Tripal BLAST Documentation

University of Saskatchewan with the Legume Federation

Apr 21, 2023

Contents:

CHAPTER 1

User's Guide

This module provides a basic interface to allow your users to utilize your server's NCBI BLAST+.

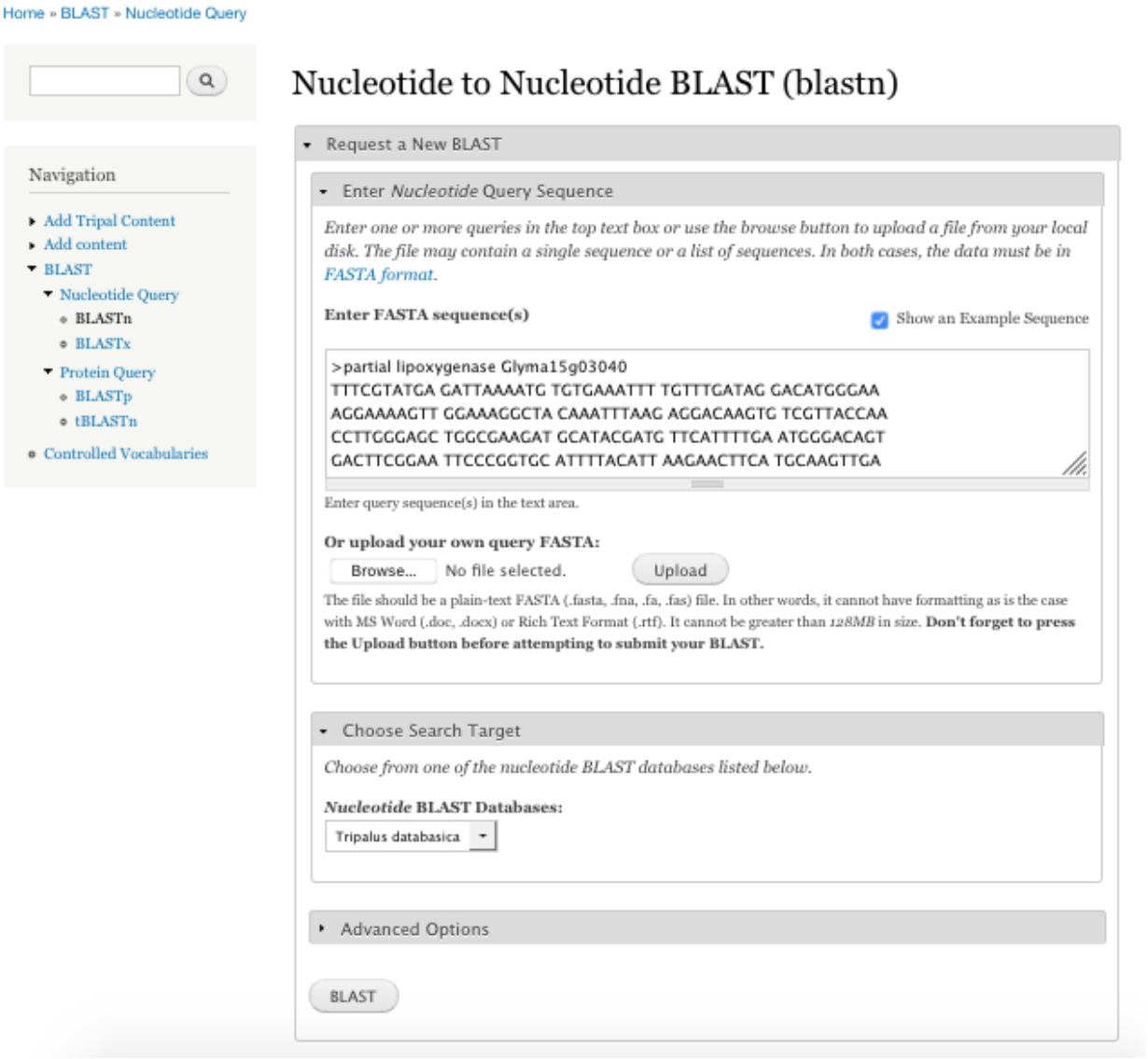
Specifically it provides blast program-specific forms (blastn, blastp, tblastn, blastx are supported). In the future, there will be a single form where you will be able to select either a nucleotide or a protein database to BLAST against regardless of the type of query and it will decide which BLAST program to use based on the combination of query/database type (ie: if you selected a protein database on the nucleotide BLAST form then blastx would be used).

BLAST submissions result in the creation of Tripal jobs which then need to run from the command-line. This ensures that long running BLASTs will not cause page time-outs but does add some management overhead and might result in longer waits for users depending on how often you have cron set to run Tripal jobs. You can alternatively use the Tripal Jobs Daemon to automate running of Tripal Jobs reducing user wait time and your own workload.

