Biomaj Watcher
User Guide

Table of contents

Table of figures

1 Application purpose

Biomaj Watcher (BW) has been designed to provide a quick overview of the banks managed by Biomaj and a simple access to the most common administration tasks.

2 Features

BW allows :

- A guest user to visualize/search/filter the online banks. It includes charts and detailed information about the bank.
- A logged-in user to :
    - Run updates
    - Visualize the status of the banks
    - Schedule updates with a cron-like syntax.
    - View the latest errors with access to the related log file
    - Create/modify the banks properties

3 Setting up

Most of the configuration for BW is done via the installer, however, you can change it by editing the right properties in the file : `[tomcat]/conf/Catalina/[host_name]/BmajWatcher.xml.`

3.1 Authentication

Two authentication methods are available : file authentication and LDAP authentication. You can use both at the same time so it is recommended that you always enable at least file authentication.

3.1.1 File authentication

BW will look in a specific file for the administrator login and password in a specific format .
To enable file authentication :

- set the USELOCAL property in the configuration file to 1.
- save the administrator login and password in a file named '`.bmajwatcherpasswd`' in your home directory as a SHA1 hash with the following format : `loginhash:passwordhash`

3.1.2 LDAP authentication

To enable LDAP authentication, modify the following parameters in the configuration file :

- USELDAP : Set to 1.
- LDAPHOST : LDAP server address
- LDAPDN : LDAP distinguished name
- OPTFILTER : Search filter

3.2 Environment variables

BW needs some contextual properties to be specified to be able to communicate with the biomaj core application. In the same configuration file, modify the following properties :

- BIOMAJ_ROOT : Biomaj core root directory
- BW_ROOT : BiomajWatcher root directory on application server
- JAVA_HOME : Location of the JDK
- ANT_HOME : Location of ant installation directory

3.3 Start up

To run BW start the application server that hosts it. In our case it is Apache Tomcat. It is done by running the `startup.sh` script located in the `bin` directory. The application can be reached at : `http://<server>:8080/BmajWatcher/`

