



Generic Database Catalog User Guide

TRIL Centre's BioMOBIUS™ Research Platform: an Open, Shareable Software
and Hardware System

Disclaimer

The BioMOBIUS database components are supplied as is with no warranty implied or otherwise. The standard BioMOBIUS™ licensing terms and conditions apply.

Audience

Platform development engineers, biomedical engineers

Purpose

Describes the use of the BioMOBIUS™ database related components

Pre-requisites

BioMOBIUS v2.0 installed.

Version: 1.6

1	Overview	3
2	Installation	4
2.1	SQLite Install.....	6
3	Design.....	7
3.1	Usage Model	8
3.2	BioMOBIUS Database Blocks	10
3.3	EyesWeb Datatypes	11
4	Configuration	12
4.1	DB_DML.....	13
4.2	DB_Query	16
5	Sample Patch and Database	19
6	Troubleshooting.....	21
6.1	Common Errors	23
6.2	Supported Database Systems	24
6.3	Limitations	25

WARNING: It is assumed that the user has an adequate knowledge of DB design and SQL before attempting to use the BioMOBIUS DB blocks. The purpose of this document is to describe the usage and limitations of the DB blocks, the user should refer elsewhere for general DB/SQL queries and for the specific RDBMS that the user employs. The user is also recommended to read this entire document before running the test patches or incorporating the DB blocks in their own patches. The user is also encouraged to take note of the guidelines and warnings labeled within this document.

1 Overview

This document describes how BioMOBIUS supports data persistence using standard off the shelf database products. The standard database products supported with a direct connection are SQLite3 and MySQL, and in theory any database system may be connected with the appropriate Open Database Connectivity (ODBC) driver.

The BioMOBIUS database support components are supplied within the BioMOBIUS release package. The components are comprised of the BioMOBIUSGenericDatabase catalog, sample EyesWeb patches, SQL script files and this document. The catalog contains two main blocks – Database Data Manipulation Language (DML) and Database Query. These two blocks enable the user to incorporate simple database functionality into an EyesWeb patch without having to develop any database related code. Database support can be added to a patch in just a few moments – the user drags a database block onto a patch and connects the input/output pins that require database storage or retrieval. Figure 1 illustrates the use of DML blocks to store X10 sensor data.

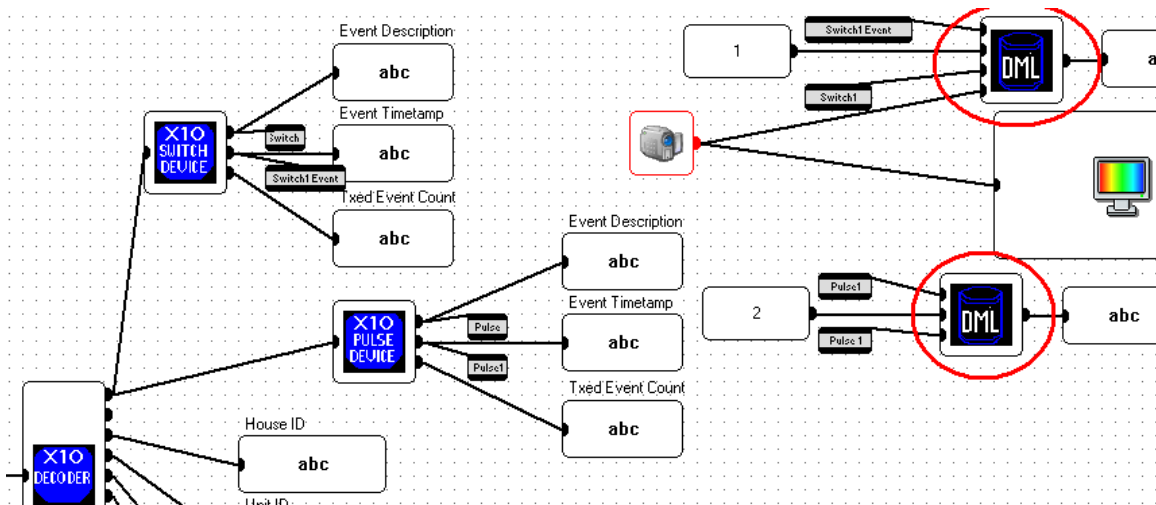


Figure 1 Typical patch containing DML blocks

The following sections describe the installation, design, configuration and typical use of these blocks. A further section describes a sample patch and database and a final section details some troubleshooting steps.

