
Tripal Apollo Documentation

Release 1.0

Bradford Condon

Jan 08, 2019

1	Introduction & Background	1
1.1	What is Apollo?	1
1.2	What does Tripal Apollo do?	1
2	Installation and Setup	5
2.1	Admin setup	5
2.2	Permissions	8
3	Apollo Configuration	11
3.1	Apollo 2	11
3.2	Apollo 1	11
4	Testing, Development, and Contributing	13
4.1	Testing	13
4.2	Contributing	14
5	Permissions Guide	15
5.1	Anonymous vs registered requests	15
5.2	Protecting your site with anonymous submissions	15
6	Customizing This Module	19
6.1	Mail Messages	19
6.2	Forms	19

Introduction & Background

The Tripal Apollo module seeks to bridge Tripal and Apollo user and data management for both Apollo 1 and 2.

1.1 What is Apollo?

Apollo is a plugin for the Genome Viewer JBrowse (<http://jbrowse.org/>)

Apollo provides:

- A user interface for editing gene annotation tracks
- GO term support
- Revision history

To learn more, visit:

<http://genomearchitect.github.io/>

1.2 What does Tripal Apollo do?

1.2.1 User account requests

Users visit `/apollo-registration` and select which organisms they would like access to.

Apollo Registration

Complete the form below and click 'Submit' to register for an Apollo account. Only registered users can view, create or change annotations.

Full Name *

Email Address *

The email address you will use to log in to the Apollo server.

Organism *

HQ_test_organism Ava Hansen
HQ_test_organism Mae Weimann V
HQ_test_organism Mr. Ludwig Hyatt PhD
N/A N/A

The organisms you would like access to.

Institution *

Genes or gene families that you intend to annotate *

Submit

An email is sent to the user and the site admin email notifying them of the request.

