Global Information System of the ITPGRFA

Integration Toolkit

Installation and operation manual

Version 2.0.5 - 27/07/2023

History of changes

Version	Date	Description
2.0.5	27/07/2023	Updated GLIS URL
2.0.4	01/07/2020	Added column size to database tables
2.0.3	17/04/2019	Corrected an error in the description of the `target` table in the simplified database
2.0.2	13/03/2019	Added `jdbcDriver` directory and explained how to use it. Explained how to use DBeaver on the embedded database
2.0.1	27/02/2019	Added `doi.log` to `config.txt` and explained what should be done on the `results` table after each run
2.0	19/02/2019	New version
1.0	25/10/2017	Initial public version

Introduction

The Global Information System (GLIS) of the International Treaty on Plant Genetic Resources for Food and Agriculture (ITPGRFA), as described in Article 17 of the ITPGRFA, aims at facilitating "the exchange of information, based on existing systems, on scientific, technical and environmental matters related to plant genetic resources for food and agriculture". The cornerstone of GLIS are information on Plant Genetic Resources for Food and Agriculture (PGRFAs) as they are made available on the Web.
To facilitate finding and accessing such information, GLIS offers the assignation of Digital Object Identifiers (DOIs) to PGRFAs and the collection of links to information resources (targets) available on the Web. The registration of PGRFAs in GLIS to obtain the corresponding DOIs is obtained, among other options, through a XML-based protocol described at
http://www.fao.org/plant-treaty/areas-of-work/global-information-system/techdoc/en/
To promote such XML protocol, the Treaty Secretariat has developed an Integration Toolkit (or Toolkit in short) providing the necessary XML formatting and communication layer thus greatly simplifying the operation for adopting stakeholders.
Adoption of the Toolkit is not required to participate in the GLIS initiative, it is offered as an alternative to those stakeholders that are unable or unwilling to implement the XML protocol but have a large enough collection to make the other available options unpractical.
Future versions will likely extend the features of the Toolkit to cover other services beyond those described in this document.
For any clarification, support request or to contact the Treaty Secretariat, please send an email to PGRFA-Treaty@fao.org.

System prerequisites

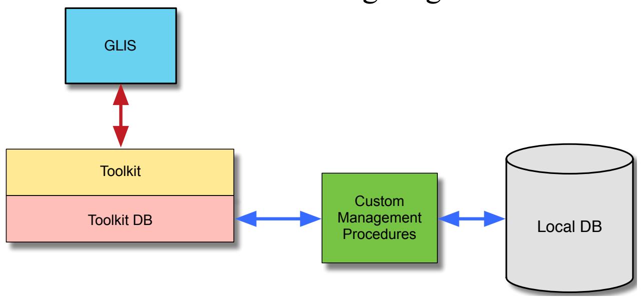
The Toolkit is a Java application and therefore can be installed on any Java-compatible Operating System such as Linux/UNIX, MacOS and Windows. The Toolkit comes with its own embedded RDBMS but it can connect to any JDBC-compliant database. The detailed system requirements are:

- Linux, any recent distribution such as Ubuntu 16.04 or later
- MacOS, any recent distribution such as 10.3 or later
- Oracle Java 1.8.0_101 or later with JAVA_HOME properly set
- 128MB RAM
- 128MB hard disk space or more, depending on the size of your collection

Java version 1.8.0_101 or later is required to use the GLIS test server. Please see the "Troubleshooting tips" section at the end of this document for details.

Adopting the Toolkit

The Toolkit can be seen as a middleware taking information from the database, converting it to the corresponding XML message, sending it to GLIS and recording the result back to the database. The rationale of this architecture being that it is more likely to find enough expertise in interested stakeholders to read and write data to a database than it would be to implement a robust XML communication layer.
The resulting architecture is described in the following diagram.



Adopting the Toolkit requires the implementation of the “Custom Management Procedures” box. This component will essentially contain the logic required to extract information from the local database mapping its structure to that of the Toolkit DB for the descriptors that need to be sent to and received from GLIS. This activity will need to take into account the documents available at:
http://www.fao.org/plant-treaty/areas-of-work/global-information-system/guidelines/en/
http://www.fao.org/plant-treaty/areas-of-work/global-information-system/descriptors/en/
for, respectively, use cases for the assignation of DOIs and the descriptors involved. Although it is recommended that a suitable application is developed to implement the “Custom Management Procedures”, it is possible to just define a set of database operations exporting the necessary data from the local DB and writing back the result of the operation returned by GLIS.
At their minimum, the "Custom Management Procedures" can be implemented as an extraction of the required information from the local DB, the manual insertion of such information into the Toolkit DB and the reverse operation to extract the assigned DOIs from the Toolkit database and writing them into the local DB.
The Toolkit is supposed to operate as a black box reducing the burden on the stakeholder being guaranteed that, if proper information is provided to the Toolkit DB, the transaction with GLIS will be successful and that future development of the GLIS workflow will be supported by the corresponding release of the Toolkit.

Using the Toolkit Local DB

To operate, the Toolkit needs to read information about the PGRFA to register or update on GLIS. This information comes from a JDBC-compliant database, either the one already present in your institution or organization, or the one provided embedded in the Toolkit itself. In the first case, you need to create the necessary tables in your own database, define a user account to be used by the Toolkit to access such tables and configure the Toolkit to use that DB.
The Toolkit comes already configured to use its own embedded database (see "Configuration file" below)

The Toolkit database versions

The Toolkit supports two different database version: a full one, already used in the previous version of the Toolkit and a new, simplified version that is intended to facilitate the insertion of data from your own DB.
The Toolkit database is used to exchange data to and from your local database when you don't want or cannot create in it the tables required by the Toolkit. Essentially, you write data to the Toolkit database, execute the Toolkit and check the result of such operation in the Toolkit database (or in the log files, see "Log files" below).