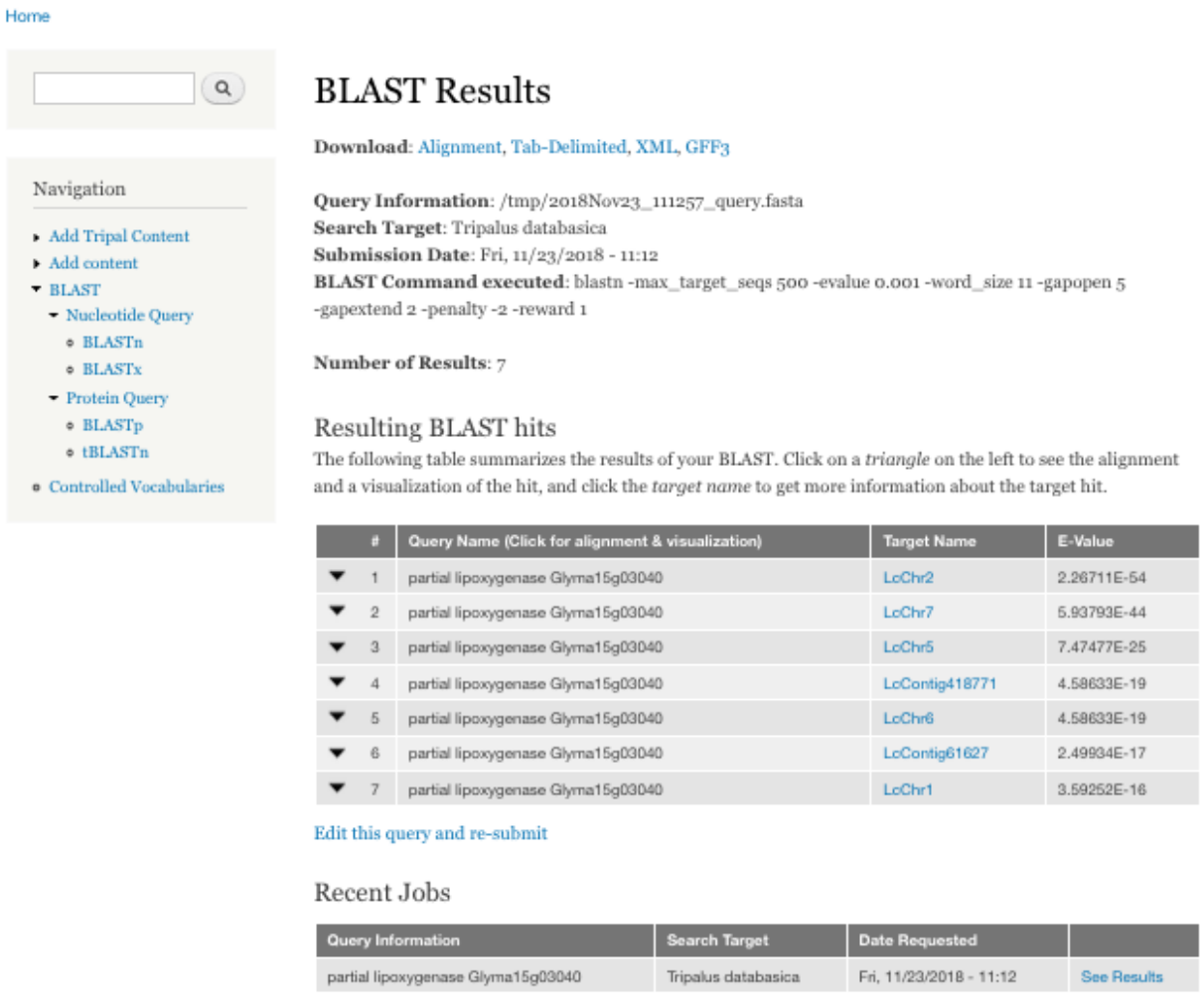
The BLAST results page is an expandable summary table with each hit being listed as a row in the table with query/hit/e-value information. The row can then be expanded to include additional information including the alignment. Download formats are allow users to download these results in the familiar tabular, GFF3 or HTML NCBI formats.

1.1 Highlighted Functionality

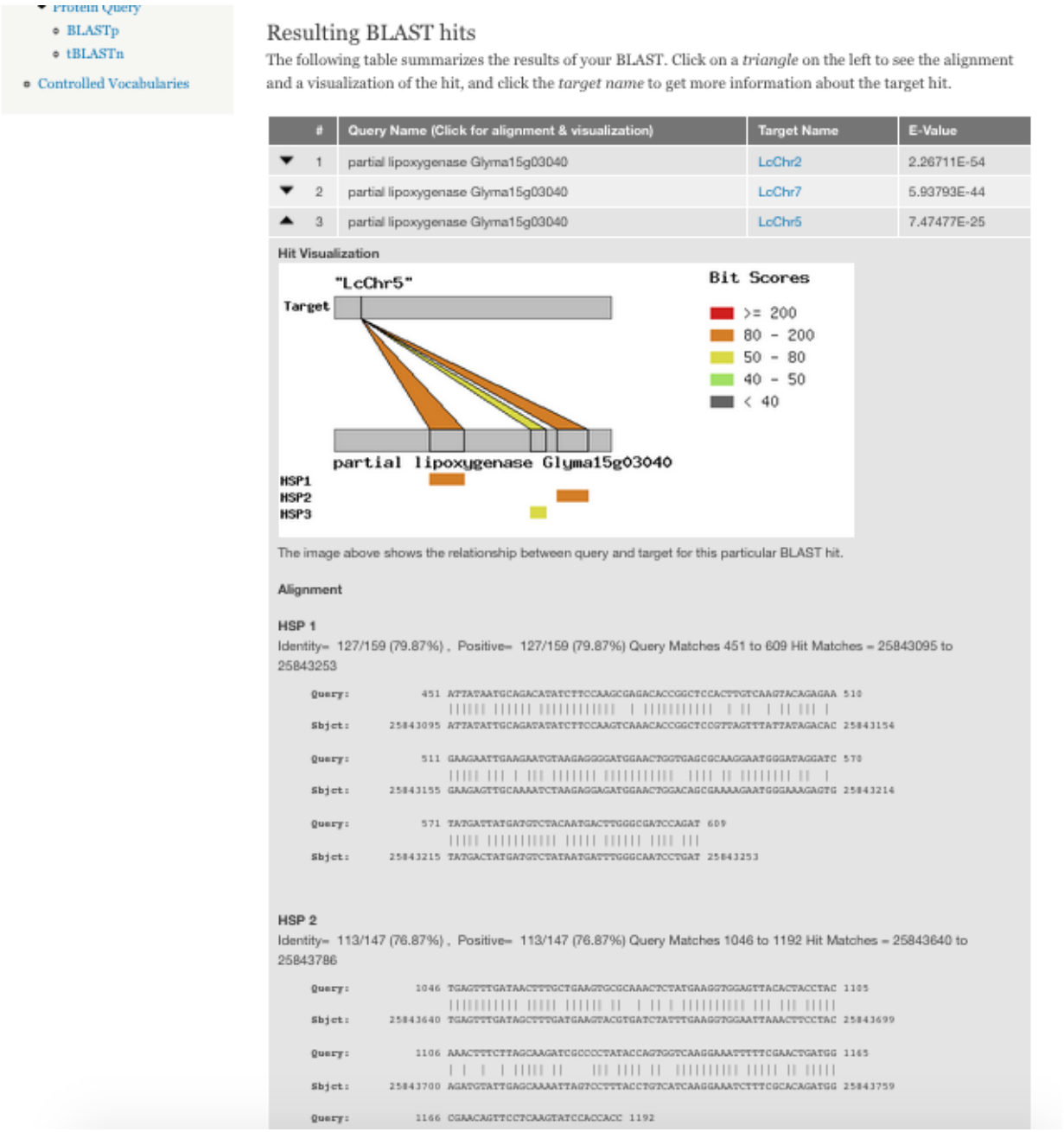
- Supports blastn, blastp, tblastn, and blastx with separate forms depending upon the query type.
- Simple interface allowing users to paste or upload a query sequence and then select from available databases. Additionally, a FASTA file can be uploaded for use as a database to BLAST against.