4 User interface

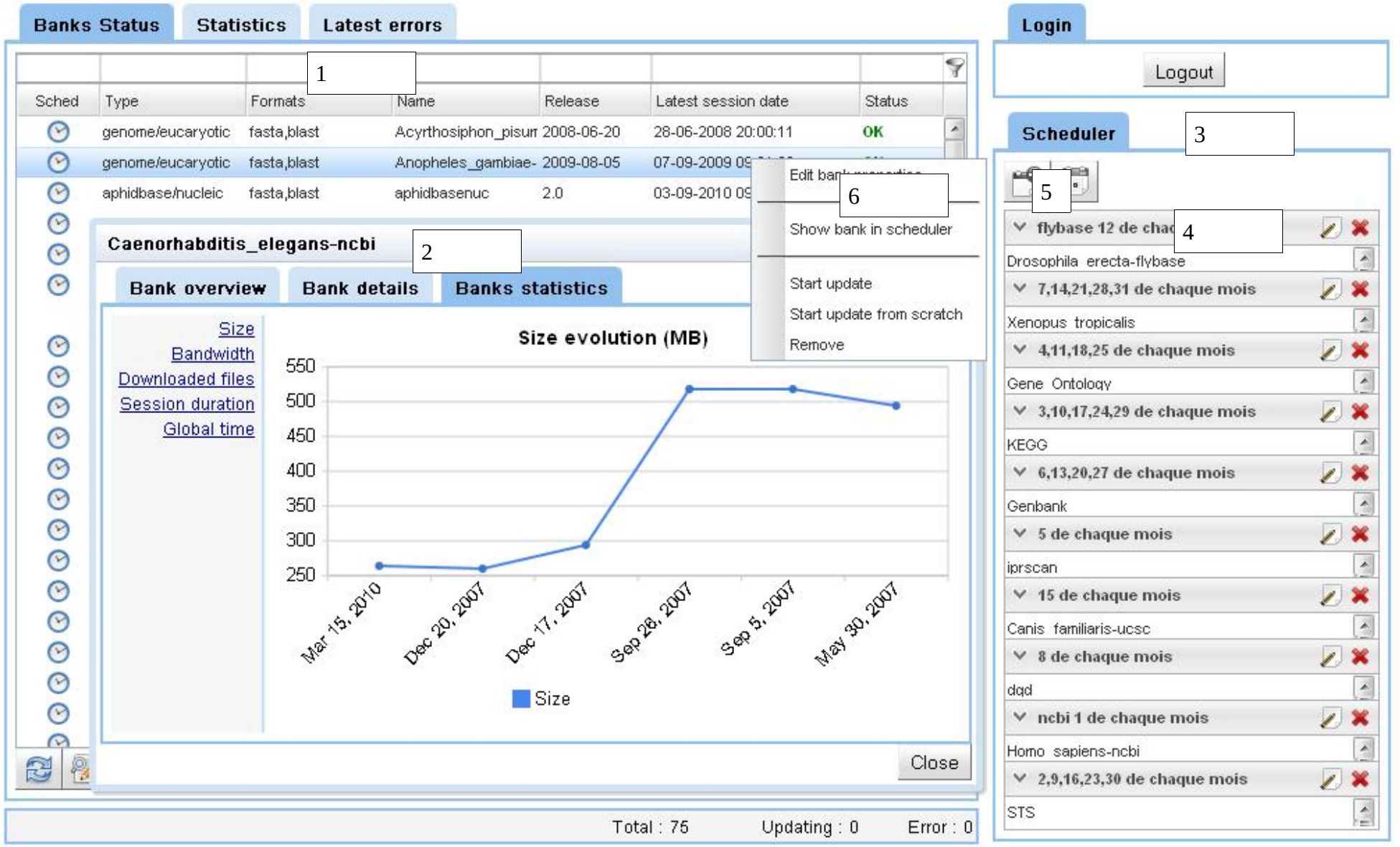

Figure 1: General administration interface

1 : Grid that contains the banks
2 : Detailed information on a bank when double clicking on a bank
3 : Scheduler where the jobs are displayed
4 : A specific job. To delete it, click on the red cross, to edit it click on the other icon. To remove a bank from a job, right clic the bank then click on 'remove'. Banks are added to a job by drag&dropping them from the main grid.
5 : The first button is for creating a new job. You will have to specify a name and a time via a cron like syntax (see chapter 6).
The second button is to display a calendar view of the jobs.
6 : The contextual menu from where you can run an update, edit the bank properties...