The Toolkit is indeed a processor of the XML messages described in the "XML Integration Protocol" document available at
http://www.fao.org/plant-treaty/areas-of-work/global-information-system/techdoc/en/
The content of the database is very closely related to the XML elements described in the document with some exceptions described below. Otherwise, please refer to the XML element indicated next to table columns, when applicable.

The full Toolkit database

The full Toolkit database was used by the previous version of the Toolkit (the one build on top of WSO2). Those stakeholders who adopted the previous version of the Toolkit, and therefore have already installed the corresponding database and have developed the procedures necessary to read and write to it, can continue using the same database schema, although some tables are not used anymore. The full database for the new version of the Toolkit includes the following tables:

actors

Stores information about providers, collectors and breeders associated to the PGRFA. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. Any number of collectors and breeders can be present, but only one provider. Please note that, if at least one between the WIEWS code or the Easy-SMTA PID are available, name, address and country can be left empty as they will be obtained from the codes. Columns are as follows:

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`pgrfa_id`	BIGINT NOT NULL	Foreign key to `pgrfas.id`
`role`	CHARACTER(2) NOT NULL	A 2-letter code identifying the actor role. `pr`: provider, `co`: collector and `br`: breeder
`wiews`	VARCHAR(16)	The FAO/WIES Institute code, if available
`pid`	VARCHAR(16)	The Easy-SMTA PID, if available
`name`	VARCHAR(128)	The organization name or the name of the individual
`address`	VARCHAR(128)	The organization address of the individual address
`country`	CHARCTER(3)	The ISO-3 code of the organization country or the individual country

identifiers

Stores information about additional identifiers not already provided in the `pgrfas` table. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. Any number of identifiers can be associated to a PGRFA. Columns are as follows:

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`pgrfa_id`	BIGINT NOT NULL	Foreign key to `pgrfas.id`
`type`	VARCHAR(16)	The code of the identifier type. See Table 4 of the XML Integration protocol document
`value`	VARCHAR(128)	The identifier value

names

Stores names associated to the PGRFA. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. Any number of names can be associated to a PGRFA, but it is critical to provide at least the English. Crop names are the names used by people to refer to the crop (e.g. "wheat", "rice" and so on). Other names, instead, are usually variety or cultivar names. Columns are as follows:

Column	Type	Description
id	BIGINT NOT NULL	Primary key
pgrfa_id	BIGINT NOT NULL	Foreign key to pgrfas.id
name_type	VARCHAR(2) NOT NULL	A 2-letter code of the type of the name. cn: common name, on: other name
name	VARCHAR(128) NOT NULL	The actual common or other name

pgrfas

This is the main table and stores information associated to the PGRFA. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. Columns are as follows:

Column	Type	Description
id	BIGINT NOT NULL	Primary key
operation	VARCHAR(8) NOT NULL	The operation associated to the PGRFA. Values recognized by the Toolkit are: register or update
sample_id	VARCHAR(128) NOT NULL	The sample identifier in your collection or the identifier in the provider's collection
processed	CHARACTER(1) NOT NULL	y or n indicating whether the row has been processed, regardless of the outcome, see the results table. The Toolkit processes only rows where processed='n'
sample_doi	VARCHAR(128)	The DOI associated to the PGRFA. The role of this value depends on the operation as follows: register leave empty if you want GLIS to assign a DOI to the PGRFA. Otherwise enter here the existing DOI you obtained using some other service and are registering the PGRFA to GLIS using this DOI update must contain the DOI (minted by GLIS or not) associated to the PGRFA you wish to update that must be already registered in GLIS
date	VARCHAR(10) NOT NULL	See XML Integration protocol [date] M03
hold_wiews	VARCHAR(16)	See XML Integration protocol [lwiews] M01
hold_pid	VARCHAR(16)	See XML Integration protocol [lpid] M01
hold_name	VARCHAR(128)	See XML Integration protocol [lname] M01. Can be left empty if at least one between hold_wiews and hold_pid is provided
hold_address	VARCHAR(128)	See XML Integration protocol [laddress] M01. Can be left empty if at least one between hold_wiews and hold_pid is provided
hold_country	CHARACTER(3)	See XML Integration protocol [lcountry] M01. Can be left empty if at least one between hold_wiews and hold_pid is provided
method	CHARACTER(4) NOT NULL	See XML Integration protocol [method] M04
genus	VARCHAR(64)	See XML Integration protocol [genus] M05
species	VARCHAR(128)	See XML Integration protocol [species] R04
sp_auth	VARCHAR(64)	See XML Integration protocol [spauth] R04
subtaxa	VARCHAR(128)	See XML Integration protocol [subtaxa] R04
st_auth	VARCHAR(64)	See XML Integration protocol [stauth] R04
bio_status	CHARACTER(3)	See XML Integration protocol [biostatus] R03
mls_status	VARCHAR(2)	See XML Integration protocol [mlsstatus] R07
historical	CHARCATER(1)	See XML Integration protocol [historical] R08
prov_sid	VARCHAR(128)	See XML Integration protocol [psampleid] A02
provenance	CHARACTER(3)	See XML Integration protocol [provenance] A03
coll_sid	VARCHAR(128)	See XML Integration protocol [csampleid] A05
coll_miss_id	VARCHAR(128)	See XML Integration protocol [missid] A06
coll_site	VARCHAR(128)	See XML Integration protocol [site] A06
coll_lat	VARCHAR(128)	See XML Integration protocol [clat] A06
coll_lon	VARCHAR(128)	See XML Integration protocol [clon] A06
coll_uncert	VARCHAR(128)	See XML Integration protocol [uncert] A10