2 Installation

The BioMOBIUS generic database blocks are automatically installed as part of the standard BioMOBIUS v2.0 release and subsequent releases. The sample patches and support files are also installed as part of BioMOBIUS. Future updates to the BioMOBIUS DB support components will be made available on the BioMOBIUS website:

www.biomobius.org.

Start the EyesWeb GDE and confirm that the database blocks appear as shown in the BioMOBIUSGenericDatabase catalogue.



Figure 2 Generic database catalog in GDE

Before using the blocks, download the relevant DB DLL/driver for your targeted RDBMS and ensure the DLL/driver is available – i.e. it is stored either in the BioMOBIUS install folder or is available using the environment path variable. The following sections detail how the DB blocks and the targeted DBMS interface – again ensure your RDBMS is setup properly before incorporating the blocks in a patch or running the sample patches.

2.1 SQLite Install

SQLite is an open source, efficient, file based, low footprint RDBMS that is used in many industrial applications. It is available from:

<http://www.sqlite.org>

From the 'Precompiled Binaries for Windows' section, download and extract:

- The command line program for modifying and SQLite databases
- The DLL of the SQLite library

Setup is simple, at runtime BioMOBIUS only requires that the `sqlite3.dll` is either present in the BioMOBIUS install folder e.g. "c:\program files\BioMOBIUSx.x\" or it is available via the PATH environment variable. It is recommended that you copy the DLL into the BioMOBIUS install folder. There are no configuration parameters to be tweaked.

To create/modify a database you will use the SQLite3 command line utility - `SQLite3.exe`, again this should be available through the PATH environment variable. There are other GUI based DB management tools available but `SQLite3.exe` and `SQLite3.DLL` are the only two components required by BioMOBIUS.

For those unacquainted with SQLite, there are many resources available ranging from user groups and tutorials – for example:

http://souptonuts.sourceforge.net/readme_sqlite_tutorial.html

3 Design

The blocks are designed to provide the basic SQL store and retrieve functionality for EyesWeb datatypes using standard relational database systems. The design is a compromise resulting from the conflicting objectives of on the one hand attempting to keep the blocks as generic as possible while on the other hand ensuring the EyesWeb data model is adhered to. This tussle inevitably leads to functional eccentricities and these are detailed in the guidelines and usage restrictions.

In theory, the design allows the persistence and query of any number of data inputs/outputs of any EyesWeb datatype to any RDBMS database. In practice, the number and nature of the data input/outputs is restricted by the host platform and the type of database is restricted by the availability of an appropriate driver.

3.1 Usage Model

1. User creates all necessary objects in the targeted database system – tables, views, triggers, etc. This is normally accomplished **prior** to patch creation using a proprietary console application. Alternatively, the user may use the BioMOBIUS blocks to create or alter database objects at runtime.
2. User incorporates the BioMOBIUS database blocks in an EyesWeb patch to store and retrieve EyesWeb datatypes at runtime as required. The blocks support a variable number of inputs and outputs thus offering a high level of flexibility in that they are capable of working with any number and combination of data objects.

WARNING: The blocks function as specified but because of their generic nature there are some straight forward rules that must be adhered to:

1. The number of parameters or their order in a partial INSERT or QUERY command cannot change at runtime, obviously values may change at runtime.
2. In the case of result sets which contain non-primitive EyesWeb datatypes, then the InitInfo table must contain the correct initialisation data for each non-primitive datatype attribute/column. This is automatically achieved when the SELECT matches the original INSERT command. In the case of partial INSERTS, then you can only query those attributes/columns with non-primitives EyesWeb datatypes which have a corresponding entry in the INSERT command.
3. All DML block input pins must be connected otherwise the patch will run but the block will be de-activated.
4. Block inputs/outputs should match the targeted DB structure, for example the order of the DML block inputs should match the order of the columns in the targeted database table.
5. If the structure of a database table changes after a DB block has been configured on a patch, then the recommended action is to delete the existing block and add a new DB block.