- Tabular Results listing with alignment information and multiple download formats (HTML, TSV, GFF3, XML) available.



- Completely integrated with Tripal Jobs providing administrators with a way to track BLAST jobs and ensuring long running BLASTs will not cause page time-outs
- BLAST databases are made available to the module by creating Drupal Pages describing them. This allows administrators to use the Drupal Field API to add any information they want to these pages and to control which databases are available to a given user based on native Drupal permissions.
- BLAST database records can be linked to an external source with more information (ie: NCBI) per BLAST database.
- Per Query result diagrams visualizing the HSPs to help users better evaluate hit quality.



- Optional Whole Genome diagrams visualizing the distribution of hits which are configurable per Blast Database.

1.2 Installation

1.2.1 QuickStart

1. Install NCBI BLAST+ on your server (Tested with 2.2.26+). Please use the official NCBI installation documentation for your server.
2. Install this module as you would any Drupal module (ie: download, unpack in `sites/all/modules` and enable through `http://[your site]/admin/modules`)

3. Create "Blast Database" nodes for each dataset you want to make available for your users to BLAST against. BLAST databases should first be created using the command-line `makeblastdb` program with the `-parse_seqids` flag.
4. It's recommended that you also install the Tripal Job Daemon to manage BLAST jobs and ensure they are run soon after being submitted by the user. Without this additional module, administrators will have to execute the tripal jobs either manually or through use of cron jobs.

1.2.2 Install NCBI BLAST+

See NCBI's Standalone BLAST Setup for Unix for extended instructions.

1.2.3 Install Tripal BLAST

This module is available as a project on Drupal.org. As such, the preferred method of installation is using Drush:

```
cd /var/www/html
drush pm-download tripal_blast libraries
```

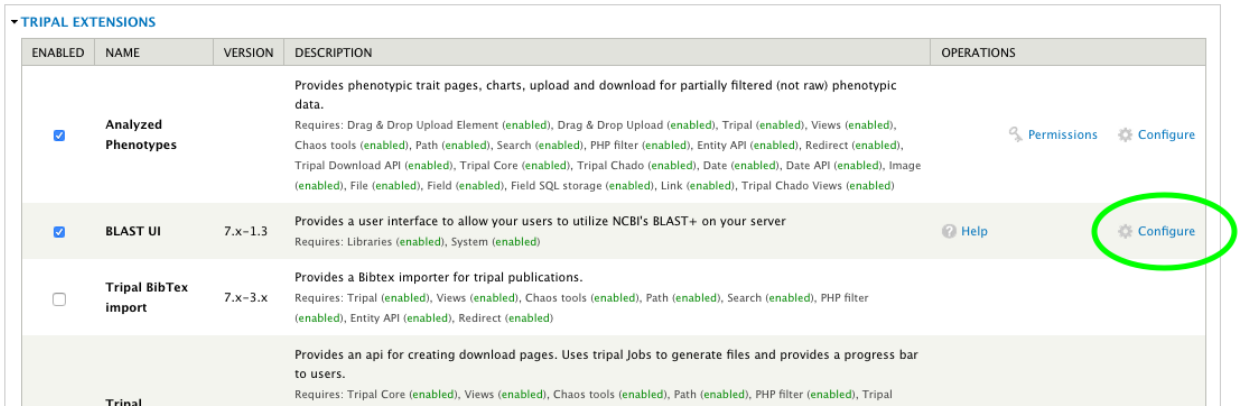
The above command downloads the module into the expected directory (e.g. `/var/www/html/sites/all/modules/tripal_blast`). Next we need to install the module:

```
drush pm-enable blast_ui
```

