

BioUML workbench

Contents

1. Introduction	3
1.1. Databases	3
1.2. Architecture overview	4
1.3. BioUML server	6
2. Installation	8
3. Configuration	10
3.1. Database module	10
3.2. Plug-in	10
4. Security and access rights	11

1 Introduction

This administrator guide describes how to install and configure BioUML server.

1.1 Databases

Modeling of biological systems requires close integration with experimental data. The distinctive feature of BioUML workbench is tight integration with biological databases.

Databases can be installed locally or can be accessed via Internet from BioUML server. BioUML server supports secure access to databases. Server administrator can configure security settings for access to each database installed on the server. Information where database installed (locally or on the server side) as well as about its availability is displayed using different icons (Figure 3.1).

Icon color indicates where database is installed and its accessibility:

- blue color - database is installed locally, accessible for reading and writing;
- yellow - remote public database, accessibility for reading and writing is specified by R and W letters;
- red color - remote protected database, user should log-in to get access to the database, accessibility for reading and writing is specified by R and W letters;
- green color - remote protected database, user successfully logged in, accessibility for reading and writing is specified by R and W letters;

Table 3.1.

Icons for local and remote databases

Local database



local database, available for reading and writing

Remote database



public remote database, read only



public remote database, available for reading and writing

before log-in



remote public database, requires log-in for writing



remote protected read only database, requires log in for reading



remote protected database, requires log in for reading and writing

after log-in



Figure 3.1. Icons for local and remote databases.

See also:

- [BioUML server](#)

1.2 Architecture overview

BioUML workbench is a plugin-based application framework that provides its extensibility and possibility of seamless integration of other tools for systems biology. It consists from an Eclipse platform (<http://www.eclipse.org>) runtime kernel that supports 'plug-ins' and a set of plug-ins that support database access, diagram editing, and biological systems simulation.

Plug-in based architecture

- Plug-in - is the smallest unit of BioUML workbench function that can be developed and delivered separately into BioUML workbench. Plug-ins are coded in Java. A typical plug-in consists of Java code in a JAR library, some read-only files, and other resources such as images, message catalogs, native code libraries, etc. A plug-in is described in an XML manifest file, called plugin.xml. The parsed contents of plug-in manifest files are made available programmatically through a plug-in registry API provided by Eclipse runtime.
- Extension points are well-defined function points in the system where other plug-ins can contribute functionality.
- Extension is a specific contribution to an extension point. Plug-ins can define their own extension points, so that other plug-ins can integrate tightly with them.

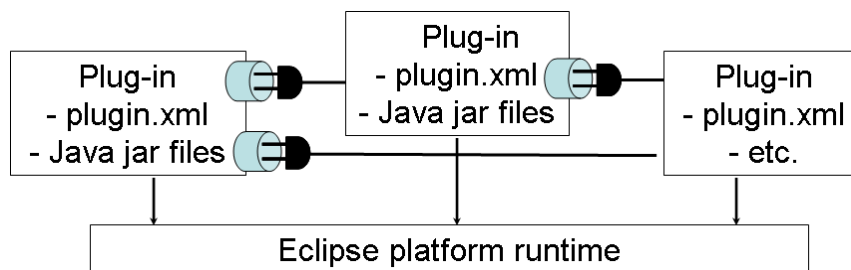




Figure 1.13. Plug-in based architecture,  - extension point;  - extension

Architecture of BioUML workbench

BioUML workbench installation includes a plugins folder where individual plug-ins are deployed. Each plug-in is installed in its own folder under the plugins folder. A plug-in is described in an XML manifest file, called plugin.xml, residing in the plug-in's folder. The parsed contents of plug-in manifest files are made available programmatically through a plug-in registry API provided by Eclipse runtime.