`coll_datum`	VARCHAR(16)	See XML Integration protocol `[datum] A11`
`coll_georef`	VARCHAR(16)	See XML Integration protocol `[georef] A12`
`coll_elevation`	INTEGER	See XML Integration protocol `[elevation] A13`
`coll_date`	VARCHAR(10)	See XML Integration protocol `[cdate] A14`
`coll_source`	CHARACTER(2)	See XML Integration protocol `[source] A15`
`ancestry`	VARCHAR(32768)	See XML Integration protocol `[ancestry] A17`

progdois

Stores the DOI(s) of the progenitor(s) associated to the PGRFA. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. The number of DOIs that can be present for a given PGRFA depends on the method; please see the XML Integration protocol document for details. As the progenitor's DOI must be already registered in GLIS; it is necessary to first register the PGRFA ancestors and then their progeny. Columns are as follows:

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`pgrfa_id`	BIGINT NOT NULL	Foreign key to `pgrfas.id`
`doi`	VARCHAR(128) NOT NULL	The progenitor's DOI. Must be already registered in GLIS

results

This table stores the result of the operation applied to the corresponding row of `pgrfas`. Columns are as follows:

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`operation`	VARCHAR(8) NOT NULL	The operation that was requested (see `pgrfas` above)
`genus`	VARCHAR(64)	The genus of the PGRFA
`sample_id`	BIGINT NOT NULL	The sample ID passed in the column `pgrfas.sample_id`
`doi`	VARCHAR(128)	The DOI associated to the PGRFA. For registration operations, this is the newly minted DOI associated to the PGRFA or the already assigned DOI that was passed in `pgrfas.sample_doi`
`result`	`VARCHAR(2) NOT NULL`	`OK` or `KO` depending whether the operation was successful or not
`error`	VARCHAR(32768)	The error message returned by GLIS, if any

targets

Stores the targets, i.e. links to associated information on the PGRFA available on the Internet. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. Columns are as follows:

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`pgrfa_id`	BIGINT NOT NULL	Foreign key to `pgrfas.id`
`value`	VARCHAR(256) NOT NULL	The target value (usually the URL to the information resource associated to the PGRFA)

tkws

Stores the target keyword codes, i.e. the codes corresponding to the keywords associated to each target. It is used for registration of PGRFAs to GLIS and update of already registered PGRFAs. Columns are as follows:

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`target_id`	BIGINT NOT NULL	Foreign key to `targets.id`
`value`	VARCHAR(128) NOT NULL	The keyword code. Please see XML Integration Protocol table 2

transfers

transfer_materials

transfer_results

These tables, if present in your database, can be safely ignored until the new Toolkit will support the transfer transaction.

The simplified Toolkit database

The simplified Toolkit database includes tables that are in most cases the same as for the full database. Changes to the full database are highlighted red. For details on the tables, please refer to the corresponding table in the full database above. The main difference is using the column `pgrfas.sample_id` as reference to relate the other tables. This simplifies importing data into the Toolkit database because `sample_id` is immediately known and table files can be fully prepared in advance. Another difference is the removal of the `progdois` and `tkws` tables that are now available as columns in `pgrfas` and `targets`.

actors

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`sample_id`	VARCHAR(128) NOT NULL	Foreign key to `pgrfas.sample_id`
`role`	CHARACTER(2) NOT NULL	A 2-letter code identifying the actor role. `pr`: provider, `co`: collector and `br`: breeder
`wiews`	VARCHAR(16)	The FAO/WIES Institute code, if available
`pid`	VARCHAR(16)	The Easy-SMTA PID, if available
`name`	VARCHAR(128)	The organization name or the name of the individual
`address`	VARCHAR(128)	The organization address of the individual address
`country`	CHARACTER(3)	The ISO-3 code of the organization country or the individual country

identifiers

Column		Description
`id`	BIGINT NOT NULL	Primary key
`sample_id`	VARCHAR(128) NOT NULL	Foreign key to `pgrfas.sample_id`
`type`	VARCHAR(16) NOT NULL	The code of the identifier type. See Table 4 of the XML Integration protocol document
`value`	VARCHAR(128) NOT NULL	The identifier value

names

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`sample_id`	VARCHAR(128) NOT NULL	Foreign key to `pgrfas.sample_id`
`name_type`	VARCHAR(2) NOT NULL	A 2-letter code of the type of the name. `cn`: common name, `on`: other name
`name`	VARCHAR(128) NOT NULL	The actual common or other name

pgrfas

Column	Type	Description
`id`	BIGINT NOT NULL	Primary key
`operation`	VARCHAR(8) NOT NULL	The operation associated to the PGRFA. Values recognized by the Toolkit are: `register` or `update`
`sample_id`	VARCHAR(128) NOT NULL	The sample identifier in your collection or the identifier in the provider's collection. Must be unique in the table
`processed`	`CHARACTER(1) NOT NULL`	`y` or `n` indicating whether the row has been processed, regardless of the outcome, see the `results` table. The Toolkit processes only rows where `processed='n'`
`sample_doi`	VARCHAR(128)	The DOI associated to the PGRFA. The role of this value depends on the operation as follows: `register` leave empty if you want GLIS to assign a DOI to the PGRFA. Otherwise enter here the existing DOI you obtained using some other service and are registering the PGRFA to GLIS using this DOI `update` must contain the DOI (minted by GLIS or not) associated to the PGRFA you wish to update that must be already registered in GLIS
`date`	VARCHAR(10) NOT NULL	See XML Integration protocol `[date] M03`
`hold_wiews`	VARCHAR(16)	See XML Integration protocol `[lwiews] M01`
`hold_pid`	VARCHAR(16)	See XML Integration protocol `[lpid] M01`
`hold_name`	VARCHAR(128)	See XML Integration protocol `[lname] M01`. Can be left empty if at least one between `hold_wiews` and `hold_pid` is provided
`hold_address`	VARCHAR(128)	See XML Integration protocol `[laddress] M01`. Can be left empty if at least one between `hold_wiews` and `hold_pid` is provided