6. Because a patch retains its state in an XML file between patch runs, changing the number of inputs/outputs on the DB blocks and altering the upline/downline blocks between patch runs may lead to unpredictable behaviour. There are no issues with a correctly configured DB block between patch runs but if a DB block needs to be reconfigured after a patch has run, then the recommendation is that the user delete and recreate the DB block rather than alter the current configuration.

7. An InitInfo table that matches the data in the database must exist prior to querying a database. The Query block uses this table to re-instantiate the EyesWeb data objects. This table is automatically created when the DML block executes. The InitInfo table must remain in synch with its related data table - changing a data table will automatically create a new InitInfo table

3.2 BioMOBIUS Database Blocks

Two blocks are provided to support the standard SQL command set as follows:

- **DB_DML:** This block supports the standard data manipulation language subset – e.g. insert, update and delete. The block supports parameterized and non-parameterized commands.
- **DB_Query:** This block supports the standard select command.

The standard EyesWeb interface – parameters, inputs and outputs – are used to send/retrieve and control block/database interaction in the following way:

- The user configures the DB connection as specified by the targeted DB system. This may be as simple as specifying a filename in the local file system for SQLite or alternatively specifying the address, access parameters and firewall port for a remote DB system
- The user configures the block's properties to match the underlying database table – e.g. number of inputs/outputs and database connection parameters. The number of inputs/outputs may correspond to all the columns in the table or to a subset of columns.
- To persist EyesWeb data objects the user connects these data objects to the block's input pins and triggers the block to persist the data when required.
- Similarly, the user retrieves data from the database on a trigger, the block creates the EyesWeb data objects and exports them on a preconfigured set of output pins.
- Both blocks provide an output pin to record the result of the last operation. This pin is used to flag success or return an error code from the targeted database system.

3.3 EyesWeb Datatypes

EyesWeb datatypes other than the primitive types bool, integer, double and string, consist of two components – the data component and initialization component. Because of this, the BioMOBIUS database blocks maintain two tables to store the EyesWeb non-primitive datatypes – e.g. matrices, images audio buffer. The user interacts normally with the data table to store and retrieve data while the blocks automatically maintain a parallel table to store and retrieve the corresponding initialization data – referred to as the InitInfo table.

This requirement imposes a small limitation on the BioMOBIUS database blocks in that each instance of a block can only map to a single database table if non-primitive types are involved. This is discussed further in the usage section below.

The following EyesWeb datatypes are supported:

3.3.1 Primitive Datatypes

Bool
Integer
Double
String

3.3.2 Non-Primitive Datatypes:

Integer Matrix
Double Matrix
Image
Datetime
Audio Buffer

3.3.3 Datatype Mapping

EyesWeb datatypes are mapped to one of the DB system domains according to the following tables. The user must ensure the mappings are adhered to when creating database tables.

EyesWeb Datatype	SQLite3 Domain
Bool	Integer
Integer	Integer
Double	Real
String	Text
Integer Matrix	Blob
Double Matrix	Blob
Image	Blob
Datetime	Blob
Audio Buffer	Blob

4 Configuration

The BioMOBIUS database blocks are contained in the BioMOBIUSGenericDatabase catalog and in the BioMOBIUSDatabase.Generic library. This section describes how one configures and uses these database blocks within the EyesWeb GDE.