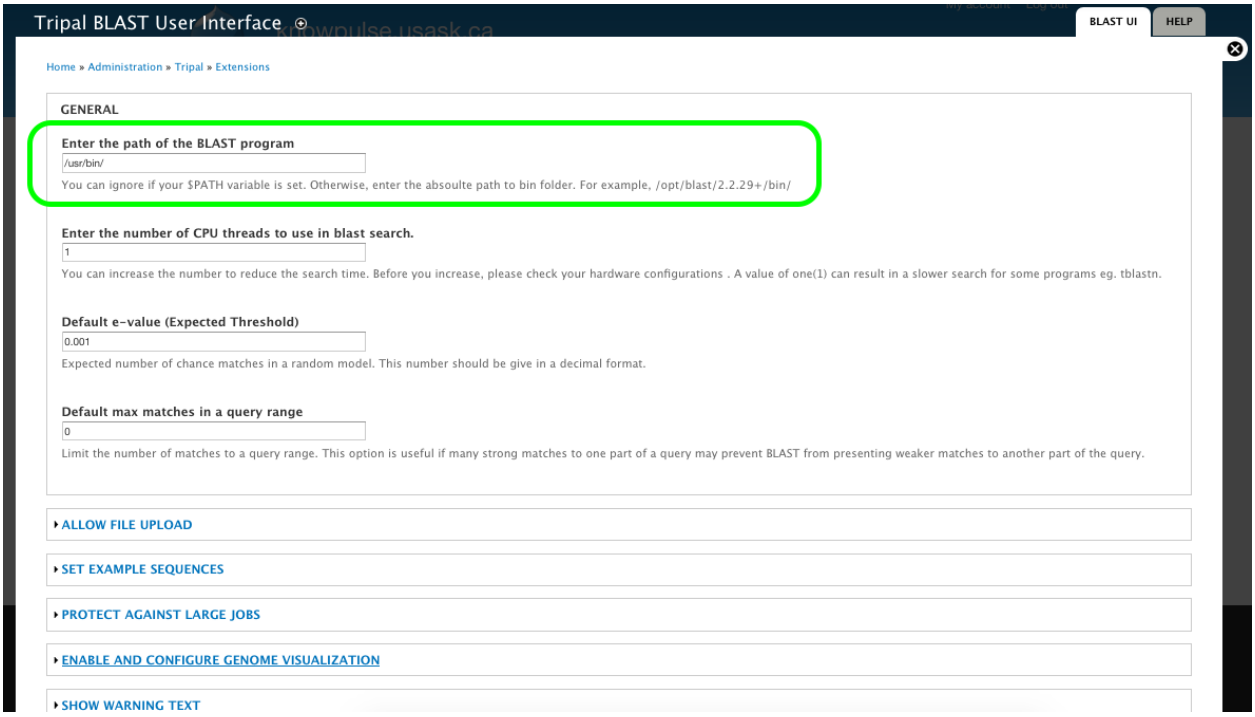
Now that the module is installed, we just need to configure it!

1.2.4 Configure Tripal BLAST

Navigate to Administration Toolbar > Modules and scroll down to BLAST UI (under "Tripal Extensions"). Then click on the configure link as shown below:

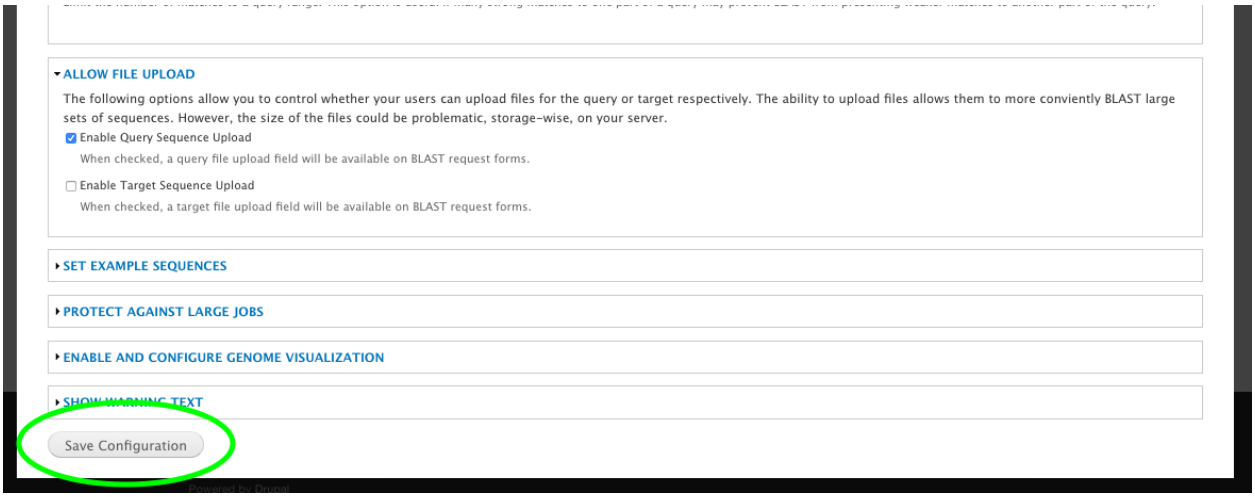


This will take you to the Tripal BLAST configuration form. The only required settings is the "path of the BLAST program". This should be set to the absolute path to the blastn executable and should include the final slash but not the program itself (e.g. `/usr/bin/`).



The remaining configuration options allow you to customize Tripal BLAST UI to your own specific needs. For example, you can use the options under "Allow file upload" to allow users to allow FASTA files for either the query and/or the target database. Additionally, you can set the example sequences, protect against large jobs by limiting the number of results and/or add a warning to the top of the blast form.

Don't forget to click the "Save Configuration" button at the bottom of the page to ensure your changes are saved!



1.3 Running Jobs Automatically

BLAST submissions result in the creation of Tripal jobs which then need to run from the command-line. This ensures that long running BLASTs will not cause page time-outs but does add some management overhead and might result in longer waits for users depending on how often you have cron set to run Tripal jobs. You can alternatively use the Tripal Jobs Daemon to automate running of Tripal Jobs reducing user wait time and your own workload.

Note: Tripal Daemon Documentation

Warning: If you find jobs are not running automatically, you may need to restart the Tripal Daemon. This is also necessary after a server restart. Navigate to your drupal root (e.g. `/var/www/html`) on the command-line and run:

```
drush trpjob-daemon stop
drush trpjob-daemon start
```

1.4 Blast Target Databases

"Target Database" is the BLAST terminology for a database you want your users to be able to BLAST against. For example, on the NCBI Blast website they have a nucleotide and protein target database.