Home » Administration » Tripal

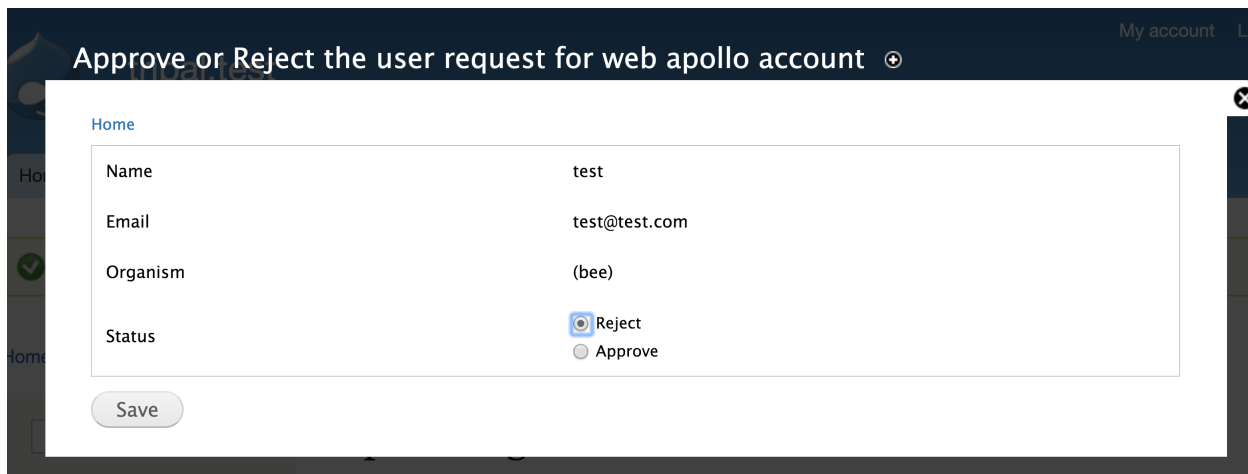
Apollo User Administration

NAME	EMAIL	ORGANISM	STATUS	CREATED	
Bradford Condon	waffles@waffle.com	kjljkjlj jkljkjlj	Pending	Oct 11 2018 03:31:06 PM	Edit
waffle	waffle@waffle.com	fakus species	Pending	Oct 11 2018 03:16:49 PM	Edit
waffle	waffle@waffle.com	democus demo species	Pending	Oct 11 2018 03:16:49 PM	Edit
waffle	waffle@waffle.com	hkjhkhjh hkjh	Pending	Oct 11 2018 03:16:49 PM	Edit
waffle	waffle@waffle.com	jlkjljk lkjj	Pending	Oct 11 2018 03:16:49 PM	Edit
waffle tst person	test@waffles.com	HQ_test_organism Mr. Ludwig Hyatt PhD	Pending	Oct 09 2018 05:18:09 PM	Edit
waffle tst person	test@waffles.com	HQ_test_organism Mae Weimann V	Pending	Oct 09 2018 05:18:09 PM	Edit
waffle tst person	test@waffles.com	HQ_test_organism Ava Hansen	Pending	Oct 09 2018 05:18:09 PM	Edit

1.2.2 Approving/denying requests

Registration requests appear at `admin/tripal/apollo/requests`.

Each row is for a single user - organism request pairing, so a single form submission may consist of several rows. The admin can click the **Approve/Deny** button to view the request, which will list the user name, email, organism. To approve or reject the request, check the appropriate box and click **Save**.



The screenshot shows a web interface for managing user requests. The main heading is "Approve or Reject the user request for web apollo account" with a plus icon. Below this is a form with a "Home" link. The form contains the following fields:

Name	test
Email	test@test.com
Organism	(bee)
Status	<input checked="" type="radio"/> Reject <input type="radio"/> Approve

At the bottom of the form is a "Save" button.

Installation and Setup

2.1 Admin setup

Download the module using git (`git clone https://github.com/NAL-i5K/tripal_apollo.git`). Enable the module with drush: `drush pm-enable tripal_apollo`. Instructions are the same for both Tripal 2 and Tripal 3 sites.

User passwords are generated using the `/usr/share/dict/words` file. If this file doesn't exist on your server, please create it and populate with words you would like your user passwords generated with (one word per line).

Warning: If your `/usr/share/dict/words` file does not exist, or does not contain new-line separated words, this module will crash on form submission.

2.1.1 Site-wide settings

Site-wide settings can be set at `/admin/tripal_apollo`.

Tripal Apollo Settings

[Home](#) » [Administration](#) » [Tripal](#)

Python Path

Path to Python on the server.

Chado Base Table

The Chado base table associated with Apollo records. This can be Organism, Analysis, and Project.

☒ Encrypt Passwords

Enable/disable encryption. Please see <https://www.drupal.org/project/encrypt> for more information.

Table 1: Site-Wide Settings

Field	Description
Python Path	Full path to python executable on the server. Only necessary for Apollo 1.
Chado Base Table	Determine what content will link to Apollo. Set to organism by default. Please see warning below regarding changing the base table.
Encrypt Passwords	Should user and admin passwords be encrypted when stored in the database? If your site is hacked, this will make user passwords more secure.

Note: Python path is only necessary for Apollo 1.

Warning: Important notice!!! Switching base tables will **wipe all information linking instances to records and records to users**. Be very mindful of changing this setting!

Password encryption

We encourage enabling password encryption. However, disabling encryption is provided in case of issues setting up the encryption module. <https://www.drupal.org/project/encrypt>

Note: You may get a the following warning:

```
Deprecated function: Function mdecrypt_generic_init() is deprecated in _encrypt_
↳ encryption_methods_mcrypt_aes_cbc() (line 121 of /var/www/html/sites/all/modules/
↳ encrypt/plugins/encryption_methods/mcrypt_aes_cbc.inc)
```

This warning is due to mcrypt being marked for deprecation in PHP 7.1. In fact, we expect this to cause a more serious error in PHP 7.2. This is a [known issue for the Drupal Encrypt module](#), and if the warning is critical for you, there is

