Tripal Apollo Documentation

Release 0.1

Bradford Condon

Sep 13, 2023

DOCUMENTATION:

CHAPTER

ONE

INTRODUCTION & BACKGROUND

The Tripal Eutils module connects your Tripal site to NCBI.

1.1 What is Eutils?

E-utilities is NCBI's API for all of its databases. Read more at: https://www.ncbi.nlm.nih.gov/books/NBK25500/

1.2 Module features

- Support for Assembly (Chado Analysis), BioProject (Project), and BioSamples (Biomaterial)
- Lookup NCBI records and create Chado base records
- Add properties, DBXREF links
- Lookup and insert records linked in the primary record.

1.2.1 Planned

- Provide fields to create entities with only an NCBI accession

CHAPTER

TWO

INSTALLATION AND SETUP

2.1 Requirements

Tripal EUtilities requires:

- Tripal 3
- PHP >= 7.0
- Drupal 7

2.2 Installation

`tripal_eutils` is not available for deployment via Drush and must be installed via git.

```
cd [location of your custom or contrib modules]
git clone https://github.com/NAL-i5K/tripal_eutils.git
drush pm-enable tripal_eutils -y
```

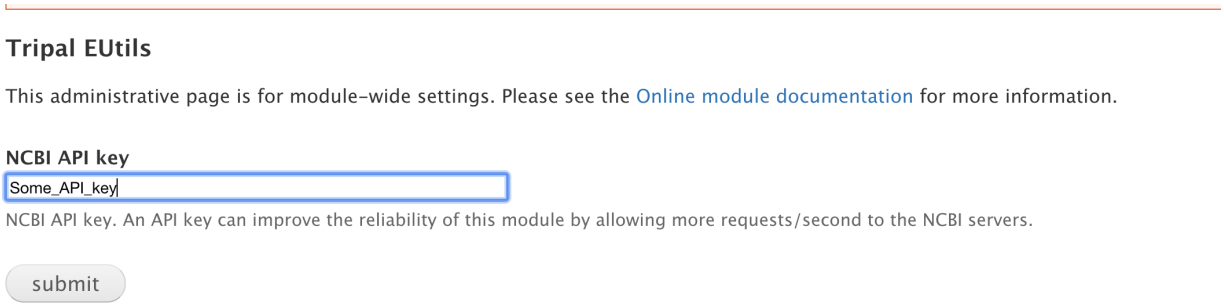
2.3 Chado

This module requires Chado 1.3 or greater. Visit `/admin/tripal/storage/chado/install` on your site to verify and/or upgrade your Chado version.

2.4 Setup

This module currently functions "as is" without setup. The Manage Analyses module provides several new fields (analysis and organism linker fields) so you should **Check For New Fields** on the content types your site utilizes that have _organism or _analysis Chado linker tables.

Additional module-wide settings can be configured at: `/admin/tripal/tripal_eutils`.



2.4.1 NCBI API Key

To get the most out of this module, we suggest setting up an NCBI API key for your site.

NCBI limits requests to a maximum of three/second. If you use this module to import linked records, you may exceed that, and might benefit from adding an API key. This NCBI blog post details the reasoning behind their policy, and provides instructions for getting a key.

2.4.2 Permissions

This module only defines one permission: `access tripal_eutils admin`. This permission will allow users to use the admin form to directly insert Chado records into the database given NCBI accessions. Because this form adds data to your db, we suggest reserving it for administrators.

2.4.3 Updating

As of database update 7303, the database and controlled vocabulary for terms used by this module have been converted from ncbi_properties to NCBI Biosample Attributes. This was done to match the Tripal Biomaterial module's schema, which also overhauled its terminology, coincidentally also in database update 7303. Instructions for updating to the new schema can be found in the Tripal Biomaterial module. There is a "Module Updating" section that applies to both of these modules.