1.4.1 Creating Blast Indicies

This section provides instructions for how to prepare a FASTA file for use with BLAST. We use the MCBI+ Blast command *formatdb* which should have been installed along-side the other blast command-line tools. The following command can be used to create a nucleotide database from the fasta file `my_nucleotide.fasta` where resulting files have the name `Genus_species_version_genome`.

```
formatdb -p F -o T -i my_nucleotide.fasta -t Genus_species_version_genome -n Genus_
↪species_version_genome
```

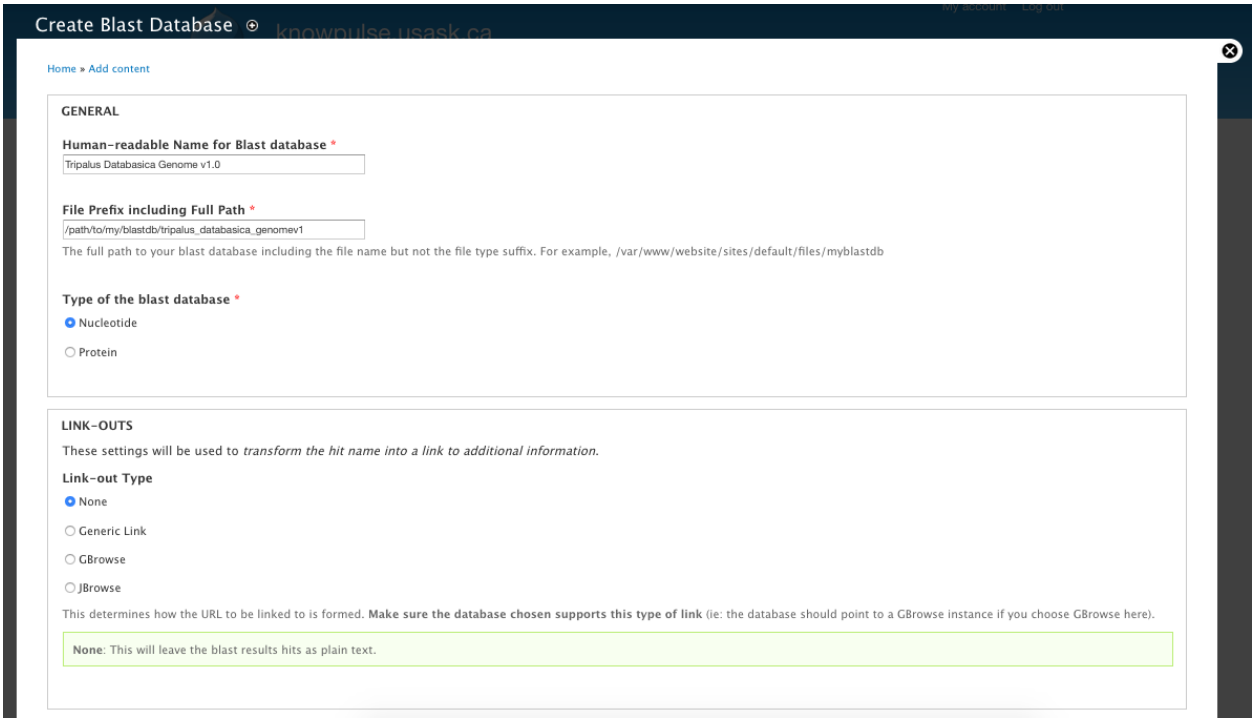
Note: The following indicates what each paramter does:

```
formatdb --help

formatdb 2.2.26   arguments:
-t  Title for database file [String]  Optional
-i  Input file(s) for formatting [File In]  Optional
-n  Base name for BLAST files [String]  Optional
-p  Type of file [T/F]  Optional
   T - protein
   F - nucleotide
-o  Parse options
    T - True: Parse SeqId and create indexes.
    F - False: Do not parse SeqId. Do not create indexes.
```

1.4.2 Add Blast Database

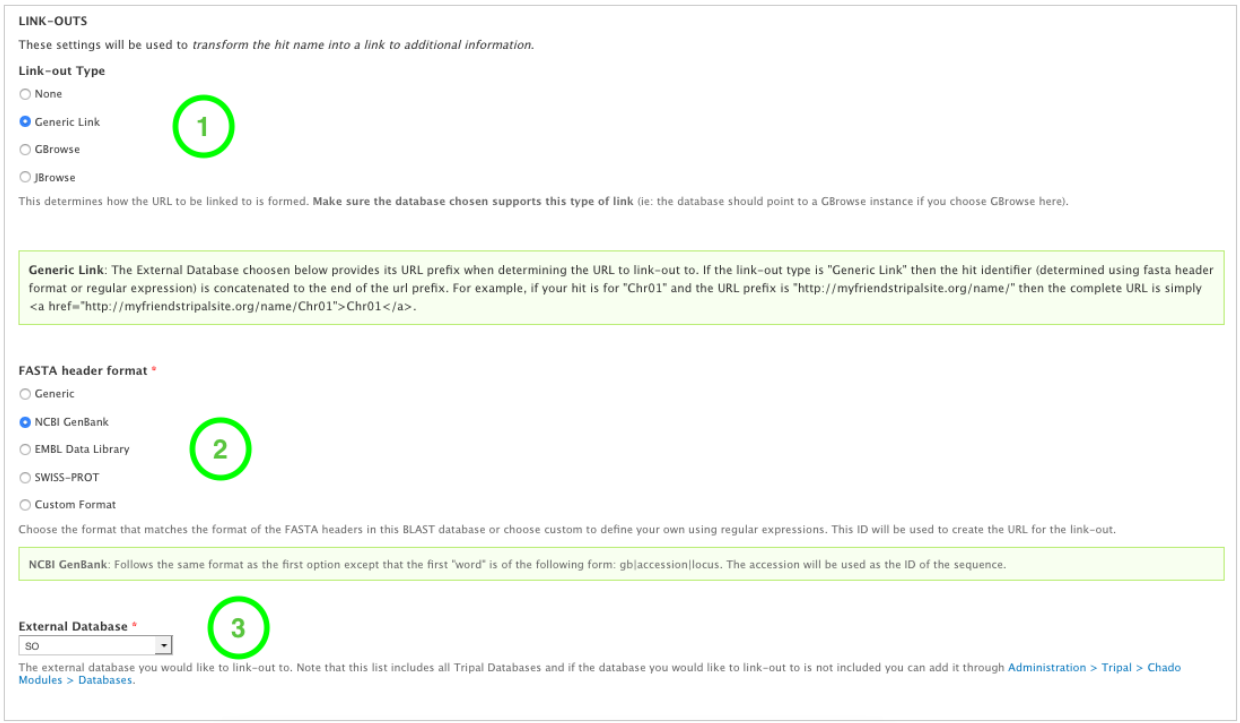
To add one to the "BLAST Databases" drop-down on the Blast program forms, in the "Navigation" menu go to "Add Content" > "Blast Database". Then fill out the form with the human readable name of your blast database (shown to the user in the drop-down) and the path to the blast database (passed to NCBI Blast).