hold_country	CHARACTER(3)	See XML Integration protocol [lcountry] M01. Can be left empty if at least one between hold_wiews and hold_pid is provided
method	CHARACTER(4) NOT NULL	See XML Integration protocol [method] M04
genus	VARCHAR(64)	See XML Integration protocol [genus] M05
species	VARCHAR(128)	See XML Integration protocol [species] R04
sp_auth	VARCHAR(64)	See XML Integration protocol [spauth] R04
subtaxa	VARCHAR(128)	See XML Integration protocol [subtaxa] R04
st_auth	VARCHAR(64)	See XML Integration protocol [stauth] R04
bio_status	CHARACTER(3)	See XML Integration protocol [biostatus] R03
mls_status	VARCHAR(2)	See XML Integration protocol [mlsstatus] R07
historical	CHARCATER(1)	See XML Integration protocol [historical] R08
prov_sid	VARCHAR(128)	See XML Integration protocol [psampleid] A02
provenance	CHARACTER(3)	See XML Integration protocol [provenance] A03
coll_sid	VARCHAR(128)	See XML Integration protocol [csampleid] A05
coll_miss_id	VARCHAR(128)	See XML Integration protocol [missid] A06
coll_site	VARCHAR(128)	See XML Integration protocol [site] A06
coll_lat	VARCHAR(128)	See XML Integration protocol [clat] A06
coll_lon	VARCHAR(128)	See XML Integration protocol [clon] A06
coll_uncert	VARCHAR(128)	See XML Integration protocol [uncert] A10
coll_datum	VARCHAR(16)	See XML Integration protocol [datum] A11
coll_georef	VARCHAR(16)	See XML Integration protocol [georef] A12
coll_elevation	INTEGER	See XML Integration protocol [elevation] A13
coll_date	VARCHAR(10)	See XML Integration protocol [cdate] A14
coll_source	CHARACTER(2)	See XML Integration protocol [source] A15
ancestry	VARCHAR(32768)	See XML Integration protocol [ancestry] A17
progdois	VARCHAR(128)	List of progenitor DOI(s) separated by "|", e.g. "10.18730/22|10.18730/3RGQ"

results

Column		Description
id	BIGINT NOT NULL	Primary key
operation	VARCHAR(8) NOT NULL	The operation that was requested (see pgrfas above)
genus	VARCHAR(64)	The genus of the PGRFA
sample_id	VARCHAR(128) NOT NULL	The sample ID passed in the column pgrfas.sample_id
doi	VARCHAR(128)	The DOI associated to the PGRFA. For registration operations, this is the newly minted DOI associated to the PGRFA or the already assigned DOI that was passed in pgrfas.sample_doi
result	VARCHAR(2) NOT NULL	OK or KO depending whether the operation was successful or not
error	VARCHAR(32768)	The error message returned by GLIS, if any

targets

Column		Description
id	BIGINT NOT NULL	Primary key
sample_id	VARCHAR(128) NOT NULL	Foreign key to pgrfas.sample_id
value	VARCHAR(256)	The target value (usually the URL to the information resource associated to the PGRFA)
tkws	VARCHAR(256)	A list of keyword codes according to table 2 of the XML Integration Protocol document, separated by "|", e.g. "1|2|3.1"

Getting the Toolkit

The Toolkit is a Java application that is distributed as a compressed archive. To obtain it, please contact PGRFA-Treaty@fao.org. You will get a file named GLIS-Toolkit-x.y.zip where x.y is the version number (currently 2.0). Inside the archive, you will find documents providing detailed instructions on how to deploy the Toolkit and all necessary files required to do so. The directory where you expanded the .zip archive will be called TKHome in the following.

Creating the Toolkit database

The Toolkit comes with the embedded database already created, empty, and ready to accept your information. If, instead, you prefer to use your own database, you will find SQL scripts to create the database schema for MySQL, HSQLDB and PostgreSQL in the directory `DB scripts` available in `TKHome`. Create the database, define an account with full privileges on it and run the script to create tables and indexes. Make sure you note the database name and the username and password of the account as you will need them later to configure the Toolkit to access the database.

Using the Toolkit embedded database

If you decide to use the Toolkit embedded database, you do not need to create a database as the Toolkit comes with the database already prepared. The Toolkit starts and stops the embedded database automatically when you run it.
Likewise, if you use a JDBC client to access the embedded database, see "Managing the embedded database below", the database will be locked to other users, including the Toolkit. Therefore, **it is important that the embedded database is not being used by other applications when you run the Toolkit as an error will occur when the Toolkit tries to start the already running embedded database**.

Using the Toolkit with a database different from the embedded one

The Toolkit uses JDBC to access its database. If you do not want to use the embedded database, e.g. because you already have the old Toolkit database up and running, you need to provide a suitable JDBC driver. Please note that the JDBC driver for the embedded database is already bundled in the `toolkit.jar` and you do not need to do anything in this case.
JDBC drivers for databases other than the embedded one can be downloaded from the corresponding websites and must be put inside the `jdbcDriver` directory located in the `TKHome`. Please make sure to download a JDBC driver that is compatible with your database server version.

Configuring the Toolkit

The Toolkit configuration is provided through a text file named `config.txt` that must be located inside the same directory as the `toolkit.jar`. The content of the config.txt file is comprised of lines of text with the syntax

```
key = value
```

where `key` is the configuration parameter and `value` is the value to assign to it. Lines beginning with a `#` are considered comments and ignored. The following table describes the configuration parameters that can be set. Where alternatives are provided, please make sure only one key is not commented!

