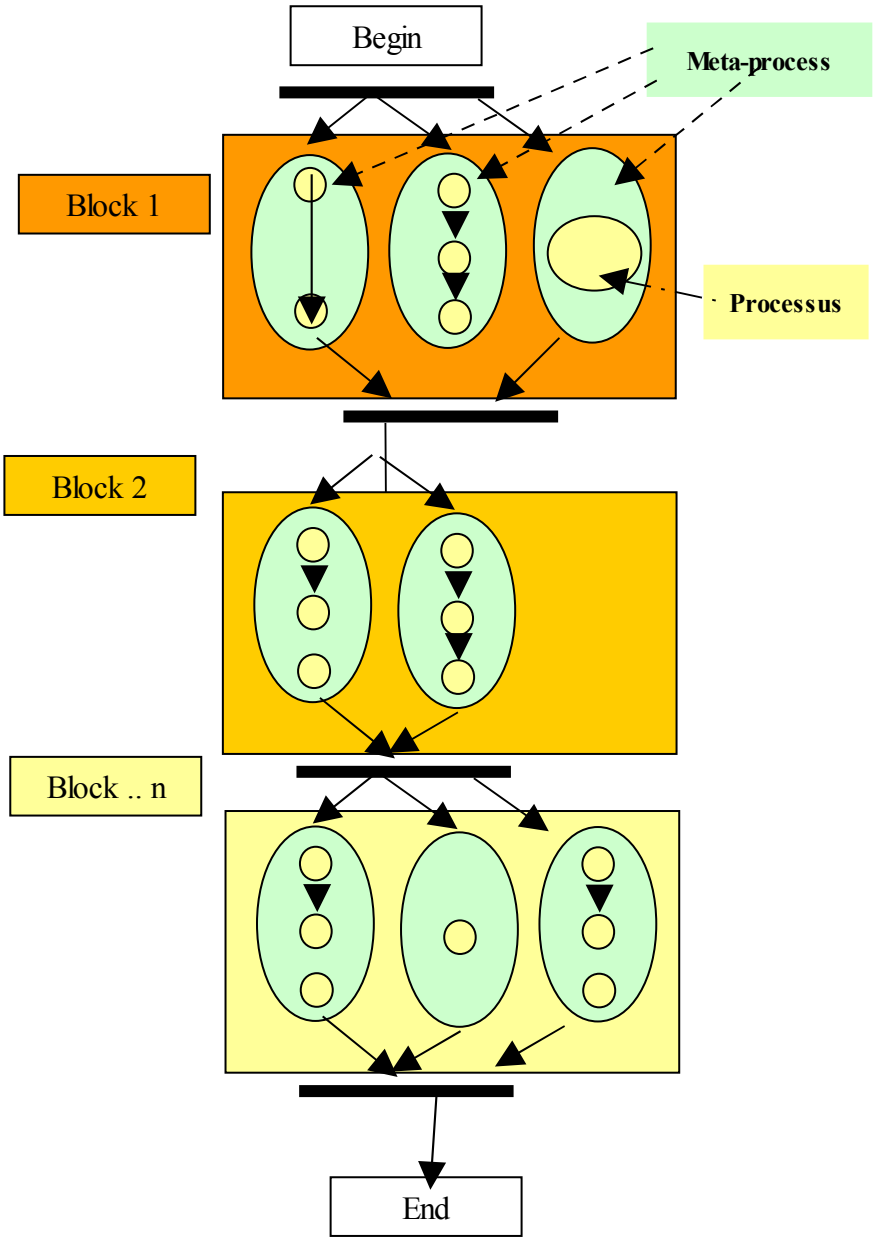


Illustration 14: schematic diagram of workflows for pre- and post-processes: dependencies between blocks, meta-processes and processes in BioMAJ.

These stages help run a range of processes (synchronous or asynchronous) before (pre-processing) or after (post-processing) making the bank available, or after (remove processing) deleting a bank version. These processes can be informative (sending an e-mail) or create indexes, provide format conversions for raw data, extract data or create reports and so on.

4.2.4 Defining elements

4.2.4.1 Block

A block is defined by a name. The list of post-processing blocks is given with the property

BLOCKS.

e.g.

*BLOCKS= **A,B,C***

In this example, three blocks A, B and C are defined in the workflow. Their order determines the order in which they are run.

The property BLOCKS must be given in capital letters !

4.2.4.2 Meta-process

•Declaration

A block's meta-processes are given in a property that describes the block name followed by **.db.post.process**

***A**.db.post.process=META1,META2*

***B**.db.post.process=META3*

***C**.db.post.process=META4*

In the above example, four meta-processes are defined in the workflow:

- 2 in block A (META1, META2)
- 1 in block B (META3)
- and the meta-process META4 is given in block C.

Note that for pre-processing and remove-processing, meta-processes are defined using db.pre.proccess and db.remove.process respectively (without any block prefix).

•***Initialisation***

Then the processes for each meta-process are given

e.g.