For example, the above form will add "Tripalus Databasica Genome v1.0" to the "BLAST Databases" drop-down on the Nucleotide BLAST (blastn) form.

1.4.3 Linkouts

These settings will be used to transform the hit name into a link to additional information.



Linkout Type

The linkout type determines how the URL will be formed. When configuring the linkouts for a given blast database, you first choose the type (i.e. Generic, GBrowse, JBrowse) based on the descriptions above. This is very dependent upon the FASTA headers used to create the BLAST database.

- **Generic Link**: Creates a generic link using a Tripal External Database and the backbone names from the blast database.
- **GBrowse Link**: Creates a link to highlight blast results on an existing GBrowse. This requires the blast database consist of backbone sequences of the same name and version as the GBrowse instance.
- **JBrowse Link**: Creates a link to highlight blast results on an existing JBrowse. This requires the blast database consist of backbone sequences of the same name and version as the JBrowse instance.

Warning: You cannot use the **GBrowse and JBrowse linkout types** unless your target BLAST database consists of the same records with the same names as the backbone of your GBrowse/JBrowse instance. For example, if your JBrowse instance consists of Lens culinaris genome v1.0 with LcChr1, LcChr2, etc. then your BLAST database must consist of the exact same genome version with the original FASTA record containing *>LcChr1*.

Note: **Generic linkouts** are great for linking BLAST results to either your own Tripal pages or external pages such as NCBI Genbank.

FASTA Header Format

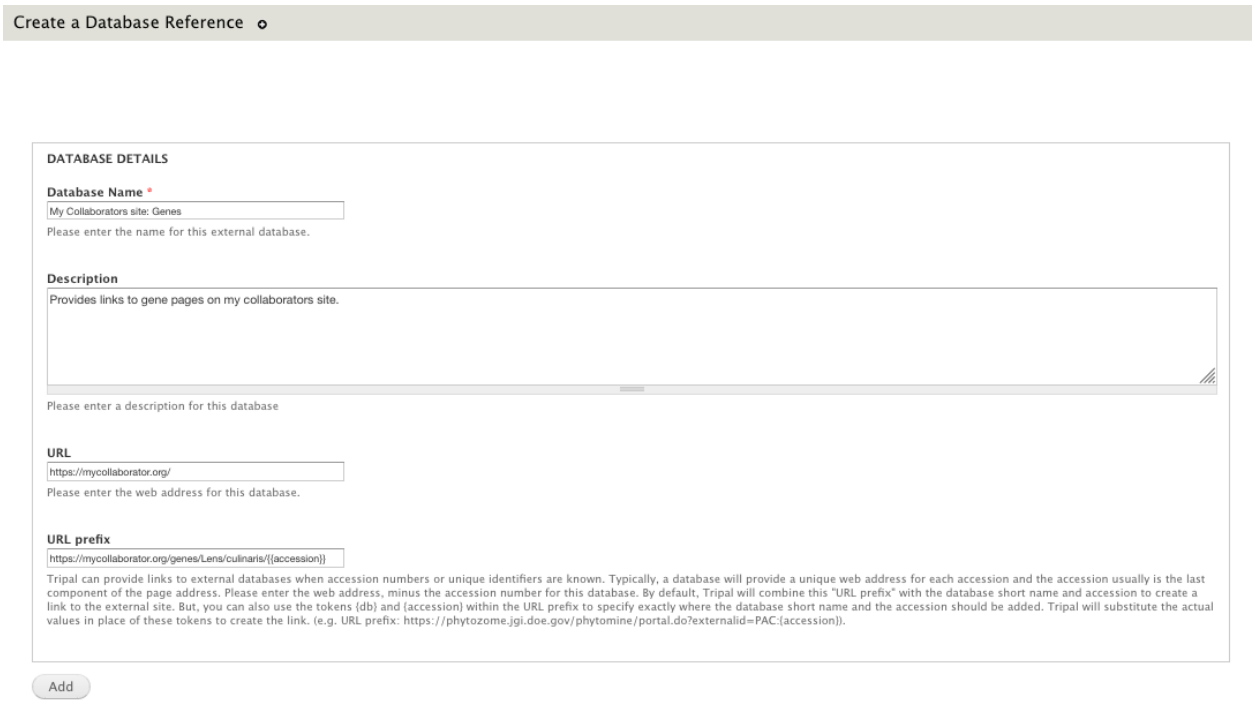
This section is for indicating the format of the original FASTA record used to create the blast database. For example, if you downloaded a FASTA file from NCBI Genbank and then used *formatdb* to make it your target BLAST database, then you want to choose "NCBI Genbank" as the **FASTA Header Format**.