Key	Description
`db.url`	The JDBC connection URL of the database that the Toolkit will use. By default, it is `jdbc:hsqldb:file:db/glistk` that instructs the Toolkit to use the embedded database. Examples are provided for other RDBMS such as MySQL and PostgreSQL
`db.username`	Is the username used to connect to the database. `glistk` is the username for the embedded database
`db.password`	Is the password used to connect to the database. `glistk` is the password for the embedded database
`db.query_limit`	Is the maximum number of rows in the `pgrfas` table to be processed in the Toolkit run. The default is 1000, but you can set it higher or lower, depending on the speed of your connection, the available RAM, how long you want to wait before looking at the results and so on
`db.version`	Defines what database schema the Toolkit will expect: `1` = full (i.e. the old database schema), `2` = simplified
`glis.url`	Is the URL of the GLIS registration service. The file contains two URLs: - the test URL (`https://glistest.planttreaty.org/glis/xml/manager`), and - the production URL (`https://glis.fao.org/glis/xml/manager`)
`glis.username`	Is the username used to access of the GLIS registration service
`glis.password`	Is the password used to access of the GLIS registration service
`doi.log`	Can be set to `y` or `n`. If set to `y` produces a TAB-separate text file with the newly assigned DOIs. Details are provided below

Please note that `glis.username` and `glis.password` are **not** those you use to login to Easy-SMTA and GLIS on the web. Rather, they are assigned to you by the GLIS System Administrator that you can contact by email to PGRFA-Treaty@fao.org.

When `db.url` refers to a database different from the embedded one, the Toolkit attempts to load the corresponding JDBC driver from the `jdbcDriver` directory located in the `TKHome`. Please make sure that the correct JDBC driver is present in the `jdbcDriver` directory.

The DOI log file

Setting `doi.log = y` in the `config.txt` file produces a TAB-separated text file named

```
<timestamp of execution start>_DOI.txt
```

located in `TKHome`. This file is a convenience to more easily access the newly assigned DOIs rather than having to look into the `results` table (see below). The file is TAB-separated and includes the following columns

- WIEWS code of the holder passed in `pgrfas.holdwiews`
- Easy-SMTA PID of the holder passed in `pgrfas.holdpid`
- Genus passed in `pgrfas.genus`
- Sample ID passed in `pgrfas.sample_ID`
- The newly assigned DOI

Please note that:

- the file only includes successful new registrations. Updates or failed registrations are not included
- if an exception occurs, the content of the file may not be reliable
- the `results` table is the ultimate reference for successful and unsuccessful operations, both registrations and updates
- the `results` table is always written to, regardless of the `doi.log` setting. Please do not forget to perform housekeeping operations on it after each run.

The results table

When the Toolkit executes an operation (e.g. the registration of a new PGRFA), it stores the result of such operation into the `results` table. Your management procedure will look at the `results` table to find out if the operation was successful and collect any output provided (in our example, the DOI assigned by GLIS). Should the operation have failed, the `results` table will provide an error message that can be used to resolve the issue. Please note that the `processed` column in the `pgrfas` table is set to `y` regardless of the outcome of the operation. Therefore, if an error occurred for a specific PGRFA, you should:

- Look at the error message in the `results` table
- Perform any correcting action to eliminate the problem
- Delete the row from the `results` table

- Set the processed column of the corresponding record in `pgrfas` to `n`
- Run the Toolkit to try the operation again with the fix

The housekeeping of the `results` table is your responsibility; when you read a row from the `results` table, you should remove it.
Please keep in mind that each run of the Toolkit may add new rows to the table and that the same Sample ID may appear multiple times if you do not properly clean the table. For instance, assume we have a PGRFA with Sample ID `A`. The initial registration is successful and the following row is added to the `results` table:

id	operation	genus	sample_id	doi	result	error
1	register	Oryza	A	10.18730/W4RTY	OK	null

If the row is not deleted after the DOI has been acquired, some confusion may occur. Let us imagine that, for some reason, the pgrfas row for sample_id `A` (that has been updated with `processed='y'` after the first run), is mistakenly updated with `processed='n'` and a new run is performed. The new run will cause an error because `A` is already registered. The `results` table will contain:

id	operation	genus	sample_id	doi	result	error
1	register	Oryza	A	10.18730/W4RTY	OK	null
2	register	Oryza	A	null	KO	PGRFA sampleid [A], genus [Oryza] already registered for this owner

By looking at the id column, that is strictly sequential, it is possible to clarify that the first run was successful and the second run resulted in an error and why.
The `results` table contains the `sample_id` and the `genus` (if provided in the request) to help you identify the proper PGRFA record in your database. This because, in some institutions, the `sample_id` alone is not sufficient to uniquely identify the PGRFA to which the result applies.

Running the Toolkit

To run the Toolkit, please follow these steps:
make sure that the configuration is correct by inspecting the config.txt file
open a terminal window and go to `TKHome` and type

```
java -jar toolkit.jar
```

This will first print the configuration read from `config.txt`, then it will look for rows in `pgrfas` with `operation = 'register'` and `processed = 'n'` and process them, then for rows with `operation = 'update'` and `processed = 'n'` and process them. In each set, up to `db.query_limit` rows will be processed. During operation, lines will be printed with, at the end, `OK` or `KO` depending whether the operation was successful or not.
When the run is complete, you should access the database (starting it if you are using the embedded one as explained above) and look into the `results` table to find out what happened. More details are provided below.
In order to make sure that only one instance of the Toolkit is running at any given time, the Toolkit creates a lock file in `TKHome` named `lock.lck`. This file is automatically deleted when the execution completes without problems. However, if the Toolkit exits abnormally, the lock file will not be deleted. To run the Toolkit again, please make sure to delete the lock file.

PGRFA registration to obtain the corresponding DOI

The purpose of this function is to register a PGRFA to GLIS and obtain the newly assigned DOI. To do so, you will populate some of the Toolkit's tables with descriptors extracted from your database.