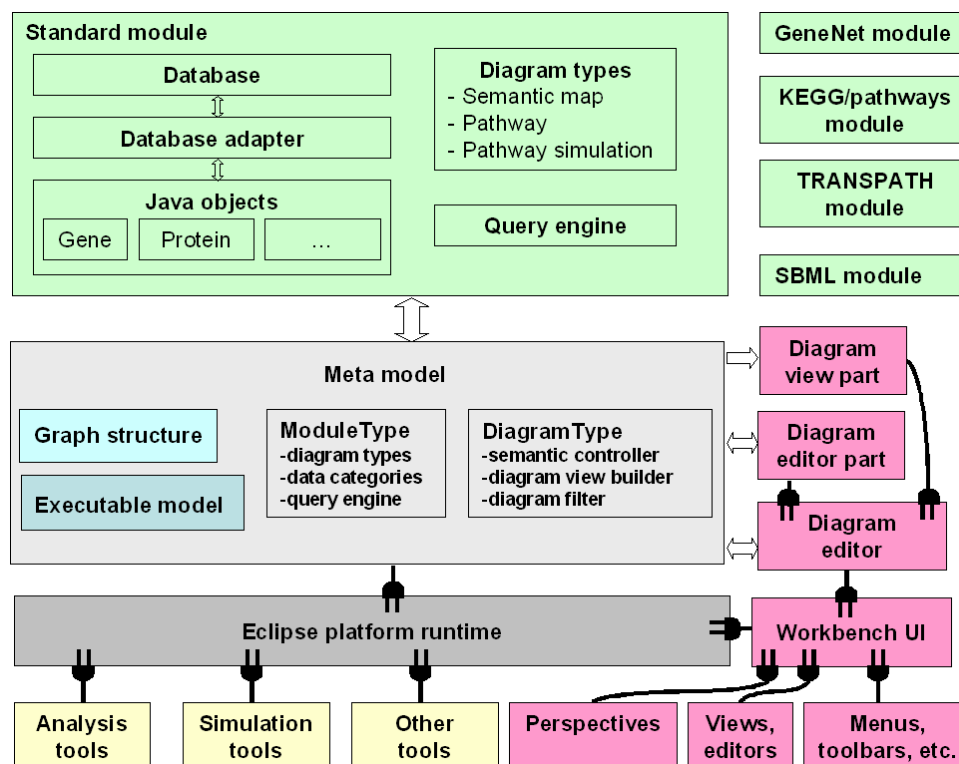


Figure 1.14. BioUML workbench - architecture overview.

Formal description and modeling of biological systems require coordinated efforts of different group of researchers:

- " programmers - they should provide computer tools for this task;
- " problem domain experts - they should specify what and how should be described;
- " experimenters and annotators - they should describe corresponding data following to these rules;
- " mathematicians - they should provide methods for models analysis and simulations.

BioUML architecture separates these tasks so they can be effectively solved by corresponding group of researchers and provides simple contract how these groups and corresponding software parts should communicate.

Architecture of BioUML server

BioUML server is Java application that is started as servlet on J2EE compatible server (we are using Tomcat server).

Like BioUML workbench it is also uses Eclipse runtime to manage by plug-ins that provide different services (Figure 1.15). Main services provided by BioUML server are:

- database service - provides information about database and secure access to it
- access service - provides access to databases (read/write)
- diagram service - provides protocol to read/write diagram and all diagram elements during one HTTP request
- Lucene service - provides full text search and its configuration
- query service - provides indexed search for a database

BioUML server supports access to different types of databases. Main of them are:

- relational databases (for example, Ensembl database that is available as MySQL dump)

- text databases (for example KEGG/Ligand database)
- XML databases (for example databases in BioPAX or SBML formats)
- databases available via web services (for example SABIO-RK database)