If you have a FASTA header that doesn't match any of those below, then you can choose **Custom Format** and enter your own *PHP-compliant regular expression <http://php.net/manual/en/reference.pcre.pattern.syntax.php>*`_. The regular expression should include the opening and closing forward slashes (i.e. `/`) and curved brackets around the section you would like to be used for the linkout (e.g. `/^>.*(LcChr\d+).*$/`) if you would like to capture LcChr1, LcChr2, etc. It is always a good idea to test your regular expression using online tools.

External Database

This section uses the Tripal API, (i.e. Tripal External Databases) to allow you to choose the URL prefix for your linkouts. A Tripal External Database consists of a label, which is shown in the drop-down, and both a URL and URL prefix. The URL prefix will be used with the record name extracted using the FASTA header settings above to create the linkout for your users. If the Tripal External Database already exists on your Tripal site, simply select it from the drop-down.

If it does not already exist then you must first create it by going Administration > Tripal > Data Loaders > Chado Databases > Add Database. The most important elements are the "Database Name", which will appear in the drop-down on the "Blast Database" page once you refresh it and the "URL Prefix" which will be used to create the linkout. For more information on configuring Tripal databases, see the Tripal User's Guide.



1.5 Whole Genome BLAST Hit Visualization (CViTjs)

1. Download CViTjs and copy the code to your webserver. It needs to be placed in `[your drupal root]/sites/all/libraries`. To download, execute the git command inside the `libraries/` directory:

```
git clone https://github.com/LegumeFederation/cvitjs.git
```

2. CViTjs will have a config file in its root directory named `cvit.conf`. This file provides information for whole genome visualization for each genome BLAST target. Make sure the config file can be edited by your web server.
3. Enable CViTjs from the BLAST module administration page.
4. Edit the configuration file to define each genome target. These will look like:

```
[data.Cajanus cajan - genome]
conf = data/cajca/cajca.conf
defaultData = data/cajca/cajca.gff
```

Where:

- the section name, “data.Cajanus cajan - genome”, consists of “data.” followed by the name of the BLAST target node,
- the file “cajca.conf” is a cvit configuration file which describes how to draw the chromosomes and BLAST hits on the Cajanus cajan genome,
- and the file “cajca.gff” is a GFF3 file that describes the Cajanus cajan chromosomes.

At the top of the configuration file there must be a [general] section that defines the default data set. For example:

```
[general]
data_default = data.Cajanus cajan - genome
```

5. Edit the nodes for each genome target (nodes of type "BLAST Database") and enable whole genome visualization. Remember that the names listed in the CViTjs config file must match the BLAST node name. In the example above, the BLAST database node for the Cajanus cajan genome assembly is named "Cajanus cajan - genome"

1.5.1 Notes

- The .conf file for each genome can be modified to suit your needs and tastes. See the sample configuration file, data/test1/test1.conf, and the CViTjs documentation.
- Each blast target CViTjs configuration file must define how to visualize blast hits or you will not see them.

```
[blast]
feature = BLASTRESULT:match_part
glyph   = position
shape = rect
color   = #FF00FF
width = 5
```

- You will have to put the target-specific conf and gff files (e.g. `cajca.conf` and `cjca.gff`) on your web server, in the directory, `sites/all/libraries/cvitjs/data`. You may choose to group files for each genome into subdirectories, for example, `sites/all/libraries/cvitjs/data/cajca`.
- It is important to make sure that `cvit.conf` points to the correct data directory and the correct `.gff` and `.conf` files for the genome in question. For more information about how to create the `.gff` file, see the documentation.

CHAPTER 2

Developer Guide

A guide for module developers on how to customize and/or extend Tripal BLAST UI.

2.1 Custom Link-outs

In Tripal BLAST "Linkouts" refer to changing the hit name in the BLAST results table to a link. This link usually gives the user additional information and may link to pages in your Tripal site, external websites or genome browsers. You can configure link-outs per BLAST Database and depending on the type, many link-outs support regular expression for extracting parts of the name. The types provided by Tripal BLAST also require you select a Tripal Database (Tripal > Chado Modules > Databases) which contains the URL information for the link. If the link-out types supplied by Tripal BLAST do not fit your needs you can create a custom type using the documentation below.

To create custom link-outs for Tripal BLAST you need to first create your own Drupal module. If you are unfamiliar with this process there are a number of good tutorial available in addition to the Drupal Documentation.

Once you have a custom module you need to implement hook_blast_linkout_info() to tell Tripal BLAST about your custom link-out. You do this by creating a function with the name of your module replacing the word "hook". For example:

```
/**
 * Implements hook_blast_linkout_info().
 * Provides a custom link-out type for my institutes genome browser.
 */
function mymodule_blast_linkout_info() {
  $types = array();

  $types['mybrowser'] = array(
    // Human-readable Type name to display to users in the BLAST Database
    // create/edit form.
    'name' => 'UofS Browser',
    // The function used to generate the URL to be linked to.
    // This function will have full access to the blast hit and database
    // prefix information and is expected to return a URL.
```

(continues on next page)

(continued from previous page)

```
    'process function' => 'mymodule_generate_linkout_mybrowser',
    // Help text to show in the BLAST Database create/edit form so that
    // users will know how to use this link-out type. Specifically, info
    // about your assumptions for the URL prefix are very helpful.
    // HTML is aloud but do not enclose in <p>.
    'help' => 'This type assumes your blast database is the reference for one
      of the University of Saskatchewan Genome Browsers and that you have selected
      the Tripal Database referencing that browser below.',
    // Whether or not the link-out requires additional fields from the nodes.
    'require_regex' => TRUE,
    'require_db' => TRUE,
  );

  return $types;
}
```

