Chado Controller Administrator Manual

V1.0.0
2012-04-02

Preface

This administrator manual targets system administrators and describes how to install, configure and manage the Chado Controller package.

Table of content

Install, update and uninstall

Get the installer package from http://www.gnpannot.org/content/chado-controller
Unzip it somewhere in en temporary directory on your server and follow the instruction from the file "INSTALL".

Users management

When you install the Chado Controller, you have 2 options for the user database. You can either store user acocunts in your Chado database or store them in a separate database. If you only run one instance of Chado, you may prefer the first solution. If you're running several instances, then you may prefer store user account data in a separate database common to all your Chado instances. This enables synchronizing user passwords across Chado instances.

When using a common user database, only user login and password are common: access rights and groups are Chado instance-specific. Also note that the login name used for the Chado Controller **must** correspond to a PostgreSQL user name as the Chado Controller uses PostgreSQL built-in functions to know who the acting user is. However, there is no way for the Chado Controller system to use PostgreSQL functions to authenticate a user, therefore the Chado Controller has to store passwords (encrypted hashes with pepper and salt) in a table in order to authenticate users from GBrowse for instance. This means that a user password may be different between a PostgreSQL access and a Chado Controller access. The provided user interface and administrator interface are supposed to handle password synchronization between those two instances of password. If a user PostgreSQL password is changed outside the Chado Controller interfaces, then the user will also have to change manually his/her Chado Controller account password if he/she want them to be identical.
Note: Chado Controller login and password should follow the same rules PosgreSQL accounts have (32 characters each).

Access management

Only Chado feature table is protected by the Chado Controller. Feature access can be managed at a feature level or a scaffold level or using a query to return related feature identifier. Access rights can be granted using groups or users. If the access right of a user to a given feature can come from both user and group access right, only the highest access right is kept. For instance, if the user U has read access to the feature F but U also belongs to group G and G has write access to F, then U has write access to F.
Note: all feature access rights are kept in the feature_access table. This table can quickly store a lot of entries and it is recommended to avoid assigning access rights to users be rather add users to groups and assign access right to those groups.

Annotators Management interface

The Annotators Management interface can be access through the web from the url <YOUR_GBROWSE_BASE_URL>/cgi-bin/annotators_management.

Once logged in, you will see the main menu with 4 items:

- Users Management
- Groups Management

- Features Access Management
- Check user accounts

Users Management interface

This interface allows you to add, modify or remove users. The “Role” and “Comments” fields are here for administrative purpose only (it helps you remember why you created an account) and are not used by the system.
Available user flags and descriptions:

- Disabled: when set, the user can not log in to the Chado Controller.
- Locked: this flag is automatically set when too many bad login attempts are made (invalid passwords) to protect the system against password brute-force attack from the GBrowse interface. The attacked user account is locked for a certain amount of time and the lock is automatically released after that delay and the user can login again. The maximum number of bad attempts a user can made is set by $ACCOUNT_MAX_FAILURE (default: 5) in the Access.pm perl library (somewhere in Bio/Graphics/Browser/) and the amount of time the account is locked is set in the same file by $ACCOUNT_LOCK_DELAY (default: 300sec).
- No Password: when set, the user doesn’t need to enter a password to login. Only a valid login name is required.
- Password Locked: when set, the user can not change his/her password from the GBrowse login interface.
- Password to Change: usually set for newly created users. When set, a message will be displayed on the login interface (once logged in) reminding the user he/she has to change his/her password.
- Write Access: when not set, a user can not modify a feature even if he/she has write access rights on that feature. It an easy way to have several users in a same (write) group and allow or deny write access to some specific users of that group.
- Gene Editor Access: if set, a link like “Edit with Artemis” can be displayed in GBrowse feature details popup. This can be used in GBrowse config file.

  ```
  [TOOLTIPS]
  default = sub {
  ...
     my $use_gene_editor      = $args->{gene_editor};
  ...
     if (($class =~ /manual_curation/) && $use_gene_editor) {
        $mess .= "<a href=\"$artemis\" class=\"menu\">Edit with artemis</a><br/>\n";
     }
  ...
  ```

- Admin: if set, the user is an administrator which means he/she has access to the Annotators Management interface and also have write access to any feature in the database (faster direct access to feature without access checking).