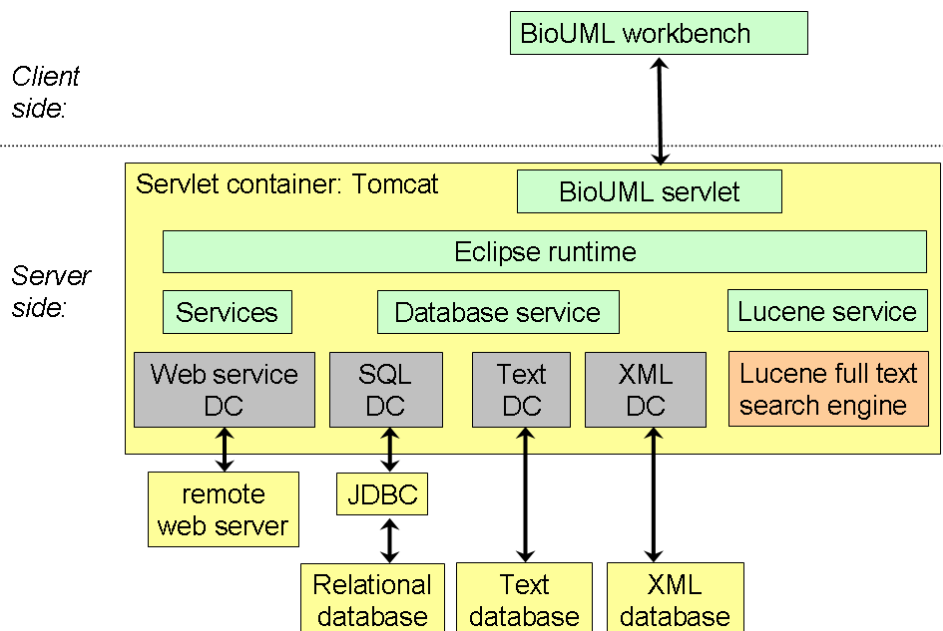


Figure 1.15. Architecture of BioUML server.

1.3 BioUML server

BioUML server provides access to databases installed on the server side for BioUML workbench via the Internet.

When BioUML workbench is started first time Setup wizard (step 3) helps a user to configure connection of BioUML workbench with BioUML server. Dialog Load databases provides the same functionality.

Security support

BioUML server supports secure access to databases. Server administrator can configure security settings for access to each database installed on the server.

User interface

From a user view point databases installed on the server side (remote databases) look similarly with databases installed locally and a user can use remote databases by the same way as local databases:

- a remote database content can be shown in a repository tree;
- user can add, edit and remove records from remote database;
- user can open an edit diagrams from remote database;
- all search engines (data search, full text search, graph search) are working with remote databases via special protocol;
- simulation engine can store simulation results and plots in a remote database.

Figure 1.11 demonstrates user interface provided by BioUML workbench for KEGG/Ligand database:

- repository pane (top left) - shows KEGG/Ligand database structure. Yellow icon indicates that KEGG database installed on the server and is publicly available, letter 'R' on the icon indicates that database is read only;
- diagram pane (top, right) - shows the diagram for Glycolysis/Gluconeogenesis metabolic pathway;
- object view (bottom left) - shows description of a selected node on the diagram or repository tree;
- tab for full text search (bottom right) - shows result of full text search in Compound table with query 'acid'.

The screenshot displays the BioUML workbench interface. The top-left pane shows a tree view of databases, with KEGG selected. The top-right pane shows a metabolic pathway diagram for Glycolysis / Gluconeogenesis. The bottom-left pane shows the 'Node' description for D-Glucose. The bottom-right pane shows a search results table for the query 'acid'.

Node Description:

Title: -D-Glucose
Size: Dimension (width = 16, height = 16)
Data:
ID: C00267
TI: -D-Glucose
NM: alpha-D-Glucose
FM: C6H12O6

Search Results Table:

	Full name	Name	title	completeName	synonyms	Score
0	databases/KEGG/Data/i C03794	Adenylosuccinate	N6-(1,2-Dicarboxyethyl)	Adenylosuccinic acid		1.0
1	databases/KEGG/Data/i C03782	D-(1-Aminoethyl)phosp	D-(1-Aminoethyl)phosp			1.0
2	databases/KEGG/Data/i C04871	propionic acid	(RS)-2-[4-(3-Chloro-5-			1.0
3	databases/KEGG/Data/i C04849	acid	(5Z,9E,14Z)-(8xi,11R,1	(5Z,9E,14Z)-(8xi,11R,1		1.0
4	databases/KEGG/Data/i C04843	(5Z,9E,14Z)-(8xi,11xi,1	(5Z,9E,14Z)-(8xi,11xi,1	acid, Trioxilin A3		1.0
5	databases/KEGG/Data/i C04834	acid	(2E,6E)-(10R,11S)-10,	(2E,6E)-(10R,11S)-10,		1.0
6	databases/KEGG/Data/i C04805	5-Hydroxyeicosatetraen	5(S)-HETE	5-HETE;(6E,8Z,11Z,14		1.0

Figure 1.11. User interface provided by BioUML workbench for working with remote database (here KEGG/Ligand database).

See also:

- Setup wizard (step 3)
- Load databases dialog
- [Architecture overview](#)

2 Installation

Installing Tomcat

BioUML server is running under Apache Tomcat server. We recommend to use 4.1.3 version.

Some libraries are necessary for BioUML server. Copy it to correspond folders:

- TOMCAT_IN_ZIP\common\endorsed
- TOMCAT_IN_ZIP\common\lib
- TOMCAT_IN_ZIP\shared\lib

Note: you can just copy Tomcat from *.zip archive (for example to **C:\jakarta-tomcat-4.1.30** folder). In this case libraries copying is not necessary.

Check environment variable **CATALINA_HOME**. It should target to Tomcat folder (for example, **C:\jakarta-tomcat-4.1.30**). You should create it if it isn't set.

Go to **bin** directory of Tomcat and run server (use **startup.bat**). By default Tomcat configured for 8080 port. Open your Internet browser and type <http://localhost:8080> in address field. You should see Tomcat start page.

Deploying BioUML server

Stop Tomcat server if it running (use **shutdown.bat** in **bin** directory of Tomcat).

To deploy BioUML server you should download and install BDk version of BioUML.

There is special Ant target for server deploying. Go to **build.xml** file and find target "**deploy.server**".

There are two properties defined there:

- **SERVER_PATH** - path for saving BioUML server configuration, plugins and repositories. We recommend to use something like this: **C:/BioUML_Server**
- **TOMCAT_PROJECT_PATH** - path to the project in Tomcat installation. For example, if Tomcat installed in **C:\jakarta-tomcat-4.1.30** then the value should be **C:\jakarta-tomcat-4.1.30\webapps\biouml**. This path is used to deploy *.war file of BioUML server and to set logging properties for Tomcat.

Check this properties one more time and execute "**deploy.server**" ant target.

After successfully execution you should find 4 new folders in **SERVER_PATH**

- **configuration** - configuration of BioUML server plugins
- **plugins** - installed plugins
- **repo** - BioUML server databases repository
- **resources** - BioUML server data repository

Run Tomcat server and go to <http://localhost:8080/biouml>. The message "**Internal BioUML application server**"

indicates that BioUML server successfully deployed.

3 Configuration

This chapter describes how to set up databases and plugins for BioUML server.

3.1 Database module

Database module in BioUML server is the same as database module in client BioUML application.

The simplest way to setup server database is to copy it from **data** folder of client BioUML application to **repo** folder of BioUML server.

To check database availability run BioUML server and try to load database by client BioUML application using address and port of your server.

Read [Security and access rights](#) to set up access rights for server database.

3.2 Plug-in

All plug-ins of BioUML server are saved in the **plugins** folder of SERVER_PATH.

To install new plugin to BioUML server just copy it compiled version from **plugins** folder (**bdk\plugins**) to **SERVER_PATH\plugins**. Check availability of plugin.xml and *.jar files in plugin folder.

Note: to allow BioUML server to refresh plug-ins configuration you should delete all **org.eclipse.*** folders from **SERVER_PATH\configuration** before Tomcat server starting.