Next you need to implement the process function that you indicated. This function is given a number of variables providing information about the hit, etc. and is expected to generate a fully rendered link based on that information. For example,

```
/**
 * Generate a link to the UofS Genome Browser for a given hit.
 *
 * @param $url_prefix
 *   The URL prefix for the BLAST Database queried.
 * @param $hit
 *   The blast XML hit object. This object has the following keys based on the
 *   XML: Hit_num, Hit_id, Hit_def, Hit_accession, Hit_len and Hit_hsps.
 *   Furthermore, a linkout_id key has beek added that contains the part of the
 *   Hit_def extracted using a regex provided when the blastdb node was created.
 * @param $info
 *   Additional information that may be useful in creating a link-out. Includes:
 *    - query_name: the name of the query sequence.
 *    - score: the score of the blast hit.
 *    - e-value: the e-value of the blast hit.
 * @param $options
 *   Any additional options needed to determine the type of link-out. None are
 *   supported by this particular link-out type.
 *
 * @return
 *   An html link.
 */
function tripal_blast_generate_linkout_link($url_prefix, $hit, $info, $options =␣
↪array()) {

  if (isset($hit->{'linkout_id'})) {

    // This is where you would generate your link. If your link requires query␣
↪parameters
    // then we suggest you use l() $options['query'] to encode them rather than␣
↪appending
    // them to the URL prefix directly.
    // This StackExchange question shows a good example:
    //   http://drupal.stackexchange.com/questions/38663/how-to-add-additional-url-
↪parameters
    $hit_url = $url_prefix . $hit->{'linkout_id'};
```

(continues on next page)

(continued from previous page)

```
    // See the documentation for l():
    //  https://api.drupal.org/api/drupal/includes%21common.inc/function/l/7
    return l(
      $hit->{'linkout_id'},
      $hit_url,
      array('attributes' => array('target' => '_blank'))
    );
  }
  else {
    return FALSE;
  }
}
```

2.2 Custom Styling

The BLAST module forms can be styled using CSS stylesheets in your own theme. By default it will use the default form themeing provided by your particular Drupal site allowing it to feel consistent with the look-and-feel of your Tripal site without customization being needed.

Additionally, the results page, waiting pages and the alignment section of the results page have their own template files (`blast_report.tpl.php`, `blast_report_pending.tpl.php`, and `blast_report_alignment_row.tpl.php`, respectively) which can easily be overridden in your own theme providing complete control over the look of the BLAST results.

2.3 Contribution Guidelines

The following guidelines are meant to encourage contribution to Tripal BLAST UI source-code on GitHub by making the process open, transparent and collaborative.

2.3.1 Github Communication Tips

- Don't be afraid to mention people (@username) who are knowledgeable on the topic or invested. *We are academics and overcommitted, it's too easy for issues to go unanswered: don't give up on us!*
- Likewise, don't be shy about bumping an issue if no one responds after a few days. *Balancing responsibilities is hard.*
- Want to get more involved? Issues marked with "Good beginner issue" are a good place to start if you want to try your hand at submitting a PR.
- Everyone is encouraged/welcome to comment on the issue queue! Tell us if you
  - are experiencing the same problem
  - have tried a suggested fix
  - know of a potential solution or work-around
  - have an opinion, idea or feedback of any kind!
- Be kind when interacting with others on Github! (see Code of Conduct below for further guidelines). We want to foster a welcoming, inclusive community!

  - Constructive criticism is welcome and encouraged but should be worded such that it is helpful :-) Direct criticism towards the idea or solution rather than the person and focus on alternatives or improvements.

2.3.2 Bugs

- Every bug **should** be reported as a Github issue.
  - Even if a bug is found by a committer who intends to fix it themselves immediately, they **should** create an issue and assign it to themselves to show their intent.
- Please follow the issue templates as best you can. This information makes discussion easier and helps us resolve the problem faster.
  - Also provide as much information as possible :-) Screenshots or links to the issue on a development site can go a long way!

2.3.3 Feature Requests

- Every feature request should start as an issue so that discussion is encouraged :-)
- Please provide the following information (bold is required; underlined strengthens your argument):
  - **Use Case:** fully describe why you need/want this feature
  - Generally Applicable: Why do you feel this is generally applicable? Suggest other use cases if possible. Mention (@) others that might want/need this feature.
  - Implementation: Describe a possible implementation. Bonus points for configuration, use of ontologies, ease of use, permission control, security considerations
- All features **should** be optional so that site admin can choose to make it available to their users.
  - When applicable, new features should be designed such that site admin can disable them.
  - Bonus points: for making new features configurable and easily themed.

2.3.4 Pull Request (PR) Guideline

The goal of this document is to make it easy for **A)** contributors to make pull requests that will be accepted, and **B)** Tripal committers to determine if a pull request should be accepted. - PRs that address a specific issue **must** link to the related issue page.

- Really in almost every case, there should be an issue for a PR. This allows feedback and discussion before the coding happens. Not grounds to reject, but encourage users to create issues at start of their PR. Better late than never :).
- Each PR **must** be tested/approved by at least one "trusted committer."
  - Testers **should** describe how the testing was performed if applicable (allows others to replicate the test).
  - Our guiding philosophy is to encourage open contribution. With this in mind, committers should **work with contributors** to resolve issues in their PRs. PRs that will not be merged should be closed, **transparently citing** the reason for closure. In an ideal world, features that would be closed are discouraged at the **issue phase** before the code is written!
  - The pull request branch should be deleted after merging (if not from a forked repository) by the person who performs the merge.
- PRs **should** pass all Travis-CI tests before they are merged.

- Branches **should** follow the following format: [issue_number]-[short_description]
- **Must** follow Drupal code standards:
- PRs for new feature should remain open until adequately discussed (see guidelines below).

Note: If you need more instructions creating a pull request, see for example the KnowPulse workflow

2.3.5 Code of Conduct

- Be nice! If that's insufficient, Tripal community defers to https://www.contributor-covenant.org/