4.1 DB_DML

4.1.1 Block Signature

Label: DB_DML

Catalogs: BioMOBIUSGenericDatabase

Libraries: BioMOBIUS.Database.Generic

4.1.2 Block Description

Supports the modification of a database relation using the standard database manipulation language commands - e.g. INSERT, UPDATE and DELETE. The block supports both parameterized and non-parameterized commands, e.g.

```
INSERT INTO tbltest1 (id, name) VALUES (1,'Joe Bloggs');  
INSERT INTO tbltest1 (id, name) VALUES (?,?);
```

In the case of the parameterized command, the values are determined from the block's input pins. In this case the user first specifies the number of inputs to match the table columns that will be manipulated and then connects the corresponding input pins such that their EyesWeb datatypes map to the correct DB domain type as listed in section 3.1.3.3 above.

The following rules apply:

1. The number of parameters in a parameterised DML command must match the number of block inputs.
2. The command string may not change at runtime – connecting a string to the command parameter pin has no effect.
3. All inputs must be connected otherwise the patch runs but the DML block is deactivated.

4.1.3 Block Icon



4.1.4 Parameters

DML Command – Parameterized or non parameterized command. This command string must conform the SQL standard and terminated in a `;`

DB Type - Specifies the targeted database type – e.g. SQLite3, MySQL or ODBC.

DB Connection String – Specifies the location and access parameters and name of the database that contains the targeted table. In the case of SQLite3, this is simply the name of a file on the local file system.

Number of Inputs – This value must equal the number of values that are specified in the parameterized command. Increasing or decreasing this value appends or removes the last series of input pins from the block.

Init Info Table – Specifies the name of the temporary table that is used to hold the init info associated with any non-primitive EyesWeb datatypes. The default value is `tblInitInfo`. A unique table name must be specified for each table in the database that a DML block manipulates. For example, if there are two DB_DML blocks in the patch manipulating tables tbltest1 and tbltest2, then you must specify unique names for the corresponding InitInfo tables – e.g. tbltest1InitInfo and tbltest2InitInfo.

4.1.5 Inputs

Trigger – The database table is accessed on each trigger received on this input. Note that access does not occur on input pin updates.

DataInx – A series of generic datatype inputs to which the user connects the EyesWeb data objects.

4.1.6 Outputs

Result Flag – an integer value corresponding to an error code returned from the targeted database system. A zero value indicates successful completion of the DML command.

4.1.7 Unusual behavior

4.1.8 Other Comments

4.1.9 Parent

BioMOBIUS.Database

4.2 DB_Query

4.2.1 Block Signature

Label: DB_Query

Catalogs: BioMOBIUSGenericDatabase

Libraries: BioMOBIUS.Database.Generic

4.2.2 Block Description

Supports the querying of a database relation using the standard SQL SELECT command. The block supports both full, partial and predicated SELECT commands e.g.

```
SELECT * FROM tbltest1;  
SELECT id FROM tbltest1;  
SELECT name FROM tbltest1 WHERE id=1;
```

The following rules apply:

1. The order of the columns in a partial query command should match the order of the columns in the DB table. A correct example is:

```
SELECT colName1, colName3 FROM tbltest1;
```

while the following is an incorrect example:

```
SELECT colName3, colName1 FROM tbltest1;
```

2. Certain predicated commands may not function across all RDBMS API's, For example, the ORDER BY predicate is not supported by SQLite3.

4.2.3 Block Icon



4.2.4 Parameters

SQL Query Command – specifies the query to execute against the targeted database table. The command must conform to the standard SQL query syntax and terminate in a `;`. Note that the result set must consist of columns from a table whose InitInfo was previously created by the DB_DML block.

DB Type - Specifies the targeted database engine – e.g. SQLite3, MySQL or ODBC.

DB Connection String – Specifies the location and access parameters and name of the database that contains the targeted table(s). In the case of SQLite3, this is simply the name of a file on the local file system.

Number of Outputs – This value must equal the number of values that are specified in the query command. Note the current version contains the imposition in that changing this value from a non-zero value is best accomplished by removing the block from the patch and re-inserting.

On changing this number, a parameter pin is automatically created for each output. An output pin is not created on the block until the datatype parameter is specified.