4 Security and access rights

BioUML server provides a way to set access rights on databases or data collections for users.

User names and their access rights should be saved in SQL tables. You can create it in any database but we recommend you create new database (for example, **biouml**) for this purpose.

BioUML security system needs two SQL tables:

```
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `user` varchar(30) NOT NULL default '',
  `password` varchar(50) default NULL,
  `email` varchar(50) default NULL,
  `firstname` varchar(50) default NULL,
  `lastname` varchar(50) default NULL,
  `comment` text,
  UNIQUE KEY `IDX_UNIQUE_concepts_ID` (`user`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
DROP TABLE IF EXISTS `permissions`;
CREATE TABLE `permissions` (
  `user` varchar(30) NOT NULL default '',
  `module` varchar(50) default NULL,
  `permission` int(11) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Table **users** contains information about BioUML server users. Required fields are **user** and **password**. Note: in current version passwords saved in not encrypted format.

Table **permissions** contains access rights for users.

- **user** field value must be one of user described in **users** table. To set permission for unauthorized users use "<guest>" value for **user** field.
- **module** field must contain path to protected collection. For example "**databases/test**".
- **permission** field shows the access level. There are 5 different roles for user:
 - INFO = 1 - access to DataCollection **getSize**, **getDataElementType**, **isMutable**, **contains**, **getInfo**, **getDisplayName**, **getDescription** methods.
 - READ = 2 - access to DataCollection **get**, **iterator**, **getNameList** methods.
 - WRITE = 4 - access to DataCollection **put** method.
 - DELETE = 8 - access to DataCollection **remove** method.
 - ADMIN = 16 - access to DataCollection **close**, **release**, **addDataCollectionListener**, **removeDataCollectionListener**, **setDisplayName**, **setDescription**, **setVisible** methods.

You can combine different roles by summing their values. For example, INFO+READ=1+2=3, full access=1+2

+4+8+16=31

Security properties

Create **security.properties** file in **repo** folder of BioUML server. This file should contains parameters for **biouml** database connection (database with **users** and **permissions** tables)

Example:

```
jdbcDriverClass=org.gjt.mm.mysql.Driver
jdbcURL=jdbc:mysql://lachesis.developmentontheedge.com:3306/biouml
jdbcUser=biouml
jdbcPassword=biouml
```

- **jdbcDriverClass** - name of JDBC driver class
- **jdbcURL** - database connection URL
- **jdbcUser** - database user name
- **jdbcPassword** - database password

Protect database

To protect database you should wrap it into **biouml.model.ProtectedModule**. Rename **default.config** file of unprotected database (for example, to **default.primary.config**). Create new **default.config** file with the next properties:

```
name=DATABASE_NAME
class=biouml.model.ProtectedModule
nextConfig=default.primary.config
protectionStatus=2
showStatistics=true
version=0.8.6
update=12.05.2009
```

- **name** - the name of database.
- **class** - database class (should be **biouml.model.ProtectedModule**)
- **nextConfig** - the name of primary config file (in our example **default.primary.config**)
- **protectionStatus** - protection status visible for client application (type of database icon, read [Databases](#) for more information)

Possible values:

- **0** - not protected database
- **1** - public database, read only
- **2** - protected database, requires log-in for writing

- **3** - protected read only database, requires log-in for reading
- **4** - protected database, requires log-in for reading and writing
- **showStatistics** - indicates where module statistics can be show to users
- **version** - database version
- **update** - update data

Protect data collection

In BioUML you can protect each data collections of database. The way is the same as for databases: you should rename coll

```
name=COLLECTION_NAME
class=ru.biosoft.access.security.ProtectedDataCollection
nextConfig=PRIMARY_CONFIG_FILE
```

- **name** - collection name
- **class** - collection class (should be ru.biosoft.access.security.ProtectedDataCollection)
- **nextConfig** - neme of renamed data collection config file