***META1**=P1,P2*

***META2=**Z1*

***META3**=P3,test7*

***META4**=message*

In this example:

2 processes P1 and P2 are given in META1. The process Z1 is declared in META2. META3 contains two processes, P3 and test7. META4 contains one process called « message ».

Each meta-process and process has a unique name. It is made up of a freeform character string ([a-z,A-Z,0-1]) without spaces. The process declaration order defines the order in which they are run in the meta-process.

4.2.4.3 Process

Then each declared process needs to be defined in the meta-processes:

Each process is defined by six attributes: name, executable, arguments, description, type and cluster. Each attribute is defined in a property named after the process it refers to. The property extension is fixed:

- ***process*.name:** process name
- ***process*.desc:** process description
- ***process*.type**: process type
- ***process*.exe:** absolute path of an executable (or relating to $BIOMAJ_ROOT/conf/process)
- ***process*.args:** executable parameters
- ***process*.cluster:** Can be "true" of "false". If enabled; the process will be submitted to the default queuing system. This property is not mandatory. Its default value is "false".**It is assumed that your queuing system is correctly configured and that the possibly required environment variables are set**.

So for a sample process <MYprocess> we have:

*MYprocess.**name=** MYprocess*

*MYprocess.**desc= test process***

*MYprocess.**type=test***

*MYprocess.**exe=Myshell.sh*** ***# this script must be placed in the directory***

 # $BIOMAJ_ROOT/conf/process

*MYprocess.**args= -a***

*MYprocess.**cluster=true***

It is possible to give properties as argument.

Example :

```
print.name=echo
print.exe=echo
print.args=Bank : ${db.name} VERSION : ${dir.version} RELEASE : ${remote.release}
print.desc=Affichage des proprietes du workflow
print.type=Affichage
```

Will print:

Bank : alu VERSION : TEST RELEASE : 2008-01-31

If we consider the example given above, 7 processes must be defined: *P1,P2,Z1,P3,test7,message*

The bank properties file will contain 35 declarations (5X7).

P1 ➔	P1.name, P1.desc, P1.type, P1.exe, P1.args
P2 ➔	P2.name, P2.desc, P2.type, P2.exe, P2.args

Z1 ➔	Z1.name, Z1.desc, Z1.type, Z1.exe, Z1.args
P3 ➔	P3.name, P3.desc, P3.type, P3.exe, P3.args
Test7 ➔	test7.name, test7.desc, test7.type, test7.exe, test7.args
Message ➔	message.name, message.desc, message.type, message.exe, message.args
Z1 ➔	Z1.name, Z1.desc, Z1.type, Z1.exe, Z1.args

Note that the order of declaration is not important. They must however all be declared. Two processes can use the same executable.

4.2.5 Workflow example from scripts available in BioMAJ

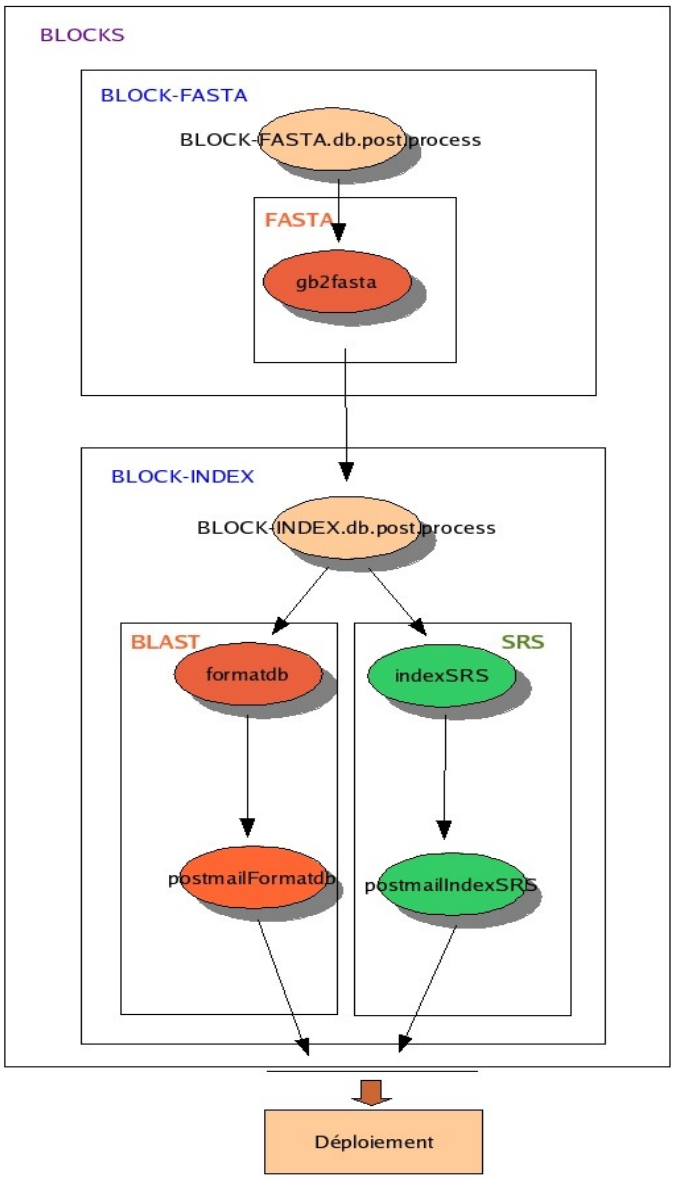


Illustration 15: Running processes with BioMAJ

The example uses four post-processes:

All scripts are distributed with the application:

- formatdbTLSE.pl: format blast bank

  Documentation: $BIOMAJ_ROOT/doc/process/formatdbTLSE.html
- gb2fasta.sh: generate fasta file from blast indexes
- indexSrsTLSE.pl: format banks for SRS

Documentation: $BIOMAJ_ROOT/doc/process/indexSrsTLSE.html

- sendMailTLSE.pl: send e-mail.

Documentation: $BIOMAJ_ROOT/doc/process/sendMailTLSE.html

Configuring the properties file:

```
#Definition des blocs, qui seront exécutés (exécution séquentielle)
BLOCKS=BLOCK-FASTA,BLOCK-INDEX
#Le bloc BLOCK_FASTA est composé d'un meta-process
BLOCK_FASTAdb.post.process=FASTA
#Le bloc BLOCK_INDEX est composé de deux méta-processus qui seront exécutés en parallèle
BLOCK_INDEX=SRS,BLAST
```

```
#Meta-processes define all processes (run sequentially)
FASTA=gbtofasta
BLAST=formatdb,postmailFormatdb
SRS=indexSRS,postmailSRS
```

```
#DEFINING PROCESSES FOR FASTA
#-------------------------------------------------------
gbtofasta.name=gb2fasta
gbtofasta.exe=gb2fasta.sh
gbtofasta.args=
gbtofasta.desc=Index blast
gbtofasta.type=index
```

```
#DEFINING PROCESSES FOR BLAST
#-------------------------------------------------------

#define the process formatdb
formatdb.name=formatdbTLSE
formatdb.exe=formatdbTLSE.pl
formatdb.args= '*.seq' '.seq' gb
formatdb.desc=Index blast
formatdb.type=index

#define the process postmailFormatdb
postmailFormatdb.name=sendMail
postmailFormatdb.exe=sendMailTLSE.pl
postmailFormatdb.args=-s '[EBI - db.name remote.release] End Post Process formatdb' -m 'local.time'
```

```
postmailFormatdb.desc=mail
postmailFormatdb.type=info
```

```
#DEFINING PROCESSES FOR SRS
#-----------------------------------------------


#défine process indexSRS
indexSRS.name=indexSRS
indexSRS.exe=indexSrsTLSE.pl
indexSRS.args=-v -d genbankrelease --pvm --execute pbs -c 6
indexSRS.desc=Index srs
indexSRS.type=index


#define process postmailSRS
postmailSRS.name=sendMail
postmailSRS.exe=sendMailTLSE.pl
postmailSRS.args=-s '[EBI - db.name remote.release] End Post Process formatdb' -m 'local.time'
postmailSRS.desc=mail
postmailSRStype=info
```

4.2.6 Deployment

This stage involves the following operations:

- Delete the link *future_release* (used after the synchronisation stage and by the post-processing stage)
- Apply the rights defined by the property **production.directory.chmod** to the new version.
- Create or move the link *current_release.*
- Delete obsolete versions, i.e. those above the property **keep.old.version**
- For each deleted version, remove processes will be launched

e.g.
if **keep.old.version = 0** then the current version will be deleted after a new version has been built.
If **keep.old .version =1** then the programme will keep two versions. The current version and the previous one.

 **Given how the BioMAJ cycle works, it is important to note that just before deployment,**

a version is present in the directory future_release. To work, even if keep.old.version = 0, BioMAJ will need free space equal to the size of both versions!

4.3 Information overload: global.properies

The properties files **global.properties**[2] is used to define information common to all banks maintained by the application. For the needs of a specific source, you can redefine some properties in the bank properties file to adapt BioMAJ's behaviour to the context.

Engine parameter:

- Number of workflows to run in parallel (**bank.num.thread**): BioMAJ can run workflows in parallel during the same session, this property limits the number of banks accessing remote servers.

Log generation:

- Generate task name in logs (**historic.logfile.task**)
- Generate meta-task name in logs (**historic.logfile.target**)
- Generate all ant properties in a workflow (**historic.logfile.properties**)
- Generate logs corresponding to a daily update level (**historic.logfile.level**)

Organising data / data transfer policy:

- Root directory for all banks (**data.dir**): this is /db in the examples given.
- Access rights to files in production (**production.directory.chmod**)
- Number of release versions to keep in production (**keep.old.version**)
- Method of getting local files by symbolic link or copy (**do.link.copy**)

Executables path:

- Tar executable (**tar.bin**)
- gunzip path (**gunzip.bin**)
- bunzip path (**bunzip.bin**)

Release format:

- the release format if a data gathering policy is the date of the most recent file (**release.dateformat**) (see http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html)

Warning mail:

- smtp server for sending mail (**mail.smtp.host**)
- administration mail (**mail.admin**)
- mail from an administrator (**mail.from**)

Default properties of ftp/ http protocols:

- Number of files to download concurrently *(***file.num.thread***)*

Default properties of the ftp protocol:

- Default port (**port**)

[2] An example of global.properties file is presented as an annex.