The tables in the Toolkit’s database involved in this function are:

`pgrfas`	This is the base table containing one row for each PGRFA to be processed. Populate the row with the descriptors coming from your local database. Make sure that processed is set to `n` until all corresponding rows in related tables are completely populated. For this function, the column `sample_doi` must be left `NULL`.
`actors`	This table stores information on the provider, collector(s) and breeder(s) associated to the `pgrfas` row. The `pgrfa_id` column is the foreign key to the column `pgrfas.id`. For any given PGRFA, there must be zero or one provider; instead, zero or more collectors and zero or more breeders can be provided. The role column is as follows:

`pr`	Provider
`co`	Collector
`br`	Breeder

`identifiers`	This table stores the other identifiers associated to the PGRFA
`names`	This table stores the crop (`cn`) names and the other (`on`) names
`progdois`	This table stores the progenitor(s) DOIs. Such DOIs must be already registered to GLIS (only for the full database. `progdois` is a column of the `pgrfas` table in the simplified database)
`targets`	This table stores the URLs of web resources about the PGRFA
`tkws`	This table stores the keyword codes associated to each target pointing to `targets.id` as foreign key (only for the full database. `tkws` is a column of the `targets` table in the simplified database)
`results`	This table stores the result of the registration request

Once the tables are populated with the information provided, run the Toolkit and look at the `results` table to get the DOI associated to each PGFRA or identify what went wrong.

Update of descriptors associated with a PGRFA

The purpose of this function is to update one or more descriptors associated with a PGRFA already registered to GLIS. To do so, you will populate some of the Toolkit’s tables with descriptors extracted from your database. You must provide all applicable descriptors, not just the changed ones, because GLIS replaces the entire PGRFA record with what you provide.
The tables in the Toolkit’s database involved in this function are the same as for registration with:

`pgrfas.operation='update'`	to inform the Toolkit that an update is required and not a registration
`pgrfas.sample_doi`	set to the DOI associated to the PGRFA

Providing `pgrfas.sample_doi` is critical to uniquely identify the PGRFA being updated. As explained before, in fact, the `sample_id` may not be unique, even for the same holder.
Once the tables are populated with the information provided, run the Toolkit and look at the `results` table to get the DOI associated to each PGFRA or identify what went wrong.

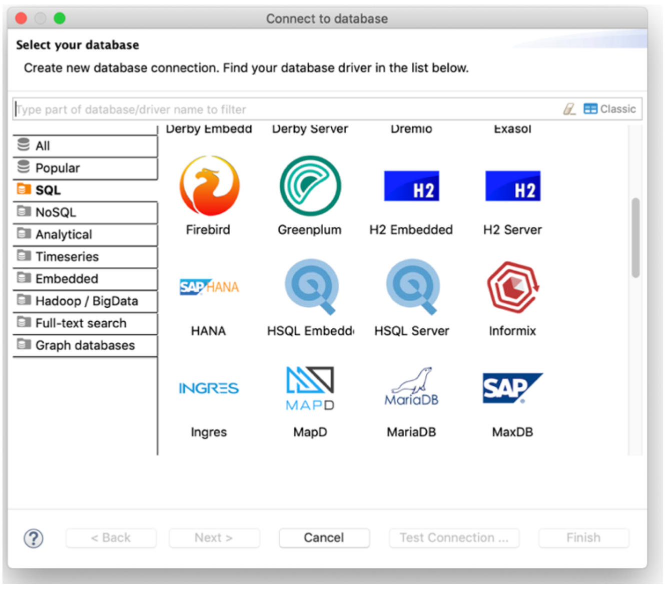
Managing the embedded database

To access the embedded database, e.g. to import or export records, the easiest solution would be to use any of the several free JDBC clients available. One we have tested is DBeaver (https://dbeaver.io/download/) but others may be just as good or even better. In the following, we briefly describe how to obtain, configure and use DBeaver on the embedded database but some of the information provided will be useful for other tools as well.
Please be aware that the Toolkit must not be running when attempting to access the embedded database using a JDBC client! This is because access to the database in embedded

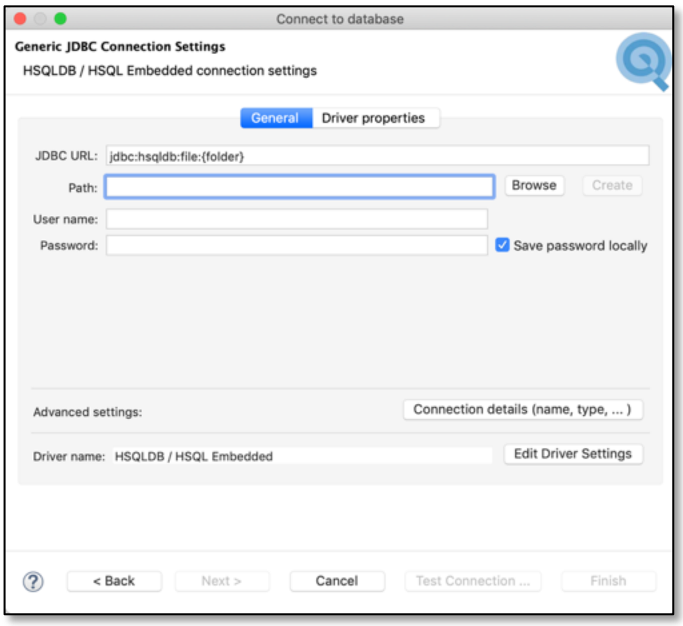
mode is possible only from one client at a time. Likewise, you need to quit the JDBC client, or at least disconnect from the database) before you run the toolkit.
The first thing is to download DBeaver from the link provided above. Installers are available for Linux, Mac and Windows, just choose the one that is best for you.
Before proceeding, please start the embedded database.