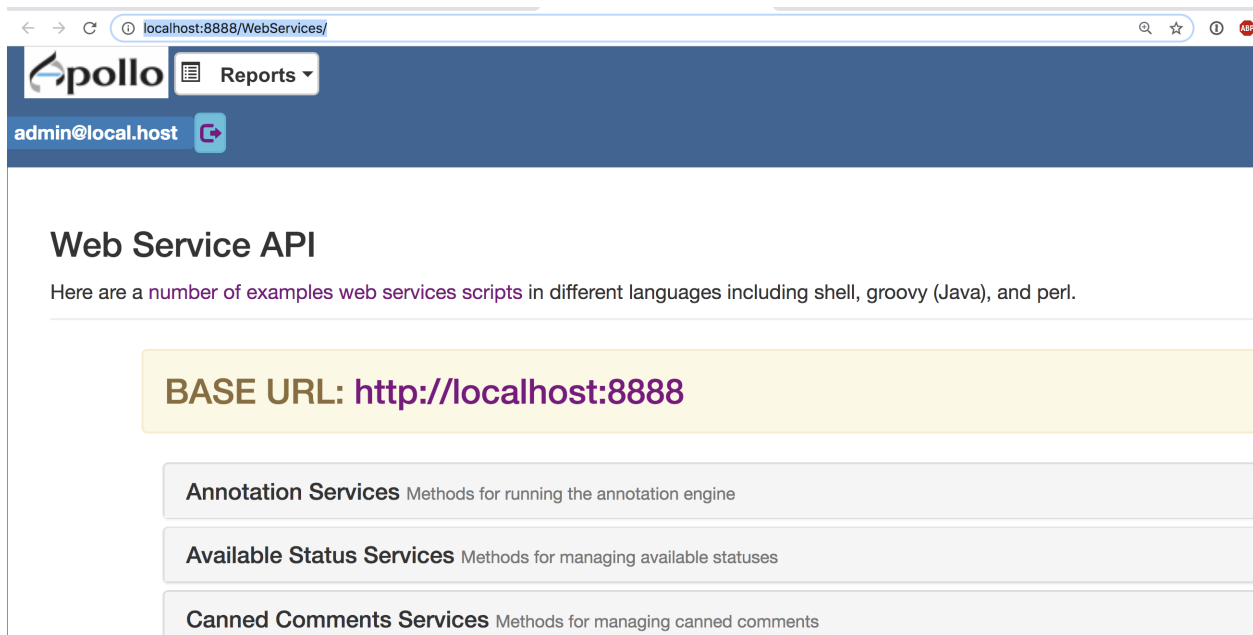
a patch available for the encrypt module here: <https://www.drupal.org/project/encrypt/issues/2983555>

Adding an Apollo Instance to Tripal

First, you must tell the module about your Apollo server. To do so, go to **Content → add Content → Apollo Instance**.

If your server is Apollo 1, you will need to provide the db name, db admin name and password. Apollo 2 will instead require the admin username and password: the db username is not required.

The URL should be the full apollo server URL without a trailing slash, for example, `http://localhost:8888`. The correct URL is listed in your web services API on your apollo server:



Select all of the organisms you would like linked to this Apollo instance. Note that Apollo 1 mappings for multiple organisms makes several assumptions: see *Apollo Configuration*.

GENERAL

Human-readable Name *

apollo 2

URL of Apollo Server *

http://localhost:8888

Apollo Instance Type *

Apollo 2

Database Name

The name of the database on the Chado instance. Only required for Apollo 1.

Admin Username *

admin

For Apollo 1 this is the DATABASE admin username. For Apollo 2, it is the Apollo admin user account name.

Database admin password *

pass

For Apollo 1 this is the DATABASE admin username. For Apollo 2, it is the Apollo admin user account name.

Associated Records

apis mellifera

Select which records are associated with this instance. A record can only be associated with one instance: setting this instance will overwrite others.

If your instance is successfully linked, the “Users” field will display the number of non-admin users on your instance.

2.2 Permissions

This module defines the following permissions:

- administer tripal apollo: Administer the module itself. This permission is for site admins.
- administer apollo users. Allows admins to approve/deny Apollo access requests. This permission is for site admins and/or community leaders.
- access apollo: allows users to make Apollo registration requests. You can give this permission to anonymous users, allowing users to register for apollo accounts without a Drupal account.

To learn more about setting up permissions and roles, please see https://www.drupal.org/docs/_static/img/7/managing-users/user-roles

To learn more about the consequences of allowing anonymous users to make Apollo requests, see here: [Permissions Guide](#).