CHAPTER

THREE

USING TRIPAL EUTILS

3.1 The EUtils accession importer form

The EUtils loader provides a fast and convenient way to import records from NCBI into Chado. It is available at `admin/tripal/loaders/eutils_loader`.

In order to import records, you must choose a **NCBI Database** and provide an **NCBI Accession Number**. Note that the accession number can be provided with, or without, the text database accession. While these numbers are generally equivalent (IE `PRJNA13179` vs `13179` for BioProjects), in some cases they are not (Assemblies).

Note: Please see *Mapping NCBI content into Chado* for a list of currently supported databases, and for more information on how the information from NCBI will be loaded into Chado.

The **Create Linked Records** box, when checked, will not only create the primary accession input above, but any secondary accessions directly referenced in that record.

3.1.1 Previewing Records

After you've selected a database and an accession, you can use the **Preview Record** button to view the metadata and linked records that will be inserted.

The preview will first show the base Chado record created. The values here are typically the primary fields of the Chado table (in this case, `chado.analysis`), as well as any DBXrefs associated with it.

The properties associated with a record come from different tags depending on the NCBI database (as detailed in *Mapping NCBI content into Chado*). They are inserted into Chado into the property linker table: in this case, `chado.analysisprop`.

Finally, the linked records section demonstrates what additional, cross-linked records will be inserted into Chado. In the below example, two BioSamples, a BioProject, and an organism will be inserted.

Note: If the linked record already exists in your database, it will be retrieved and reused.

The linked records area links out to the record on NCBI: click on the **Value** link to double check the record information.

Fig. 1: The EUtils import form (`admin/tripal/loaders/eutils_ncbi_import`).

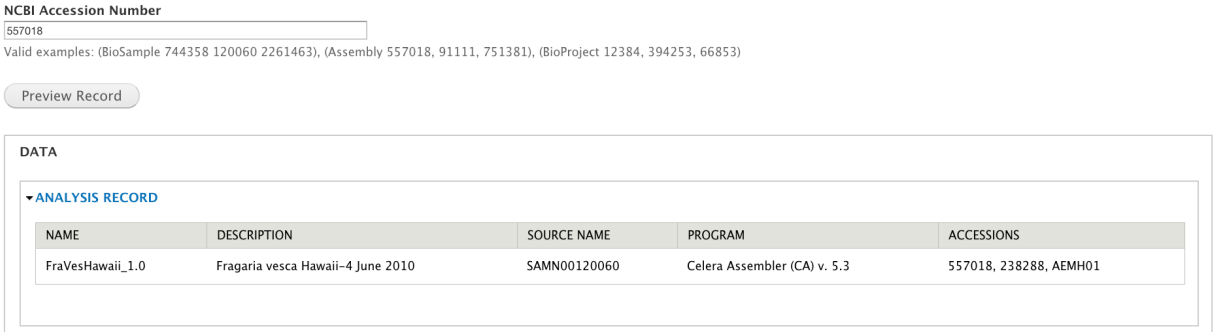
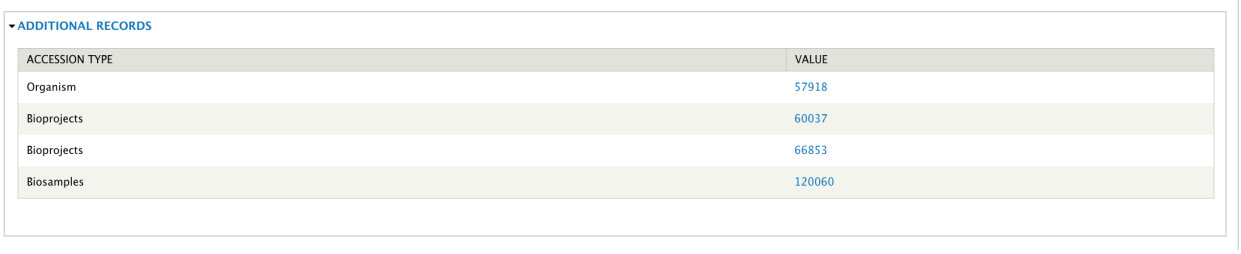


Fig. 2: Previewing the base record of an Assembly.