Setting up the connection to the embedded database

In order to use DBeaver on the embedded database, open it and click SQL in the list of database types at the left of the window that appears. Then click the `HSQL Embedded` icon.



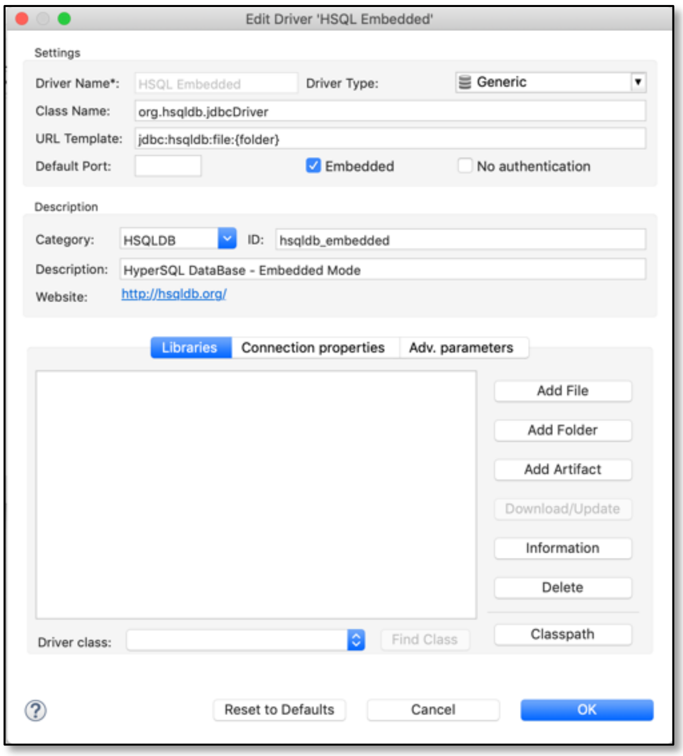
Click the `Next` button at the bottom of the window. The following window will appear


Connect to database
Generic JDBC Connection Settings
HSQLDB / HSQL Embedded connection settings
General
Driver properties
JDBC URL: jdbc:hsqldb:file:{folder}
Path:
Browse
Create
User name:
Password:
Save password locally
Advanced settings:
Connection details (name, type, ...)
Driver name: HSQLDB / HSQL Embedded
Edit Driver Settings
< Back
Next >
Cancel
Test Connection ...
Finish

The `JDBC URL` string is updated when you type anything in the fields below. Enter data as follows:

`Path`	Click the `Browse` button and locate the `db` directory inside `TKHome`. Click `Open` in the file selection dialog to accept and close it. When back to the window, please make sure to type `/glistk` at the end of the text in the `Path` box
`Database/schema`	`glistk`

```
    User name         glistk
    Password          glistk
```
Check that the `JDBC URL` is now
```
            jdbc:hsqldb:file:/<Your path to the db directory>/db/glistk
```
Now click the `Edit Driver Settings` button and delete any library that is shown in the list by selecting it and clicking the `Delete` button.



Edit Driver 'HSQL Embedded'

Settings

Driver Name*: HSQL Embedded   Driver Type: Generic

Class Name: org.hsqldb.jdbcDriver

URL Template: jdbc:hsqldb:file:{folder}

Default Port: [✓ Embedded]   No authentication

Description

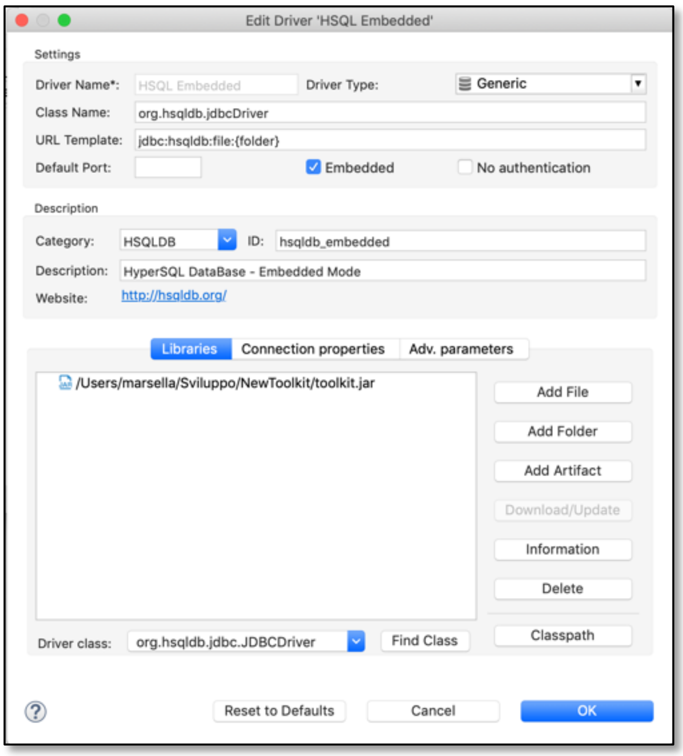
Category: HSQLDB   ID: hsqldb_embedded

Description: HyperSQL DataBase - Embedded Mode

Website: http://hsqldb.org/

Driver class:

When the list is empty, click `Add File` and locate the `toolkit.jar` in `TKHome`, again press `Open` in the file selection dialog to accept and close it. Back to the window above, you will see the `toolkit.jar` appear in the list, click the `Find Class` button. `org.hsqldb.jdbc.JDBCDriver` should appear in `Driver class`.