- Username (**username**)
- Password (**password**)
- Define a timeout (**ftp.timeout**)
- Number of automatic reconnections (**ftp.automatic.reconnect**)

Default properties of the http protocol:

- Regular expression for an html link with a directory (**http.parse.dir.line**)
- Regular expression for an html link with a file with attributes (**http.parse.file.line**)
- Group number in brackets matching directory name (**http.group.dir.name**)
- Group number in brackets matching directory date (**http.group.dir.date**)
- Group number in brackets matching file name (**http.group.file.name**)
- Group number in brackets matching file date (**http.group.file.date**)
- Group number in brackets matching file size (**http.group.file.size**)

4.4 Computed bank : bank dependencies

Since version 1.1.2, BioMAJ fully integrates the notion of computed bank, hence, of bank dependencies.

A computed bank is a bank (child bank) that is built from others (parent banks). Such a bank has special parameters to access the content of its parents banks. It also means that updating the child bank triggers the update of the parent banks (dependency).

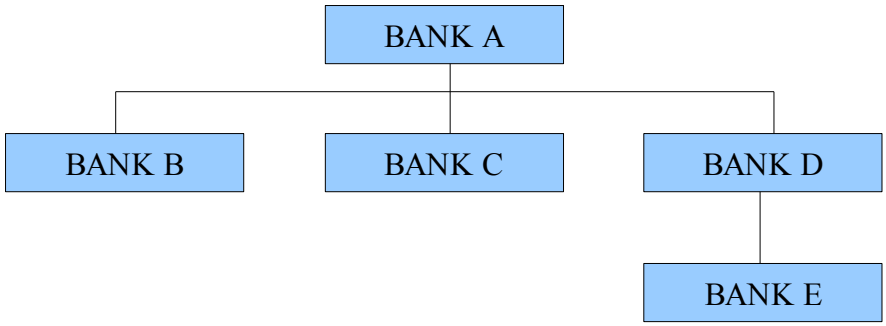


Illustration 16: Bank dependencies

Updating Bank A will trigger the update of Bank B, Bank C, Bank D, which will trigger the update of bank E.

A computed takes as its source its parent banks and only its parent banks. It means that if you declare your bank to be computed, you cannot download files from an FTP server for example, you will need an intermediate bank to do that.

Here is how you declare a computed bank :

```
db.source=b,c,d
b.files.move=**/*
c.files.move=flat/file1 flat/file2
ref.release=d
```

The only mandatory property is `db.source`. Its presence labels the bank as « computed », and all non related parameters will be ignored (server, remote.dir, remote.files).

The properties `<bank name>.files` specify what files of the given bank should be copied in the target bank production directory. The path is relative to the production directory current release root. To copy all the files under `<production_directory>/current/flat`, you need to write `flat/.*`, not just `.*`.

The target bank production directory will look like :

```
<production directory>
      current
            flat
                  b
                        flat
                              file1
                              file2
                  c
                        ...
```

The property ref.release specifies which bank release number should be taken as the target bank release. If this value is not specified, the release is the current date.

In addition, during the post-processing phase, environment variables that contain the current production directory of each parent bank are generated. Their name has the following form: <parent bank name>source, for example « pdbsource » if the parent bank name is « pdb ».

5 Developing and integrating post or pre-processes

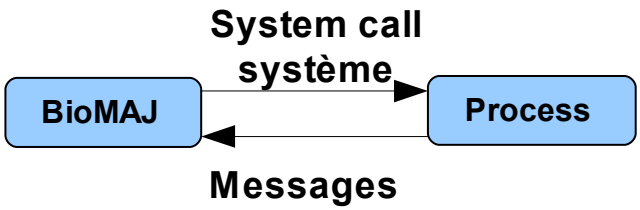


Illustration 17: BioMAJ process communication

BioMAJ can run user scripts/binaries. The engine interacts with scripts by defining environment variables. To be usable, executables must be placed in the directory $BIOMAJ_ROOT/conf/process.

As we have mentioned, during post- and pre-processing stages, BioMAJ launches a system call to programs and monitors their function. By default, BioMAJ simply gets the return value at the end of the program run. If the value is null, the engine thinks everything went well and runs the next process. If the return value is not null, the process receives an error status and the update cycle is stopped. The processes already run concurrently will be stopped at the next sync point (end of block or meta-process).

It is however possible to go further and increase interaction between the engine and processes. To do this, you need to take into account the following specifications in developing the program code.

Integration includes three parts:

- Messaging
- Temp file management
- Using BioMAJ session context

These three parts will allow you to develop processes that fully use the capacities of the BioMAJ engine.

5.1.1 Communication management: messaging

For communication with the administrator, BioMAJ uses three types of message: standard information, warning messages and error messages.