PROPERTIES

PROPERTY NAME	VALUE
alt_loci_count_all	0
chromosome_count_all	7
contig_count_all	16479
contig_l50_all	2104
contig_n50_all	28051
non_chromosome_replicon_count_all	1
replicon_count_all	8
scaffold_count_all	3048
scaffold_count_placed	8
scaffold_count_unlocalized	0
scaffold_count_unplaced	3040
scaffold_l50_all	4
scaffold_n50_all	27879571
total_length_all	214373013
ungapped_length_all	202038240
Assembly_rpt	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/155/GCF_000184155.1_FraVesHawaii_1.0/GCF_000184155.1_FraVesHawaii_1.0_assembly_report.txt
GenBank	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/184/155/GCA_000184155.1_FraVesHawaii_1.0
RefSeq	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/155/GCF_000184155.1_FraVesHawaii_1.0
Stats_rpt	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/155/GCF_000184155.1_FraVesHawaii_1.0/GCF_000184155.1_FraVesHawaii_1.0_assembly_stats.txt

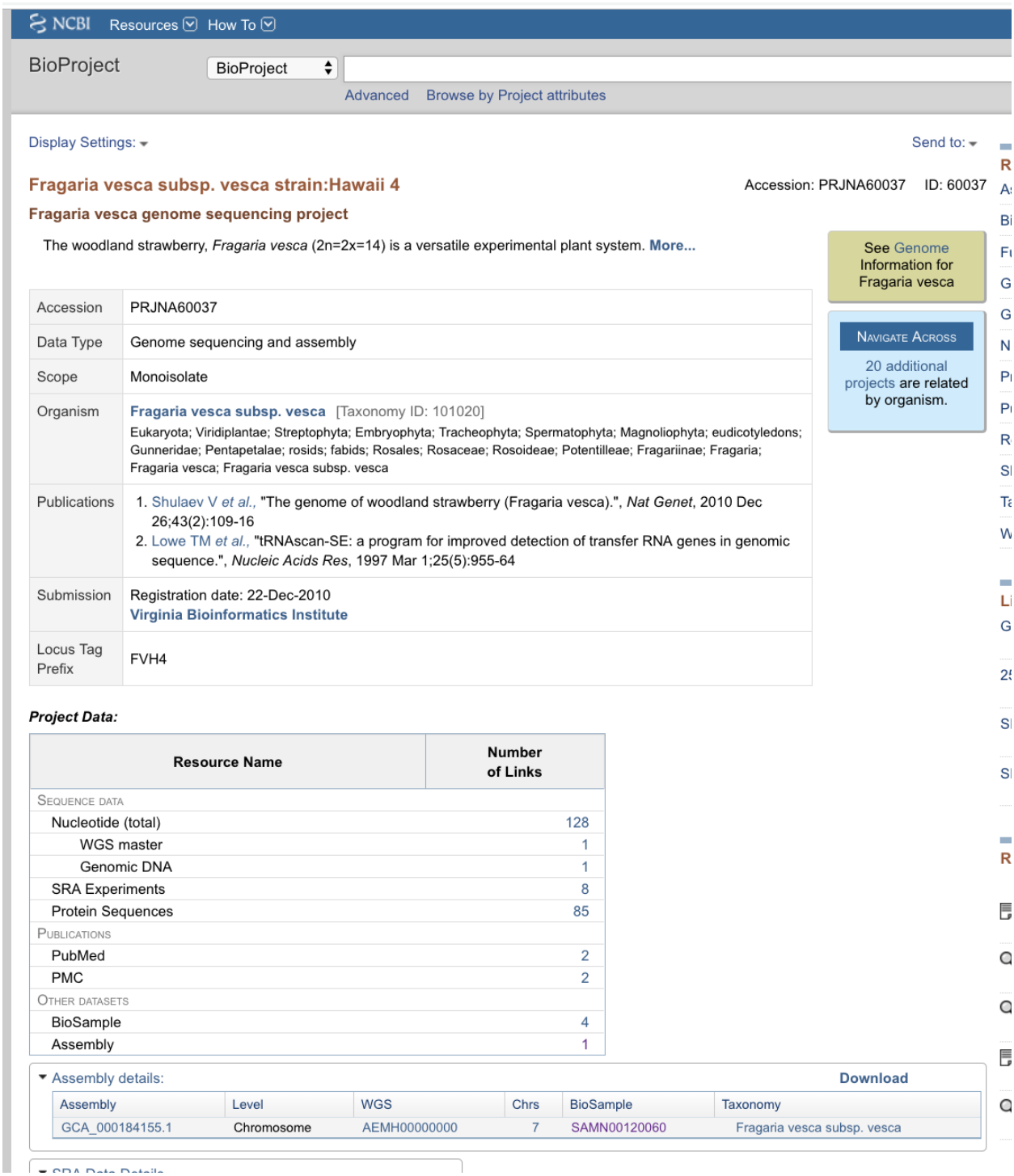
Fig. 3: Previewing the properties of an Assembly.