Groups Management

Group flags are the following:

- Disabled: access right from this group won’t be taken in account when computing user access rights even if the user belong to that group.
- Write Access: users of that group are allowed to modify features.
- Gene Editor: users of that group will see the link “Edit with Artemis” (see corresponding flag in Users Management section for details)

- Admin: users of that group have administrator access (see corresponding flag in Users Management section for details).

Setting user of group access rights

On both User Management and Group Management interfaces, you can set access right to features. Features are grouped by scaffold. For each scaffold, when you select:

- Keep current: no access right is changed for that scaffold;
- Use default: all access right specific to that scaffold for current user or group will be removed and the default behaviour will be used. It's a way to remove specific right from the database and make the feature_access table lighter.
- Forbiden: the user or group won't be able to see (and edit) the features of selected scaffold;
- Read: user or group can only view the features of the scaffold and not edit them;
- Write: the user or group can view and edit the features of the scaffold.

Features Access Management

This interface allows the administrator to set feature access rights either using default feature selection (ie. by scaffolds) or using a custom query that returns the related feature_id. The default query that selects all the features of scaffolds ('contig') is the following:
SELECT f.feature_id FROM feature_data f, cvterm cvt, featureloc fl WHERE (f.feature_id = fl.srcfeature_id) AND (f.type_id = cvt.cvterm_id) AND (cvt.name = 'contig')

If you would like to select all the features of a database, use the following query:
SELECT f.feature_id FROM feature_data f

If you would like to change the access right of features of a specific track which has the accession "<TRACK_NAME>":
SELECT f.feature_id FROM feature_data f JOIN feature_dbxref fd ON fd.feature_id = f.feature_id JOIN dbxref d ON fd.dbxref_id = d.dbxref_id WHERE d.accession = '<TRACK_NAME>'

If you would like to change the access rights of features of a given type_id <TYPE_ID>:
SELECT f.feature_id FROM feature_data f WHERE f.type_id = <TYPE_ID>

If you have a list of feature ID that you'd like to protect, just enter the ID separated by comas:
123, 456, 987, 1234

In any of these cases, just enter the subquery in the text area "Features selection query:" and hit the button "Update" below the tex area. Once some example of selected features are displayed, you can either just edit a single feature of the list using its "Edit" button or **all** the selected features (even the one that are not displayed because of the "LIMIT" clause in the SQL query) using the "Edit rights of selected features" button.

The "Auto-set access rights for new features" button can be use to assign to new features (with no access right set) the same access rights the source feature they are located on has. For instance, a new feature added to an existing scaffold will get the same access rights than the scaffold. The Chado Controller uses the table featureloc to find what is the "source" feature of new features.

Finally “Set public read access rights for non-contig features” button can be used to allow everybody to see features that are not directly linked to a scaffold (contig). For instance, a feature corresponding to a BLAST match won’t be localized on a scaffold and using “Auto-set access rights for new features” won’t have any effect but “Set public read access rights for non-contig features” will.

Annotation Inspector

Unfortunately, no web interface has been developed yet to handle Annotation Inspector tasks. However, you can use the Annotation Inspector using PostgreSQL queries.

Main validation procedure

The Annotation Inspector can be called to inspect a specific transaction group “<transaction_group>” using the procedure “validate_annotations”.
Example:
SELECT * FROM validate_annotations(<transaction_group>, FALSE);

The second parameter called “force_validation“ is a boolean value (set to FALSE in the example above). It is used to select the way the procedure should be run. When set to FALSE, The “validate_annotations” procedure will just generate a report with errors and warning met. When set to TRUE, the “validate_annotations” will still reports error but will also make some changes in the database by adding some properties or CV terms to annotated features reporting the encountered problems. For instance, if a stop codon is met inside a coding sequence, the feature CV term “stop_in_frame” (CV “structural_completeness”) will be added to the polypeptide. When encountered problems are fixed, such added properties and CV terms are automatically removed.