- Standard information is informative messages.
- Warning messages draw attention to a specific event. They are not for serious errors requiring the session to halt but they need to be brought to the attention of the administrator so they can make a decision if necessary (after the end of the session).
- Error messages are created when the application detects a fatal error stopping operation. This is often the last action carried out before the session stops.

Environment variables are used to tag messages sent and class them in one of the three levels: INFO,WARNING and ERROR.

To send a messages on the INFO and ERROR channels, you just need to use classic Unix commands or those in your development language to write on the standard output for the INFO channel and the standard error output for ERROR.

To write a warning message, on the WARNING channel, you just need to tag your warning message adding the environment variable PP_WARNING upstream.

Example in perl:

Info → print STDOUT msg
Warning → print STDOUT ENV{PP_WARNING}msg
Error → print STDERR msg
PP_WARNING to message and display it on the standard output.

Example in csh:

echo "INFO exit »

echo $PP_WARNING"This is a warning"

echo "error output" 1>&2

 **BioMAJ logs this information (by writing on the file system). Intensive use of messages can mean a slowdown when using the application's verbal mode.**

5.1.2 Managing dependencies of produced files

By default, BioMAJ draws up a list of files from versions it is monitoring. It uses this information to optimise re-running of a workflow. It can use a file dependency mechanism attached to a process. Each process must then declare the files that it generates. In the case of a re-run, BioMAJ re-runs the processes if the declared files have not been found.

Files produced during processing can be placed in three categories: declared files, temporary files and unknown status files.

-Unknown status files are in the directory structure. BioMAJ does not recognise them and they are not declared.

-Temporary files are deleted at the end of a session. These are volatile files that disappear at the end of a cycle.

-Declared files are usually results files that match the process results.

The programme can, via declaration, delegate actions to the engine such as file integrity checks or temporary file deletion. To carry out this delegation, the program must declare the files it is using to the engine as well as their type (volatile = temporary and result). Dependency signifies that the generated file will exist in the production directory after the bank has been placed online. A volatile dependency signifies that the file will be deleted once the workflow has been run. The latter is also used for managing temporary files used by several processes (e.g. file concatenation and results indexing).

Declaration uses the same principles as those used for messages. This is a message with a tag written on the standard output of the program that is run.

Messages have the following structure:

Declaration of dependencies:

echo $PP_DEPENDENCY$AbsoluteFilepath

echo $PP_VOLATILE_DEPENDENCIES$AbsoluteFilepath

In PERL, you just need to create a function that declares file dependencies.

Final results file → print STDOUT ENV{PP_DEPENDANCY}file

Volatile (temporary) file → print STDOUT ENV{PP_VOLATILE_DEPENDANCY}file

 **BioMAJ deletes the declared volatile files and it is not recommended to delete them manually.**

5.1.3 Context to script: information on the version number

While running each process, BioMAJ makes several environment variables available for the process being run. These help transit a part of the update cycle context and give dynamic information relating to the new bank version. The list of variables follows:

- **dbname**: bank name
- **datadir**: root directory for all production directories
- **offlinedir**: temporary directory
- **dirversion**: production directory for the bank containing all versions already downloaded and the future version.
- **remotedir**: remote server directory
- **noextract**: Boolean telling whether the downloaded files are extracted
- **localfiles**: downloaded files that will be available during production
- **remotefiles**: regular expression for downloaded files
- **mailadmin**: administration mail
- **mailsmtp**: smtp server for sending mail
- **remoterelease**: version number (available only for post-processes)
- **removedrelease**: removed version number (available only for remove processes)
- **PATH_PROCESS_BIOMAJ**: post processes root directory.
- **PATH_LOG_BIOMAJ**: log directory path defined in file general.conf.
- **PATH_WORKFLOW_BIOMAJ**: workflow files (.properties) directory.
- **RELEASE_ALL_COMPRESSED_FILES_LIST**: Downloaded release files list (a sort of distant ls).
- **RELEASE_ALL_UNCOMPRESSED_FILES_LIST**: List of generated files in flat directory of the current version.
- **RELEASE_OLD_FILES_LIST**: List of files copied from previous release (part of **RELEASE_ALL_UNCOMPRESSED_FILES_LIST**).
- **RELEASE_NEW_FILES_LIST**: List of newly downloaded files (the other part of **RELEASE_ALL_UNCOMPRESSED_FILES_LIST**).

RELEASE_ALL_UNCOMPRESSED_FILES_LIST = RELEASE_OLD_FILES_LIST + RELEASE_NEW_FILES_LIST

To obtain the last four environment varariable, you have to set the property *list.files.available* on true

A few conventions and uses of these variables:

- Downloaded data can be found in `$datadir/$dirversion/future_release/flat`
- To use the version in production: `$datadir/$dirversion/current_release/flat`
- When indexing, create a directory under `$datadir/$dirversion/future_release/` containing the index name (e.g. `$datadir/$dirversion/future_release/blast`) and generate files in this directory.

5.1.4 Return code

In the event of an error being detected, the script should return a non-null code to stop the workflow.