ADDITIONAL RECORDS

ACCESSION TYPE	VALUE
Organism	57918
Bioprojects	60037
Bioprojects	66853
Biosamples	120060

Fig. 4: Previewing the linked records of an Assembly.


NCBI Resources How To
BioProject
BioProject
Advanced Browse by Project attributes
Display Settings:
Send to:
Fragaria vesca subsp. vesca strain:Hawaii 4
Accession: PRJNA60037 ID: 60037
Fragaria vesca genome sequencing project
The woodland strawberry, Fragaria vesca (2n=2x=14) is a versatile experimental plant system. More...
See Genome Information for Fragaria vesca
Navigate Across
20 additional projects are related by organism.
Accession PRJNA60037
Data Type Genome sequencing and assembly
Scope Monoisolate
Organism Fragaria vesca subsp. vesca [Taxonomy ID: 101020]
Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta; Spermatophyta; Magnoliophyta; eudicotyledons; Gunneridae; Pentapetalae; rosids; fabids; Rosales; Rosaceae; Rosoideae; Potentilleae; Fragariinae; Fragaria; Fragaria vesca; Fragaria vesca subsp. vesca
Publications
1. Shulaev V et al., "The genome of woodland strawberry (Fragaria vesca).", Nat Genet, 2010 Dec 26;43(2):109-16
2. Lowe TM et al., "tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence.", Nucleic Acids Res, 1997 Mar 1;25(5):955-64
Submission Registration date: 22-Dec-2010
Virginia Bioinformatics Institute
Locus Tag Prefix FVH4
Project Data:
Resource Name
Number of Links
SEQUENCE DATA
Nucleotide (total) 128
WGS master 1
Genomic DNA 1
SRA Experiments 8
Protein Sequences 85
PUBLICATIONS
PubMed 2
PMC 2
OTHER DATASETS
BioSample 4
Assembly 1
Assembly details:
Download
Assembly Level WGS Chrs BioSample Taxonomy
GCA_000184155.1 Chromosome AEMH00000000 7 SAMN00120060 Fragaria vesca subsp. vesca