Init Info Table – Specifies the name of the temporary table that contains the init info associated with any non-primitive EyesWeb datatypes in the targeted table. The default value is `tblInitInfo`. A unique table name must be specified for each table in the database that a DML block has manipulated. For example, if there are two DB_Query blocks in the patch querying tables tbltest1 and tbltest2, then you must specify the unique names for the corresponding initinfo tables that were previously configured for the DB_DML blocks.

Trigger Type – Specifies as to how the query is triggered. The options here are to trigger using an input pin or alternatively trigger on changes to the query command string. If the former is specified then an input pin is created automatically and assigned the EyesWeb Trigger datatype.

If the blocks triggers on a change to the command string, then the number and order of column names must remain constant.

O/P Datatype_x – Specifies the EyesWeb datatype for each output pin. When the block is initially created each output pin defaults to `Not`

Defined'. The user selects a valid datatype from the dropdown list corresponding to the datatype in the matching database table column. Note that changing the output datatype more than once for a single pin may cause the patch to become unstable and the best practice is to recreate the DB_Query from scratch if datatypes are changed multiple times.

4.2.5 Inputs

Trigger – This is an optional input - the database table is accessed on each trigger received on this input.

4.2.6 Outputs

Result Flag – an integer value corresponding to an error code returned from the targeted database system. A zero value indicates successful completion of the Query command.

4.2.7 Unusual behavior

4.2.8 Other Comments

4.2.9 Parent

BioMOBIUS.Database

These patches will persist objects corresponding to all the supported EyesWeb datatypes and subsequently allow you to retrieve them from a database and re-instantiate the objects.

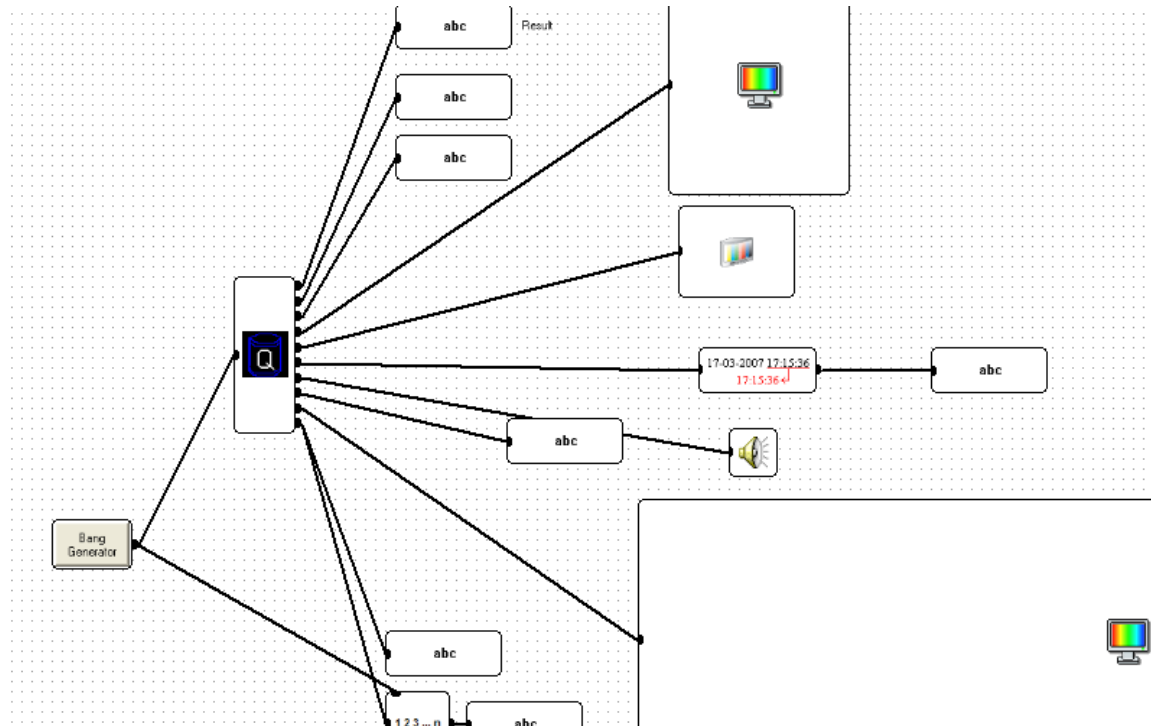


Figure 4 QueryDatatypeTest.eywx

There are a number of guidelines to observe when running these test patches:

1. The targeted DBMS interface must be in place – i.e. relevant drivers are in the correct locations and configured correctly as outlined in section 2. The DB management application must also be available using the PATH environment variable.
2. The test database is created using the supplied script prior to running the test patches.
3. Persist data to the database before querying the data

Follow these steps to run the test patches:

Step 1: Create the test database for your targeted DBMS.

