NAME

Spkdb − Perl Binding for the Spk Database API

SYNOPSIS

```
use Spkdb;

See individual function descriptions.
```

EXPORT

None by default. All subroutines names may be imported.

ABSTRACT

Module Spkdb provides an Application Programming Interface (API) to the Spk Database. It accesses the database via the Perl modules DBI and DBD::Mysql. This module is simply a collection of Perl subroutines which correspond to the functions defined in the Spk Database API Specification.

DESCRIPTION

A short description of the usage of each of the subroutines follows.

connect — open a database connection

Establish a connection to the database, returning a handle which is a required argument to all other subroutines in this package:

$dbh = &Spkdb::connect($dbname, `$hostname`, `$username`, `$password`);

$dbname is a string containing the name of an existing database.

$hostname is a string containing the hostname where the database resides.

$username is a string containing the name of a valid user of Spkdb.

$password is a string containing the password of the given user.

Returns

```
success: a valid database handle
failure: undef
```

disconnect — close a database connection

Close a connection to the database, releasing resources:

```
&Spkdb::disconnect($dbh);
```

$dbh is the handle of an open database connection.

Returns undef.

new_job — submit a new job

Submit a new job to be compiled and run.

```
$job_id = &Spkdb::new_job($dbh,
                          $user_id,
                          $abstract,
                          $dataset_id,
                          $dataset_version,
                          $model_id,
                          $model_version,
                          $xml_source,
                          );
$dbh is the handle to an open database connection.
```

$user_id is the user_id number of an existing user.

$abstract is a string containing a brief description of the model.

$dataset_id is the dataset_id number of an existing dataset

$dataset_version is the version string of the dataset

$model_id is the model_id number of an existing model.

$model_version is the version string of the model.

$xml_source is the source string.

Returns:

```
success: automatically generated value of the job_id primary key
failure: 0
      $errstr contains an error messages string
      $err == $Spkdb::INSERT_FAILED
```

job_history —get event history for this job

Returns a reference to an array of rows from the history table. Each row is a reference to a hash table, with the field name as key and field value as value. Within the array, rows are sorted in order of occurence. Each row contains all fields of the history table.

```
  $array_row = $Spkdb::job_history($dbh, $job_id);
```

Returns

```
success:
  reference to an array of references to hash tables, each
  representing a row in the history table
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

get_job —retrieve a job for a user

Retrieve the job corresponding to a job_id

```
  $row = &Spkdb::get_job($dbh, $job_id);
```

$dbh is the handle to an open database connection

$job_id is the integer which uniquely identifies the job

Returns

```
success: reference to a hash table of column/value pairs
         0 if no datatset corresponds to the given job_id
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

job_status —return the current state of a job

Return the state_code, event_time, and end_code for a job. Note that the event_time is the time of the last job state transistion, such as the transition from ‘‘queued to compile’’ to ‘‘compiling’’.

```
  $row = &Spkdb::job_status($dbh, $job_id);
```

$dbh is the handle to an open database connection.

$job_id is the primary key of the job in question.

Returns

```
  success: reference to a hash table containing the following
    columns from the selected row of the job table:
        state_code
        event_time
        end_code
    In the hash table, the above field names are the keys and
    the column values the values.
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function fail
                == $Spkdb::EXECUTE_FAILED if execute function fails
```

user_jobs —get status for the most recent jobs of a user

Returns a reference to an array of rows. Each row is a reference to a hash table, with the field name as key and field value as value. Within the array, jobs are sorted in reverse order of job_id, hence the most recently submitted job appears first. The maximum number of jobs to return is the third argument.

The fields returned are job_id, abstract, state_code, start_time, event_time and end_code.

```
    $array_row = $Spkdb::user_jobs($dbh, $user_id, $maxnum);
```

Returns

```
  success:
    reference to an array of references to hash tables, each
    representing a subset of a row in the job table
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

set_state_code —set job state_code

Set the state_code of a specified job.

```
    $r = &Spkdb::set_state_code($dbh, $job_id, $state_code);
```

$dbh is the handle to an open database connection.

$job_id is the key to the job table

$state_code is the state_code to set.

Returns

```
  success: 1 if state_code of the job is set, 0 otherwise
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::UPDATE_FAILED
```

de_q2c —remove highest priority job from compile queue

Remove the highest priority job from the compiler queue, so that it can be compiled.