5.1.5 Debugging

A new properties file can be created stage by stage using the following command:

- `BioMAJ .sh –d <dbname> --stage <preprocess|sync|postprocess|deployment>`

The --stage option refers to the session stop point.

Unlike the standard update command, the cycle will be stopped at the stage specified.

5.2 Virtual bank concept

This is an alias containing several independent banks.

e.g. genbank = genbank + genbank_new

- `biomaj.sh –d genbank,genbank_new`

Or create a property file for your virtual bank, and add the following line:

virtual.list=genbank, genbank_new

Two sessions are opened independently of one another ;

The list of banks can be more than two. To avoid network saturation, the number of sessions open is limited by the property **bank.num.thread.**

6 Metadata and classification of sources and processes

6.1 Definition

•Metadata:

According to Wikipedia, **metadata** (from the grec meta "after" and latin data "informations") is donnée that defines or describes other data.

•Classification diagram:

A **classification diagram** is a description of an organisation or division of objects into groups based on common characteristics of objects.

6.2 BioMAJ implementation

BioMAJ is intended to manage sources and monitor their maintenance. Properties have metadata functions to classify sources and processes. They are variables defined in the properties file during declaration of the source and each process.

➢Source

Sources can be placed in a classification structure.

One part of the structure is defined by its BioMAJ name, the dbname, its description, db.fullname and data format, db.formats.

Classes are defined in the variable db.type.

e.g.
```
db.fullname="Genome Gallus gallus (NCBI)"
db.name=Gallus_gallus
db.type=genome
db.formats=gb,fasta,gff
```

➢Post-processing

For information, the property **db.type** can contain classification of the type:

`type1/type2/type3:` i.e. « type2 » is a sub-type of « type1 » and « type3 » is a sub-type of « type2 ».

Classification examples:

```
db.type=genome/eucaryote
db.type=genome/bacteria
db.type=nucleic
db.type=proteic
```

Post-processing also has a similar classification system. Each post-process has a classification

and description field: e.g.

```
process.name=Ncbi_blast_index
process.desc=bank format for ncbi blast
process.type=blast
```

Each time it is run, session production takes on these attributes. They are stored in the status file for the bank and in an index file (generated by the option --index) that contains an overview of bank versions contained in the local warehouse.

6.3 Potential uses

BioMAJ metadata can be used on several levels to help the user or applications to better retrieve the contents of the local warehouse. For example, when generating reports to classify sources.

Properties files sent with BioMAJ are arranged in five groups: *nucleic, proteic, nucleic_proteic, genome,* and *other*.

These groups can be found in the web report. More complex classifications are possible, especially for genomes. It should be possible to attach each genome to its taxonomic classification node and thus improve classification information.

Another application involves making more intensive use of data. Work still has to be done on this but it could be advantageous to make the use of formats and types of post-processes to connect applications and data sources. You could then use the index file to automate configuration of some applications or simply let the user interrogate the warehouse with data format.

7 F.A.Q

1) My workflow is not working any longer

Check the status of the workflow (`biomaj.sh -S dbname`) then analyse the logs (repertoire `$BIOMAJ_ROOT/log/[dbname]/[date]`).

2) The session is still updating

Check that BioMAJ is still working: `ps -aux | grep biomaj`

If the application has stopped, see the previous question.

3) How do I automate BioMAJ operation?

Just place a call to biomaj in your crontab (cf. manual)

4) How can I easily monitor application operations?

`biomaj.sh -S Mybank` et `make_biomaj_report.sh all` .
See results in the directory `$BIOMAJ_ROOT/rapport`.

5) How can I use a properties file?

Create your file in `$BIOMAJ_ROOT/conf/db_properties`.
Run the workflow step by step using the command:
`biomaj.sh -d <dbname> --stage sync`
to validate each stage. Verbose mode can be useful also.

6) How can I develop a new post-process?

Use the language of your choice and in the script source get the environment variables for the BioMAJ session to get all session context information. Create a results directory and launch the system call, taking care to place the results in this directory. For more information, see Manual chapter 5.

7) How does BioMAJ monitor post-processes?

`biomaj.sh --status <dbname>` ; then create a –f tail on the log file associated with the post-process.

8) How does version number detection work?

If nothing is defined for getting version numbers, the version number will be the date of the most recent file on the remote server (this file must be expressed in the property **remote.files**)

This value can be set with the property **release.dateformat** and takes as its default value: *yyyy-MM-dd*

9) **My log files are too large. What shall I do?**

`biomaj.sh --clean-statefile <dbname>` to clean up unnecessary information.
Delete the directory `$BIOMAJ_ROOT/log/[dbname]`

10) **How can I generate a bank from data that is already there?**

Use the calculated bank notion. (cf. Manual 5.2.2)

11) How do I know which banks are updating?

```
biomaj.sh -S --updating
```

12) How do I find out a bank's update frequency?

If the crontab call is regular. See bank statistics to get an indication of the frequency. See the average update frequency value in the html report.

13) Where can I get support?