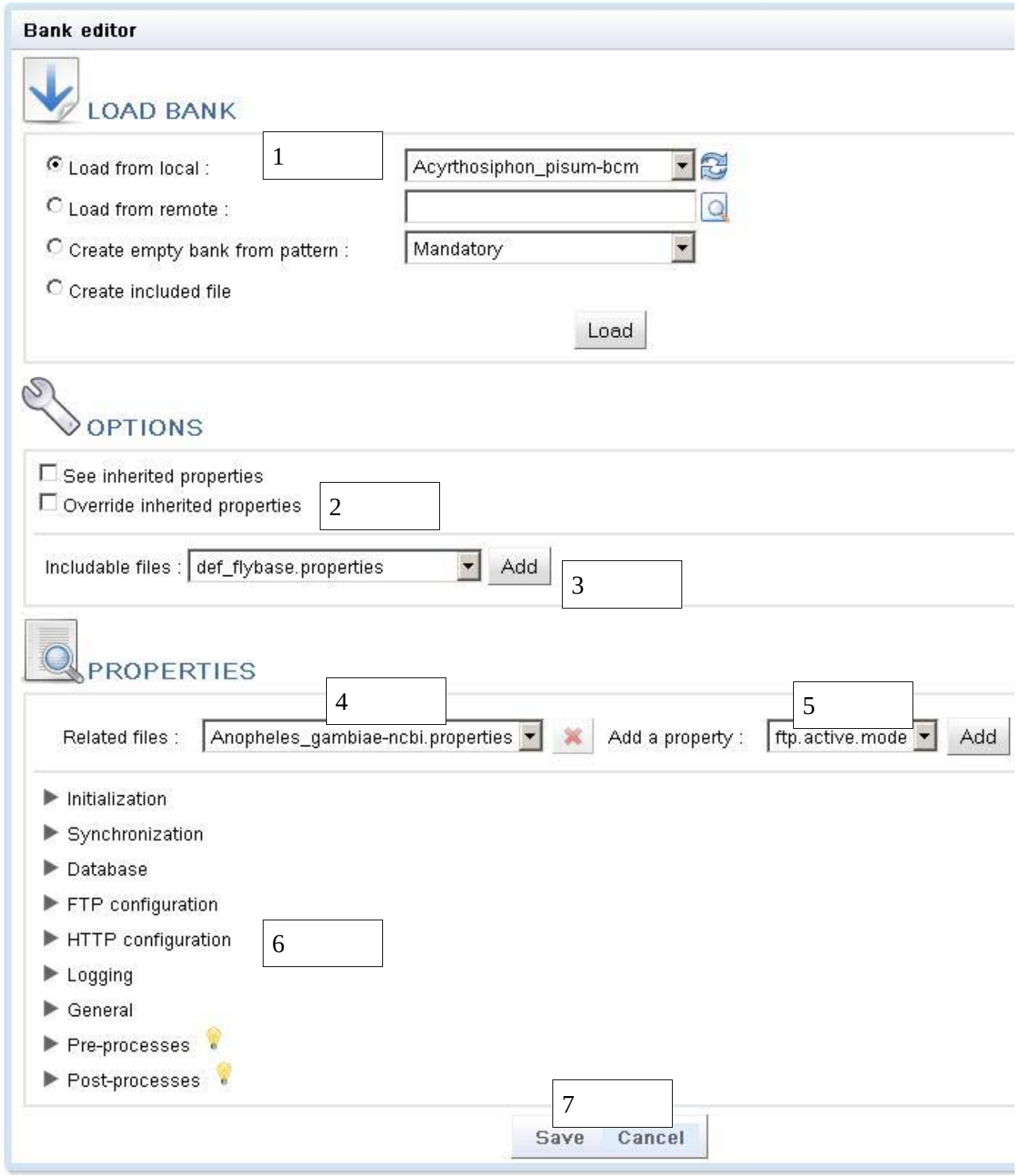

Figure 2: Bank edition interface

1 : In this section you can load/create a bank's properties in various ways :

- Load from local : you can load one of the bank defined in the biomaj workflow directory
- Load from remote : enter an url and BW will parse the web page to find any .properties files and display them.
- Create empty bank from : this option allows you to create a bank from scratch with a defined set of properties.
- Create includeded file : this option is for creating properties files that can be included in any bank. The created files are saved in [biomaj workflow directory]/include.

2 : A bank's properties is composed of the properties contained in the bank properties file and the properties inherited from the 'global.properties' file and any other included file specified by the property 'include.properties' in the bank file. BW let you choose whether to display and override those inherited properties.

3 : The files located in [biomaj workflow directory]/include are listed and can be included to the currently loaded bank.

4 : For a bank, list all the associated files : The bank file, the global.properties file and any other files that have been included.

5 : You can add a property to the bank if it has not been defined yet.

6 : This section contains the bank's properties organized by categories.

7 : The 'save' button allows you to :

- override the loaded bank
- save the bank under a new name
- save the bank and run an update.