Edit Driver 'HSQL Embedded'

Settings

Driver Name*: HSQL Embedded | Driver Type: Generic
Class Name: org.hsqldb.jdbcDriver
URL Template: jdbc:hsqldb:file:{folder}
Default Port: | ☑ Embedded | ☐ No authentication

Description

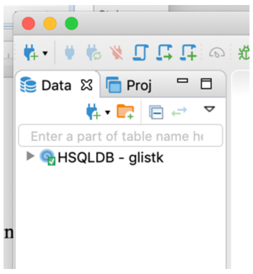
Category: HSQLDB | ID: hsqldb_embedded
Description: HyperSQL DataBase - Embedded Mode
Website: http://hsqldb.org/

Libraries | Connection properties | Adv. parameters

/Users/marsella/Sviluppo/NewToolkit/toolkit.jar

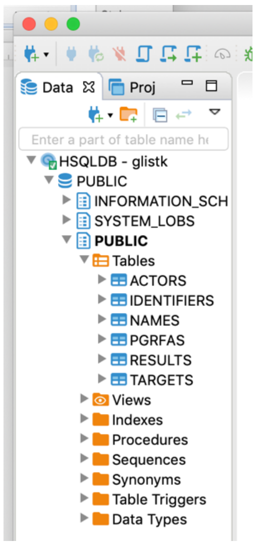
Driver class: org.hsqldb.jdbc.JDBCDriver

Please note that your path will be different from the one shown above, depending on where you put the `TKHome` directory. Click `OK` and then `Finish`. If asked whether you want to create the sample database, click `No`. The connection `HSQLDB - glistk` should appear in the list.



HSQLDB - glistk

To connect just click the little triangle and you should see the `PUBLIC` database. The TK tables are under `/PUBLIC/PUBLIC/TABLE`.



```
HSQLDB - glistk
  PUBLIC
    INFORMATION_SCH
    SYSTEM_LOBS
    PUBLIC
      Tables
        ACTORS
        IDENTIFIERS
        NAMES
        PGRFAS
        RESULTS
        TARGETS
      Views
      Indexes
      Procedures
      Sequences
      Synonyms
      Table Triggers
      Data Types
```

Double-click a table to see its contents. You can use SQL to insert or edit rows or import a CSV file. You can also edit rows manually using the buttons at the bottom of the window. Please refer to the DBeaver documentation for more details about its operation, in particular about importing and exporting data into/from tables.

Troubleshooting tips

The new version of the Toolkit is much easier to operate and troubleshoot. Some error cases are reported below. Please note that Java error messages can be very verbose. The error messages listed below are usually provided at the beginning of the Java error message. In case of doubt, please copy the full error message together with the configuration settings that appear at the beginning of the Toolkit output and send them to PGRFA-Treaty@fao.org.

Incorrect Java version

If you get an error message starting with;

```
com.mashape.unirest.http.exceptions.UnirestException: javax.net.ssl.SSLHandshake
Exception: sun.security.validator.ValidatorException: PKIX path building failed:
```

the likely cause is that your Java version is not correct. Open a terminal window, type

```
        java -version
```

and press Return. You will get a message like:

```
        java version "1.8.0_66"
        Java(TM) SE Runtime Environment (build 1.8.0_66-b09)
        Java HotSpot(TM) 64-Bit Server VM (build 25.66-b09, mixed mode)
```

If the first line reads `1.8.0_xxx` with `xxx` less than `101` (as in the example above, `66`) then you need to update your Java.

Open your browser a go to `https://www.java.com`. Click the "Free Java Download" red button at the center of the window, then click "Agree and Start Free Download" in the following window; a download will take place. Locate the installer you just downloaded and execute it. It will install the new Java version and ask you if you want to delete the old one, say "Yes". At the end of the installation process, restart your computer, then open a terminal window and check the Java version again with

```
        java -version
```

Verify that now the Java version is at least 101.

Lock file present

If you get this message

```
            Lock file 'lock.lck' exists. Either another instance of the Toolkit
is running or it has terminated in an error. Please make sure no other
instance of the Toolkit is running, or fix the error, before removing the lock
file and trying again
```

make sure to delete the file `lock.lck` in `TKHome` before trying to run the Toolkit again.

JDBC client connected to the embedded database running while executing the Toolkit

If you get an error like this

```
            org.sql2o.Sql2oException: Could not acquire a connection from
            DataSource - Database lock acquisition failure: lockFile:
            org.hsqldb.persist.LockFile@f42d63a2...
```

it is likely that you have a JDBC client connected to the embedded database. As explained earlier, Only one between the Toolkit and the JDBC client can be connected to the database at any given time. Please quit (or at least disconnect) the JDBC client before trying to run the Toolkit again..

Incorrect Username or Password for database access

If you get an error like this

```
            org.sql2o.Sql2oException: Could not acquire a connection from
            DataSource - invalid authorization specification
```

please check `db.username` and `db.password` in `config.txt`

Incorrect database name in JDBC URL

If you get an error like this

```
            org.sql2o.Sql2oException: Error preparing statement - user lacks
            privilege or object not found: PGRFAS in statement...
```

please check `db.url` in `config.txt`. The database name is likely wrong

Incorrect database version

If you get an error like this

```
          org.sql2o.Sql2oException: Database error: Unknown column 'pgrfa_id'
          in 'where clause'
```

please check that `db.version` is the correct one for the database. This error occurs when `db.version = 1` but the simplified schema is being used in `db.url`. If the reverse happens (`db.version = 2` with full database schema in `db.url`), the error would be

```
          org.sql2o.Sql2oException: Database error: Unknown column 'sample_id'
          in 'where clause'
```

Out of memory error

If you get an error like this

```
          java.lang.OutOfMemoryError
```

this means that there is not enough memory to run the Toolkit (or the embedded database, depending on what you were trying to do when you got the error). The memory requirements of the new version of the Toolkit are minimal, unless you specify an unreasonably large `db.query_limit` and use a severely under-configured computer. The Toolkit should be able to run with just 128MB RAM which is very little by today's standards. If you get this error, after having checked `db.query_limit`, please contact PGRFA-Treaty@fao.org for assistance.