```
    $row = &Spkdb::de_q2c($dbh);
```

$dbh is the handle to an open database connection.

Returns

```
  success:
    reference to a hash for the row of highest priority,
    containing the following fields:
        job_id
        dataset_id
        dataset_version
        xml_source
    false if compiler queue is empty

  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

get_q2c_job —get a queued-to-compile job from the database

Remove the highest priority job from the compiler queue, so that it can be compiled.

```
    $row = &Spkdb::get_q2c_job($dbh, $job_id);
```

$dbh is the handle to an open database connection. `$job_id` is the key to a row in the job table.

Returns

```
  success:
    reference to a hash for the row of highest priority,
    containing the following fields:
        dataset_id
        dataset_version
        xml_source
    false if compiler queue is empty

  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

get_job_ids —get job_ids of all jobs with a given state_code

Get job_ids of all jobs as an array with a given state_code.

```
    @array_job_ids = &Spkdb::get_job_ids($dbh, $state_code);
```

$dbh is the handle to an open database connection.

$state_code is the given state_code of the jobs.

Returns

```
  success:
    an array of job_ids of all the jobs with the given state_code,
    0 if there is no job with the given state_code

  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

get_cmp_jobs —get all jobs currently being compiled

Returns a reference to an array of rows. Each row is a reference to a hash table, with the field name as key and field value as value. Within the array, jobs are sorted in order of job_id, hence the first job submitted appears first. Each row contains

job_id

dataset_id
dataset_version
xml_source

```
$array_row = $Spkdb::get_cmp_jobs($dbh);
```

Returns

```
success:
  reference to an array of references to hash tables, each
  representing a subset of a row in the job table;
  false if there are no jobs with state_code 'cmp'
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

en_q2r — add a compiled job to the run queue

Add a compiled job to the queue of jobs that are ready to run.

```
$r = &Spkdb::en_q2r($dbh, $job_id, $cpp_source);
```

$dbh is the handle to an open database connection.

$job_id is the key to the job table

$cpp_source is a string of c++ source code.

Returns

```
success: true
failure: false
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::UPDATE_FAILED
```

de_q2r — remove the highest priority job from the run queue

Remove the highest priority job from the ready to run queue, so that it can be run.

```
$row = &Spkdb::de_q2r($dbh);
```

$dbh is the handle to an open database connection.

Returns

```
success:
  reference to a hash for the row of highest priority,
  containing the following fields:
      job_id
      cpp_source
  false if ready to run queue is empty
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

get_q2r_job — get a queued-to-run job from the database Remove the highest priority job from the ready to run queue, so that it can be run.

```
$row = &Spkdb::get_q2r_job($dbh, $job_id);
```

$dbh is the handle to an open database connection. `$job_id` is the key to a row in the job table.

Returns

```
success:
  reference to a hash for the row of highest priority,
  containing the following fields:
      cpp_source
      checkpoint
  false if ready to run queue is empty
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

de_q2ar — remove the highest priority job from the aborting run queue

Remove the highest priority job from the aborting run queue, so that it can be aborted by run daemon.

```
$job_id = &Spkdb::de_q2ar($dbh);
```

$dbh is the handle to an open database connection.

Returns

```
success:
  job_id of highest priority aborting run job
  false if aborting run queue is empty
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

get_run_jobs — get all jobs with state_code 'run'

Returns a reference to an array of rows. Each row is a reference to a hash table, with the field name as key and field value as value. Within the array, jobs are sorted in order of job_id, hence the first job submitted appears first. Each row contains

job_id
cpp_source

```
$array_row = $Spkdb::get_run_jobs($dbh);
```

Returns

```
success:
  reference to an array of references to hash tables, each
  representing a subset of a row in the job table;
  false if there are no jobs with state_code 'run'
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

end_job — end a job, whether successful or not

End the job, whether successfully or not. Note, this does not actually terminate the processing of a job. Instead, it sets the state of the job to 'end', and stores the final report.

```
$r = &Spkdb::end_job($dbh, $job_id, $end, $report, $checkpoint);
```

$dbh is the handle to an open database connection.

$job_id is a unique job table identifier.

$end_code is a valid end_code, defined in the end table.

$report is a string containing the report.

$checkpoint is a string containing the checkpoint file

Returns

```
success: true
failure: false
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::INVALID_END
              == $SPKDB::UPDATE_FAILED
```

job_report — retrieve final report for a job

Retrieve the report string for a job in the 'end' state.