2.2.1 Chado specific permissions

If you have the `tripal_hq` and `tripal_hq_permissions` modules enabled, you can use Chado-specific permissions! This means you can have a user role that can, for example, approve Apollo requests for a subset of organisms **only**. Simply configure HQ permissions for curators based on chado **organisms**.

Please see the `tripal_hq` module for more information: https://github.com/statonlab/tripal_hq.

2.2.2 References

Dunn NA, Munoz-Torres MC, Unni D, Yao E, Rasche E, Bretaudeau A, Holmes IH, Elisk CG; Lewis SE (2017).
GMOD/Apollo: Apollo2.0.6(JB#29795a1bbb)

CHAPTER 3

Apollo Configuration

The Tripal Apollo module makes several assumptions about your Apollo instances in order to connect. If these assumptions don't hold true for your configuration, please let us know on the issue board at https://github.com/NAL-i5K/tripal_apollo/issues and we'll try to help.

3.1 Apollo 2

3.1.1 Expected Group Names

Tripal Apollo does not configure your user groups for you. Tripal Apollo assumes that for each organism, with a particular genus and species, you have three groups: `genus_species_ADMIN`, `genus_species_USER`, and `genus_species_WRITE`. This is typically done when adding the organism to Apollo. For this module to function to correctly, the organism you create on your Tripal site must have genus and species fields which match the existing groups on your Apollo instance.

The organism *Saccharomyces cerevisiae* should therefore have groups configured `saccharomyces_cerevisiae_WRITE`, `saccharomyces_cerevisiae_USER`, and `saccharomyces_cerevisiae_ADMIN`. Approving a user request for this organism will add them to the first two.

3.2 Apollo 1

3.2.1 Server Setup

Apollo 1 does not support a REST API. Your Apollo 1 server's database must therefore be setup to accept remote connections by editing `pg_hba.conf`.

Naming Conventions

Tripal Apollo provides limited support for Apollo 1, via a python script which connects directly to the instance database. Note this is why the **Database Name** field is only required for an Apollo 1 instance. A key discrepancy between Apollo 1 and 2 is that each organism is its own server for Apollo 1. Rather than require admins to create Apollo instances for each organism separately, we assume a uniform URL for Apollo 1 organisms attached to the same instance:

`http://url/[first three letters of genus][first three letters of species].`

If your URL is set as `http://localhost:8000`, and you select organisms *Acer saccharum* & *Homo sapiens*, for example, this Apollo 1 instance will connect to the following two apollo servers:

- `http://localhost:8000/acesac`
- `http://localhost:8000/homsap`

Testing, Development, and Contributing

4.1 Testing

Tripal Apollo uses Tripal Test Suite to make configuring PHPUnit to work with Drupal and Tripal easy. Tripal Test Suite documentation is available here: <https://tripaltestsuite.readthedocs.io/en/latest/>

4.1.1 Creating a Development Environment

The travis CI environment uses the Docker compose file in this repository to launch a Tripal site and Apollo site. You can simply use this configuration locally!

An example setup:

```
# extract the example dataset
tar -xvf example_data/yeast.tar.gz -C example_data/
composer install
docker-compose up -d
## Set the APOLLO_URL variable.
APOLLO_URL=http://localhost:8888
export APOLLO_URL
/bin/bash setup/set_travis_apollo.sh
```

If you only need an Apollo container, it can be run via `docker run`:

```
# extract the example dataset
tar -xvf example_data/yeast.tar.gz -C example_data/
# run an Apollo container
docker run -it -v ${PWD}/example_data:/data -p 8888:8080 quay.io/gmod/docker-
→apollo:2.1.0

## Set the APOLLO_URL variable.
APOLLO_URL=http://localhost:8888
export APOLLO_URL
```

(continues on next page)

(continued from previous page)

```
#run the setup script, which will create the organism and groups in the Apollo_
↳instance.
/bin/bash setup/set_travis_apollo.sh
```

Note: The Apollo credentials for this container are:

- username: `admin@local.host`
 - password: `password`
-

4.1.2 Setting up Tripal Test Suite

Prior to running test suite, you must run `composer install` and copy `tests/example.env` to `tests/.env`. Note we define an extra variable in `tests/example.env`: `APOLLO_URL=http://localhost:8888`. This **MUST** include `http://` and it must point at your Apollo instance for tests to work.