User mailing list: BioMAJ-users@lists.gforge.inria.fr, in the FAQ on the website biomaj.gforge.inria.fr

14) How can I optimise downloading (via an ftp or http protocol) for a bank?

Change the parameter: **files.num.threads**. By default, three files are downloaded simultaneously.

15) BioMAJ is slow. How can I speed it up?

Initialise the property **log.files** to false if a downloaded file matches an extracted file (this is not the case for an archive tar.gz). Clean up the status file:

`biomaj.sh --clean-statefile <dbname>` , do not use the console and check the property **historic.logfile.level** in the file global.properties.

16) How can I restart indexing for a bank?

If the post-process manages the file's dependencies, just delete the resulting files and restart `biomaj.sh -d <dbname>` . Otherwise, rebuild the bank with the command `biomaj.sh –rebuild <dbname>`.

17) How can I test my post-process?

`biomaj.sh -d <dbname> --stage postprocess` biomaj will run an update cycle for the data and will stop the session after post-processing. If the data is already there, it will run post-processing. You will receive an e-mail containing a report with any errors.

18) How do I configure blast to make it compatible with BioMAJ ?

Just create a blast bank repository directory. Then run the variable with the absolute path of the directory `export BLASTDB = MYREP/BLAST/` copy the variable in the file `$BIOMAJ_ROOT/bin/env.sh`, then personalise the executables access path - formatdb, fastacmd, blastall – in the file `$BIOMAJ_ROOT/conf/process/unix_command_system.cfg`

19) How do I configure SRS to make it compatible with BioMAJ ?

Change the file srsdb.i in the application, run the following global variables ahead of the file:

dataRoot:'/bank' ==> put the value of data.dir, onData:'current/flat/', onIndex:'current/srs/', offData:'futur_release/flat/', offIndex:'futur_release/srs/'. For each indexed bank: write the value of the path dir.version for the bank after the path $dataroot (the repository root) for the offDir , indexDir and offIndexDir. dir:"($dataRoot)*/ebi/uniprot/*($onData)", offDir:"($dataRoot)*/ebi/uniprot/*($offData)", indexDir:"($dataRoot)*/ebi/uniprot/*($onIndex)", offIndexDir:"($dataRoot)*/ebi/uniprot/*($offIndex)"

define the following variables in the file:

$BIOMAJ_ROOT/conf/process/unix_command_system.cfg, EXECUTE_BATCH_OPTION_PBS_SRS=-q srsq -W block=true -A SRS -j oe -V, GETZ=/data/srs/srs/bin/linux73/getz, SRSCHECK=/data/srs/srs/etc/srscheck. Then initialise the SRS environment in the BioMAJ environment file: ($BIOMAJ_ROOT/bin/env.sh)source /MYPATH/OF/srs/etc/prep_srs.sh

8 Appendix

8.1 Example of configuration files

8.1.1 Global.properties

```
# File: global.properties

#----------------
# Mail Configuration
#---------------
#Uncomment thes lines if you want receive mail when the workflow is finished

#mail.smtp.host=
#mail.admin=
#mail.from=

#---------------------
#Proxy authentification
#---------------------
#proxyHost=
#proxyPort=
#proxyUser=
#proxyPassword=

#---------------------
# PROTOCOL
#-------------------
#possible values : ftp, http, rsync, local
protocol=ftp
port=21
username=anonymous
password=anonymous@nowhere.com


#The root directory where all databases are stored.
#If your data is not stored under one directory hirearchy
#you can override this value in the database properties file.
data.dir=/local/db2
production.directory.chmod=775
bank.num.threads=1

#
# Global system properties file

#Programs
tar.bin=/bin/tar
gunzip.bin=/bin/gunzip
bunzip.bin=/usr/bin/bunzip2

#Number of threads to use for downloading and processing
files.num.threads=3

#to keep more than one release increase this value
keep.old.version=0
#Link copy property
do.link.copy=true

#look here to specified a new format :
http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html
release.dateformat=yyyy-MM-dd

#The historic log file is generated in log/
#define level information for output : DEBUG, VERBOSE,INFO,WARN,ERR
#to have info about target and task ant use VERBOSE
historic.logfile.level=VERBOSE
historic.logfile.properties=false
```

```
historic.logfile.task=true
historic.logfile.target=true


PRE1=premail

premail.name=sendMail
premail.exe=sendMailTLSE.pl
premail.args=-s '[NCBI Blast - db.name] Start Biomaj session' -m 'local.time'
premail.desc=mail
premail.type=info

http.parse.dir.line=<a[\\s]+href=\"([\\S]+)/\".*alt=\"\\[DIR\\]\">.*([\\d]{2}-[\\w\\d]{2,5}-
[\\d]{4}\\s[\\d]{2}:[\\d]{2})
http.parse.file.line=<a[\\s]+href=\"([\\S]+)\".*([\\d]{2}-[\\w\\d]{2,5}-[\\d]{4}\\s[\\d]{2}:
[\\d]{2})[\\s]+([\\d\\.]+[MKG]{0,1})

http.group.dir.name=1
http.group.dir.date=2
http.group.file.name=1
http.group.file.date=2
http.group.file.size=3

#Needed if data sources are contains in an archive
log.files=true

local.files.excluded=\\.panfs.*

#~40mn
ftp.timeout=2000000
ftp.automatic.reconnect=2
```