```
$report = &Spkdb::job_report($dbh, $job_id);
```

$dbh is the handle to an open database connection.

$job_id is the unique numeric identifier of a job;

Returns

```
success: a string containing the report
failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::NOT_ENDED if job not in 'end' state
                == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
                == $Spkdb::GET_FAILED if retieval failed
```

job_checkpoint — retrieve final checkpoint for a job

Retrieve the checkpoint string for a job in the 'end' state.

```
$checkpoint = &Spkdb::job_checkpoint($dbh, $job_id);
```

$dbh is the handle to an open database connection.

$job_id is the unique numeric identifier of a job;

Returns

```
success: a string containing the checkpoint
failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::NOT_ENDED if job not in 'end' state
                == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
                == $Spkdb::GET_FAILED if retieval failed
```

new_dataset — add a new dataset

Add a new dataset to a users collection of datasets stored in the database.

```
$dataset_id = &Spkdb::new_dataset($dbh, $user_id, $name, $abstract, $archive);
```

$dbh is the handle to an open database connection

$user_id is the unique identifier of a user

$name is the name of the dataset

$abstract is a short description of the dataset

$archive is a version archive, rcs format

Returns

```
success: the automatically generated value of the dataset_id
failure: 0
      $errstr contains an error message string
      $err == $Spkdb::DATASET_EXISTS     if user_id + name is not unique
           == $Spkdb::INSERT_FAILED    if the adding the new record failed
```

get_dataset —retrieve a dataset for a user

Retrieve the dataset corresponding to a name

```
$row = &Spkdb::get_dataset($dbh, $dataset_id);
```

$dbh is the handle to an open database connection

$dataset_id is the integer which uniquely identifies the dataset

Returns

```
success: reference to a hash table of column/value pairs
         0 if no datatset corresponds to the given dataset_id
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

update_dataset —update a dataset

Update the abstract and/or the archive of a dataset.

```
$r = &Spkdb::update_dataset($dbh, "dataset_id", $dataset_id, ...);
```

$dbh is a handle to an open database connection.

‘‘dataset_id’’, `$dataset_id` is a name value pair which uniquely identifies a dataset.

Other name value pairs specify the fields to be change. These may be:

‘‘abstract’’, `$abstract` ‘‘archive’’, `$archive`

These pairs may be in any order, but there must be at least one pair.

Returns

```
success: 1
failure: 0
  $Spkdb::errstr contains an error message string
  $Spkdb::err = $Spkdb::KEY_REQUIRED   if "dataset_id" not supplied
              = $Spkdb::INVALID_CHANGE if you have asked to change a
                          field that must never be changed
              = $Spkdb::UPDATE_FAILED  if failure for some other reason
```

user_datasets —get descriptions of all the datasets of a user

Returns a reference to an array of rows. Each row is a reference to a hash table, with the field name as key and field value as value. Within the array, datasets are sorted in order of name. The fields returned are dataset_id, name, and abstract.

```
$array_row = $Spkdb::user_datasets($dbh, $user_id);
```

Returns

```
  success:
    reference to an array of references to hash tables, each
    representing a row in the dataset table

  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

new_model —add a new model

Add a new model to a users collection of models, stored in the database.

```
    $model_id = &Spkdb::new_model($dbh, $user_id, $name, $abstract, $archive);
```

$dbh is the handle to an open database connection

$user_id is the unique identifi er of a user

$name is the name of the model

$abstract is a short description of the model

$archive is a version archive, rcs format

Returns

```
  success: the automatically generated value of the model_id
  failure: 0
        $errstr contains an error message string
        $err == $Spkdb::MODEL_EXISTS     if user_id + name is not unique
             == $Spkdb::INSERT_FAILED    if the adding the new record failed
```

get_model —retrieve a model for a user

Retrieve the model corresponding to a name

```
    $row = &Spkdb::get_model($dbh, $user_id, $name);
```

$dbh is the handle to an open database connection

$model_id is the integer which uniquely identifi es the model

Returns

```
  success: reference to a hash table of column/value pairs
           0 if no model corresponds to the model_id
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

update_model —update a model

Update the abstract and/or the archive of a model.

```
    $r = &Spkdb::update_model($dbh, "model_id", $model_id, ...);
```

$dbh is a handle to an open database connection.