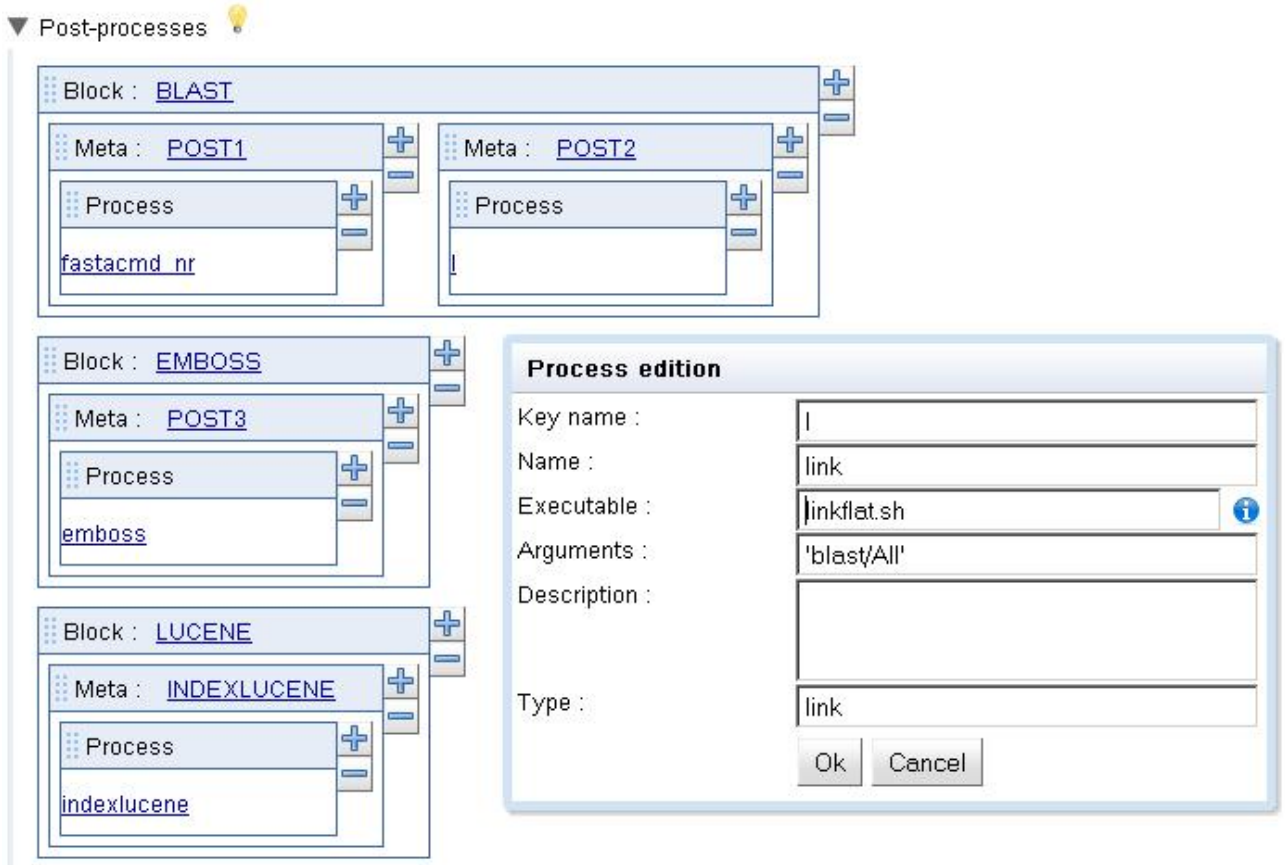

Figure 3: Post-process section detailed view

1 : The processes are displayed in boxes with the following convention : from top to bottom is sequential (blocks and processes), for left to right is concurrent (meta-processes). The boxes can be :

- reordered via drag&drop
- renamed : click on the link the change the name of a meta or a block
- Deleted : the '-' button removes a box
- Added : the '+' adds a box of the same kind

2 : Clicking on the name of a process opens a dialog box that allows the user to edit the process properties.

3 : The 'executable' field is a suggest box. It proposes as suggestions any of the process listed in the biomaj process directory. Selecting a process that way will auto fill the name and description fields (if a description was provided). The descriptions have to be entered manually in the file 'processDescription.properties' located in you webapp directory. It has the following structure :

process_name_1=process description.
process_name_2=process descrtiption
...

5 URL based bank querying

BW provides a REST service that allows the user to retrieve information on the banks in JSON

format via the URL. You can get 3 types of result :

5.1 The formats list

Syntax : **<BW URL>/GET?formats**

```
{
   "format":[
      {
         "value":"blast"
      },
      {
         "value":"emboss"
      },
      {
         "value":"fasta"
      }
   ]
}
```

5.2 The types list

Syntax : **<BW URL>/GET?types**

```
{
   "type":[
      {
         "value":"aphidbase/nucleic"
      },
      {
         "value":"aphidbase/proteic"
      },
      {
         "value":"gene/duplicate"
      }
   ]
}
```

5.3 The banks list

5.3.1 General structure

The structure of the returned JSON object is the following :

```
{
   "banks":[
      {
         "name":"bank name",
         "session_date":"session date",
         "current_release":"release number",
         "releases":{
            "release number":{
               "path":"directory path",
               "formats":[
                  {
                     "value":"format 1",
                     "sections":[
                        {
                           "name":"section name",
```

```
                              "files":[
                                 "filepath_1",...,"filepath_n"
                              ],
                              "sections":[
                                 ...
                              ]
                           },...
                        ]
                     },
                     {
                        "value":"format 2",
                        "sections":[
                           ...
                        ]
                     }
                  ]
               },
               ...
            },
            "db_type":"bank type"
         }...
      ]
   }
```

5.3.2 Section element

In further details, for each format, a list of the associated files is returned as a section element. A section structures the files according to their directory structure.

Example :

Directory structure :

```
fasta
  |- dir_1
  |    |- dir_11
  |    |     |- file_a
  |    |     |- file_b
  |    |- dir_12
  |          |- file_c
  |- dir_2
       |- file_d
       |- dir_21
            |- file_e
```

Sections :

```
"formats":[
   {
      "value":"fasta",
      "sections":[
         {
            "name":"dir_1",
            "sections":[
               {
                  "name":"dir_11",
                  "files":[
                     "file_a","file_b"
                  ]
```

```
            },
            {
               "name":"dir_12",
               "files":[
                  "file_c"
               ]
            }
         ]
      },
      {
         "name":"dir_2",
         "files":[
            "file_d"
         ],
         "sections":[
            {
               "name":"dir_21",
               "files":[
                  "file_e"
               ]
            }
         ]
      }
   ]
}
]
```

If you don't want the the sections to be shown you can use the **lightmode** parameter : only the formats name will be returned.

```
"formats":[
   {
      "value":"fasta"
   },
   {
      "value":"blast"
   }
]
```

5.3.3 Filtering the results

The list can be filtered by type and/or format. You can specify multiple types and formats with the | character.

Examples :

All banks with fasta format :
<BW URL>/GET?banks=all&formats=fasta

All banks with fasta format and nucleic type :
<BW URL>/GET?banks=all&formats=fasta&types=nucleic&ligthmode

All banks with fasta or xml format and nucleic or proteic type :
<BW URL>/GET?banks=all&formats=fasta|xml&types=nucleic|proteic

Genbank bank only if it has fasta format :
<BW URL>/GET?banks=genbank&formats=fasta

6 Scheduling syntax

The syntax used to schedule jobs is very similar to the cron syntax. The main difference is that it can handle seconds while cron smallest time unit is the minute.
http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html