See <https://tripaltestsuite.readthedocs.io/en/latest/environment.html?highlight=.env> for general information on setting up Test Suite.

4.2 Contributing

Tripal Apollo is open source and distributed via the GPL 3 license. If you have questions, feature requests, or a desire to contribute, please post to the github issues board here: https://github.com/NAL-i5K/tripal_apollo/issues

5.1 Anonymous vs registered requests

The **access apollo** permission determines if visitors can submit Apollo requests. You can choose whether or not to give this permission to anonymous users. If you do, users won't need an account to request Apollo access. This can be convenient if creating an account on your Drupal site offers little else to end users; in such cases it's silly to create two accounts (the Apollo account and the Drupal account).

Our general recommendation is to **require an account** for Apollo registration. Doing this will greatly simplify your anti-spam efforts. You can simply enable Drupal anti-spam modules, which will automatically provide barriers at the account registration step. Allowing anonymous Apollo registration form submissions means you will have to extend these anti-spam modules for each form you need to protect.

5.2 Protecting your site with anonymous submissions

If you do allow anonymous requests, how do you enable anti-spam measures on the form?

5.2.1 HoneyPot Module

Honeypot deters spam bots from completing forms on your site!

Find out more here: <https://www.drupal.org/project/honeypot>

To add Honeypot to the registration form, enable the Honeypot module, and simply add `honeypot_add_form_protection($form, $form_state, array('honeypot', 'time_restriction'))`; to the form. You can use `hook_form_alter` to do this from within your site's custom themeing module, without having to maintain a separate fork of Tripal Apollo.

```
function my_custom_module_name_form_alter(&$form, &$form_state, $form_id){  
  if ($form_id == "tripal_apollo_registration_form") {  
    honeypot_add_form_protection($form, $form_state, array('honeypot', 'time_  
restriction'));
```

(continues on next page)

(continued from previous page)

```
}
}
```

5.2.2 Captcha Module

There are multiple captcha modules available for Drupal. The below instructions are for the Captcha module.

To add a captcha to our form:

- Enable the drupal captcha module: <https://www.drupal.org/project/captcha>
- go to `admin/config/people/captcha`
- Add the registration form with this ID: `tripal_apollo_registration_form`. Use the default challenge type.
- add the **Skip CAPTCHA** permission to your registered users, so they don't have to fill out the captcha if they are registered.

administration links to forms below.

Default challenge type

Math (from module captcha) ▾

Select the default challenge type for CAPTCHAs. This can be overridden for each form if desired.

FORM_ID	CHALLENGE TYPE	OPERATIONS
tripal_apollo_registration_form	Default challenge type ▾	delete
comment_node_chado_analysis_blast_form	Default challenge type ▾	
comment_node_chado_analysis_kegg_form	Default challenge type ▾	
comment_node_apollo_instance_form	Default challenge type ▾	
comment_node_article_form	Default challenge type ▾	
comment_node_page_form	Default challenge type ▾	
user_register_form	Default challenge type ▾	
user_pass	Default challenge type ▾	
user_login	Default challenge type ▾	
user_login_block	Default challenge type ▾	
<input type="text"/>	- No challenge - ▾	

A CAPTCHA now appears on our registration page!

Apollo Registration

Complete the form below and click 'Submit' to register for an Apollo account. Only registered users can view, create or change annotations.

Full Name *

Please provide your first and last name separated by a space.

Email Address *

The email address you will use to log in to the Apollo server.

Organism Content *

The organism(s) you would like access to.

Institution *

Genes or gene families that you intend to annotate *

CAPTCHA

This question is for testing whether or not you are a human visitor and to prevent automated spam submissions.

Math question *

2 + 0 =

Solve this simple math problem and enter the result. E.g. for 1+3, enter 4.

Submit

By default, the Captcha is also added to the user registration form, so even if you don't allow anonymous users to request Apollo access, you may be interested in this module.

CHAPTER 6

Customizing This Module

6.1 Mail Messages

6.2 Forms