8.2 BioMAJ properties

Properties	***Values***	***Description***	***Optional***	***Default value***	***Default location***
proxyHost	<host.proxy>	Proxy adress	yes	optional	global.properties
proxyPort	<port.proxy>	Proxy port	yes	optional	global.properties
proxyUser	<login>	Authentification proxy login	yes	optional	global.properties
proxyPassword	<password>	Authentification proxy password	yes	optional	global.properties
mail.smtp.host	<hostname.Domain.xx>	SMTP server for sending mail	yes	optional	global.properties
mail.admin	<admin@MDomain.xx,**>	Bank administrator mail	No if *mail.smtp.host defined*	optional	global.properties or dbname.properties
Mail.from	<admin@MDomain.xx>	Bank administrator mail	No if *mail.smtp.host defined*	optional	global.properties or dbname.properties
port	number	Indication of port for ftp protocol	Yes	21	global.properties or dbname.properties
username	String	Username for ftp protocol	Yes	anonymous	global.properties

Properties	*Values*	*Description*	*Optional*	*Default value*	*Default location*
password	String	Password for ftp protocol	Yes	- (Your mail!)	global.properties
ftp.timeout	positive number or -1	Timeout in mms for ftp protocol. If –1 no timeout.	Yes	100000	global.properties
ftp.automatic.reconnect	Positive number	Number of automatic reconnections for ftp protocol	Yes	5	global.properties
data.dir	Directory path	Root directory for local banks	no	To be defined	global.properties
production.directory.chmod	String	Future access rights to the production directory	no	755	global.properties
bank.num.threads	Number	Bank number management	no	1	global.properties
tar.bin	Path and executable	Tar binary path	no	/bin/tar	global.properties
gunzip.bin	Path and executable	Gunzip binary path	no	/bin/gunzip	global.properties
bunzip.bin	Path and executable	Bunzip binary path	no	/usr/bin/bunzip2	global.properties
files.num.threads	Number	Number of files downloaded concurrently	no	1	dbname.properties
Log.files	Boolean	Logs information on downloaded files	No	True	global.properties
protocol	ftp,http,rsync,local	Protocol used to get the bank	no	ftp	global.properties
do.link.copy	link,copy	Authorises symbolic links with production files	Yes	true	global.properties
keep.old.version	Number	Number of saved complete releases	Yes	0	global.properties
frequency.update	Number	Number of days when the bank does not need updating	Yes	0	global.properties
release.dateformat	Expression DateFormat	Release format http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html	no	yyyy-MM-dd	global.properties
historic.logfile.level	DEBUG,VERBOSE,INFO,WARN,ERR	Define at display level logs and console	Yes	DEBUG	global.properties
historic.logfile.properties	true, false	Write ant properties defined by the application in logs	Yes	false	global.properties
historic.logfile.task	true, false	Write the name of workflow tasks in logs	Yes	false	global.properties

Properties	*Values*	*Description*	*Optional*	*Default value*	*Default location*
historic.logfile.target	true, false	Write the name of workflow metatasks in logs	Yes	false	global.properties
db.name	This property must carry the same name as the properties file.	Name of the virtual bank	no	-	dbname.properties
db.fullname	String	Full name	no	-	dbname.properties
db.type	String	Bank type	Yes	-	dbname.properties
server	String	Server address	no	-	dbname.properties
offline.dir.name	Path	Temporary directory for downloads and extractions	Yes	(Optional) datadir/offdir/bname_tmp	dbname.properties
dir.version	Path	Bank version directory	Yes	(Optional) Datadir/dbname/	dbname.properties
remote.dir	Path	Location of the directory where the bank is held on the remote server	no	-	dbname.properties
remote.files	Regular expressions	Filter on the files to be downloaded	no	-	dbname.properties
Remote.excluded.files	Regular expressions	Filter on the files not to download that can match with remote.files	Yes	-	dbname.properties
local.files	Regular expressions	Filter on files to be put into production	no	-	dbname.properties
local.files.excluded	Regular expressions	Filter on files not to be put into production	Yes	-	
release.file	String	Name of file containing the release	Yes	-	dbname.properties
release.regexp	Regular expressions	Regular expression for getting the release	Yes	-	dbname.properties
release.file.compressed	true, false	True if file containing the release is compressed	Yes	false	dbname.properties
no.extract	True,false	If True, deletes the file extraction stage for tar,gz,etc.	Yes	False	dbname.properties
list.files.available	True,false	Set up environments varaibles: `RELEASE_ALL_COMPRESSED_FILES_LIST,RELEASE_ALL_UNCOMPRESSED_FILES_LIST,RELEASE_OLD_FILES_LIST,RELEASE_NEW_FILES_LIST`	yes	False	dbname.properties
visibility.default	private,public	Default access for a bank	yes	private	global.properies