‘‘model_id’’, `$model_id` is a name value pair which uniquely identifi es a model.

Other name value pairs specify the fi elds to be change. These may be:

‘‘abstract’’, `$abstract` ‘‘archive’’, `$archive`

These pairs may be in any order, but there must be at least one pair.

Returns

```
success: 1
failure: 0
  $Spkdb::errstr contains an error message string
  $Spkdb::err = $Spkdb::KEY_REQUIRED   if "model_id" not supplied
              = $Spkdb::INVALID_CHANGE if you have asked to change a
                          field that must never be changed
              = $Spkdb::UPDATE_FAILED  if failure for some other reason
```

user_models —get descriptions of all the models of a user

Returns a reference to an array of rows. Each row is a reference to a hash table, with the field name as key and field value as value. Within the array, models are sorted in order of name. The fields returned are model_id, name, and abstract.

```
$array_row = $Spkdb::user_models($dbh, $user_id);
```

Returns

```
success:
  reference to an array of references to hash tables, each
  representing a row in the model table
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

new_user —add a new user

Add a new Spk user to the database.

```
$user_id = &Spkdb::new_user($dbh,
                            "username", $username,
                            "password", $password,
                            ...);
```

$dbh is the handle to an open database connection

$username is a unique string that will be used to identify an Spk user.

$password is a string which will be used to authenticate this Spk user.

Other name/value pairs corresponding to column names defined for the user table.

Returns

```
success: the automatically generated value of the user_id
failure: 0
      $errstr contains an error message string
      $err == $Spkdb::COLUMN_REQUIRED if username or password not supplied
           == $Spkdb::USER_EXISTS     if the username is not unique
           == $Spkdb::INSERT_FAILED   if the adding the new record failed
```

update_user —update a user record

Update the row in user identified by `$user_id:`

```
$rv = &Spkdb::update_user($dbh, "user_id", $user_id, ...);
```

$dbh is a handle to an open database connection.

$user_id is value of the numerical primary key for an existing row.

Other name/value pairs specify columns to be changed.

Returns

```
success: 1
failure: 0
  $Spkdb::errstr contains an error message string
  $Spkdb::err = $Spkdb::KEY_REQUIRED  if "user_id" not suppled
              = $Spkdb::UPDATE_FAILED if failure for some other reason
```

get_user — retrieve a user record by username

Retrieve the row corresponding to a username.

```
    $row_hash_ref = &Spkdb::get_user($dbh, $username);
```

$dbh is the handle to an open database connection

$username is the value of username for an existing row.

Returns

```
  success: reference to a hash table of column/value pairs
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
                == $Spkdb::EXECUTE_FAILED if execute function failed
```

set_mail_notice — set end-job email notice request option Given a job_id, set end-job email notice request option for the job.

```
    $r = &Spkdb::set_mail_notice($dbh, $job_id, $email);
```

$dbh is the handle to an open database connection

$job_id is the key to a row in the job table

$email is true if end-job email notice is requested, false if otherwise

Returns

```
  success: true if end-job email notice request option is set, false if otherwise
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::UPDATE_FAILED
```

get_mail_notice — get end-job email notice request option Given a job_id, get end-job email notice request option for the job.

```
    $r = &Spkdb::get_mail_notice($dbh, $job_id);
```

$dbh is the handle to an open database connection

$job_id is the key to a row in the job table

Returns

```
  success: true if end-job email notice request option is set, false if otherwise
  failure: undef
    $Spkdb::errstr contains an error message string
    $Spkdb::err == $Spkdb::UPDATE_FAILED
```

email_for_job — get email address for a job

Given a job_id, retrieve the email address of the user.

```
    $email = &Spkdb::email_for_job($dbh, $job_id);
```

$dbh is the handle to an open database connection

$job_id is the key to a row in the job table

Returns

```
success: string containing an email address
failure: undef
  $Spkdb::errstr contains an error message string
  $Spkdb::err == $Spkdb::PREPARE_FAILED if prepare function failed
              == $Spkdb::EXECUTE_FAILED if execute function failed
```

SEE ALSO

For examples of the use of the subroutines in this module, see the test driver, driver.pl, in the t directory of this release.

AUTHOR

Alan Westhagen

COPYRIGHT AND LICENSE