Other validation procedures

“validate_annotations” procedure calls all the procedures that are enabled in the “annotation_inspector_procedures” table (added to Chado schema) in their priority order (negative values are lower priorities, positive values are higher priorities). Each of these procedures can be called the same way “validate_annotations” is called. They return a status string reporting if errors were found.
Each of these functions can also be enabled or disabled at will by changing the corresponding “enabled” boolean value of the “annotation_inspector_procedures” table. For instance, to disable the “check_gene_structure” procedure, execute the following query:
UPDATE annotation_inspector_procedures SET enabled = **FALSE** WHERE name = ‘check_gene_structure’;

New validation procedure can also be added: please refer to the Chado Controller Technical Documentation for details on how to create such procedures.

Transaction groups

You can find out what transaction_group you are looking for by querying the table that have been modified (feature, featureprop, featureloc, etc.) using a query similar to this one:

SELECT transaction_group FROM feature_audit WHERE transaction_user = ‘user_name’ AND transaction_date > ‘2012-09-01 17:00:00’ AND transaction_date < ‘2012-09-01 18:00:00’ ORDER BY transaction_date DESC;

SELECT transaction_group FROM featureprop_audit WHERE transaction_user = ‘other_user_name’ AND transaction_date > ‘2012-09-01’ AND transaction_date < ‘2012-09-02’ ORDER BY transaction_date ASC LIMIT 20;

If you would like to check all the features of the manual annotation track without taking in account any transaction group, you can use the value “0” as transaction group. This will tell the Annotation Inspector to process all the features of the manual annotation track.
Note: if you have a lot of data, checking the whole manual annotation track may take a lot of time (several hours and maybe even days!).

Troubleshooting

Access Restriction

Compatibility

In some rare cases, you may use software that requires the feature table to exist. Since the Chado Controller Access Restriction module replace this table with a view, PostgreSQL queries like “COPY” will fail. In these cases, you can either tell your program to use the feature_data table instead or put the Access Restriction in compatibility mode using the script provided in the installation package:
cc_compatibility.pl -h localhost -p 5432 -d chado -U postgres -W your_password -on

Important: don’t forget to switch compatibility mode off once you are done otherwise, other users will experiment access issues!
cc_compatibility.pl -h localhost -p 5432 -d chado -U postgres -W your_password -off

For more details on how to use cc_compatibility.pl, just use the command line “cc_compatibility.pl –help”.

Account creation/access to tables or procedure issues

The administration interface has the menu item “Check user accounts”. This menu item performs several checks on the database and can fix many issues. You can use it any time you want to check if it can fix your issues.

Password synchronization

As described in the “user management” section, user passwords have to be synchronized between Chado Controller access and PostgreSQL. If they are not synchronized, you will have to synchronize them manually either by changing the Chado Controller password form the administrator interface or by changing the PostgreSQL user account password as annotators may be confused by having two different passwords.

Annotation Inspector

If an annotation validation procedure raises some (recurrent) issues, it can be disabled by changing is “enabled” value in the table annotation_inspector_procedures. For instance, if you would like to disable the “check_introns” procedure, do:

UPDATE annotation_inspector_procedures SET enabled = **FALSE** WHERE name = ‘check_introns’;

To reactivate it:
UPDATE annotation_inspector_procedures SET enabled = **TRUE** WHERE name = ‘check_introns’;

No solution found

If you couldn’t find any solution to your problem, you can use the e-mail address provided in the “Contact” section below.

Contacts

valentin.guignon@cirad.fr
stephanie.sidibe-bocs@cirad.fr