For SQLite:

1. Open a command line box and change directory to the folder:
`C:\biomobius_workspace\Patches\TestDBResourceFiles`
2. Execute the command :
`SQLite3 TestDB_Blocks.db < TestDB_SQLite3.sql`

Using the full path names the command would look something like this:

```
From the
C:\biomobius_workspace\Patches\TestDBResourceFiles>
Directory
Type
"c:\sqlite-3_6_14\sqlite3.exe"
TestDB_Blocks.db < TestDB_SQLite3.sql
For example
```

This will create the TestDB_Blocks.db database file containing the test database table tblTest1 in the C:\biomobius_workspace\Patches\TestDBResourceFiles folder.

Step 2: Persist some data. Double click on the DMLDatatypeTest patch and the patch will load in the EyesWeb GDE. Run the patch and a new row is written to the tblTEST1 database table each time you click on the bang button.

Step 3: Query the data. Load the QueryDatatypeTest.eywx patch in the GDE and click on the bang button. This patch will retrieve all the rows from the tblTEST1 table and display them.

Step 4: Check that the number of rows returned in the query patch matches the number of rows that you create with the DML patch in step 2.

6 Troubleshooting

This section describes troubleshooting steps, known limitations and back up material to aid the use of these database blocks.

6.1 Common Errors

6.1.1 Failed to initialise the InitInfo data objects.....

This error indicates that the database engine could not create or locate the temporary InitInfo table. This usually results from either specifying the wrong target database in the block or alternatively specifying the wrong InitInfo table. In the case of the query block, ensure that the InitInfo table matches that of the corresponding DB_DML block.

6.1.2 Unserialize failed.....

The block could not retrieve the data objects binary info from the database. Use the database console application to check if the specified InitInfo table contains a single row of data. If not then either the wrong table has been specified or the EyesWeb datatype attached to the DB_DML block does not match that specified for the corresponding output on the DB_Query block.

6.1.3 Error initializing the EyesWeb execution engine.....

Such an error in the case of the DB_Query block indicates that there is a mismatch in the mapping of the EyesWeb datatypes. Typically, data on an output pin is not compatible with the input of the down line block. Check the SELECT command syntax and database table structure to ensure that the correct mapping is used. Alternatively, redefine the query block adding one output at a time and determine the errant output accordingly.

6.1.4 Failed to initialize the column/index mapping.....

This error usually indicates that the specified table in the SQL command does not exist.

The following blocks are excluded from execution:

6.1.5 The following blocks are excluded from execution.....

This error occurs because all the block inputs are not connected to the upline block outputs.

6.2 Supported Database Systems

The BioMOBIUS database blocks have been tested on the following DB system versions:

Database System	Version
SQLite3 Direct	V 3.5.9
MySQL Direct	TBD
ODBC/MySQL	TBD
ODBC/SQL Server	TBD

6.3 Limitations

There follows a list of known limitations in the usage of the blocks. These may be eliminated or by passed in the future. Please forward any further usage issues to the BioMOBIUS web site.

6.3.1 Support for query joins.

The necessity of maintaining initinfo data in parallel with the object data prohibits query joins where the result set contains non-primitive EyesWeb attributes from more than one table.

6.3.2 Removing input and output pins

The effort required in supporting the removal of redundant block inputs and outputs is large and ommitted from the original release. The current workaround where the user deletes the block and reinstates it and configures its pins is not too heavy a burden unless one is dealing with a very large number of pins.