CHAPTER

FOUR

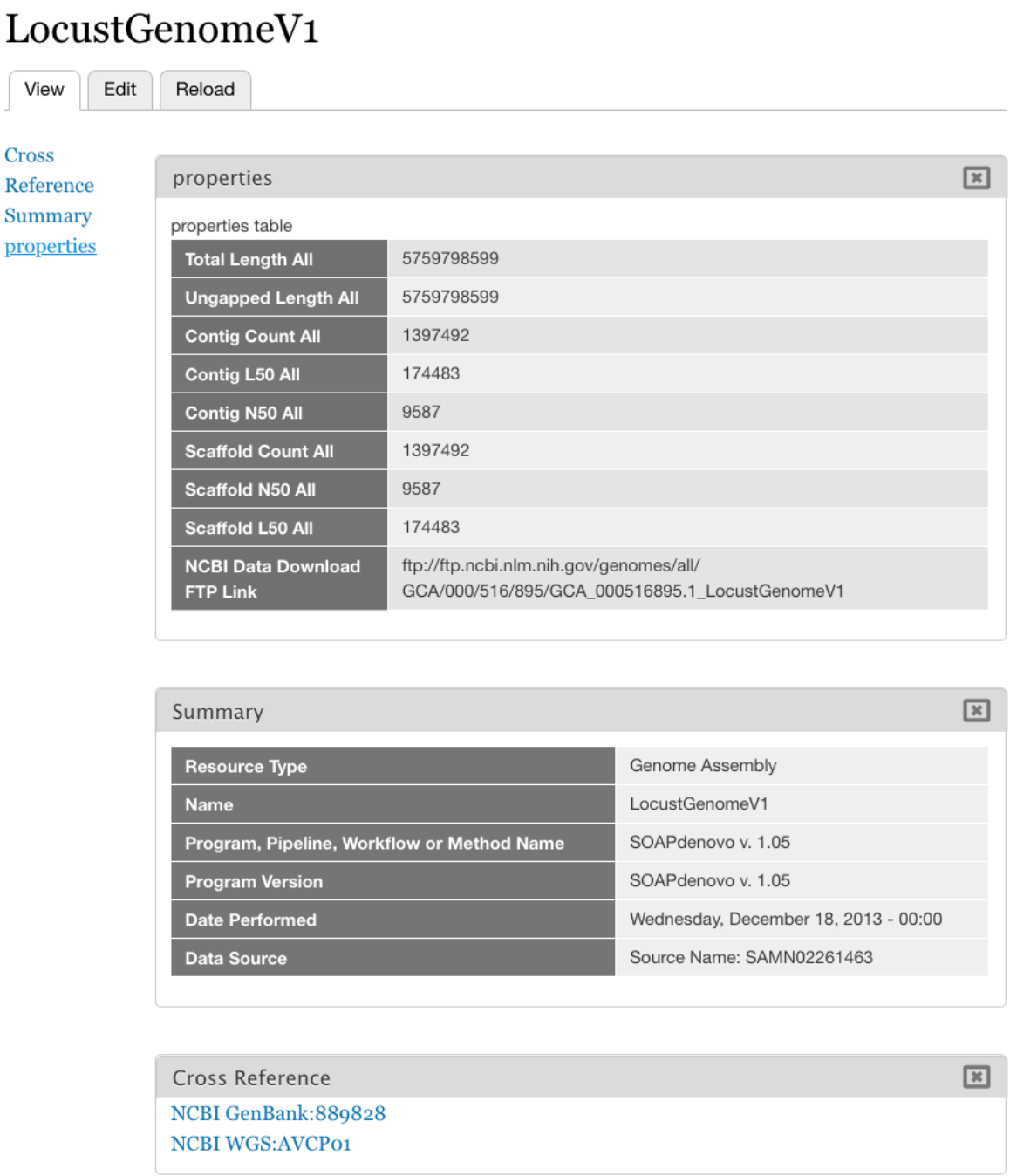
EXAMPLE CREATED CONTENT

This page demonstrates examples of content imported by this module. These are the linked records inserted starting from the BioProject PRJNA185471[1].

[1] 1: Wang X, Fang X, Yang P, Jiang X, Jiang F, Zhao D, Li B, Cui F, Wei J, Ma C, Wang Y, He J, Luo Y, Wang Z, Guo X, Guo W, Wang X, Zhang Y, Yang M, Hao S, Chen B, Ma Z, Yu D, Xiong Z, Zhu Y, Fan D, Han L, Wang B, Chen Y, Wang J, Yang L, Zhao W, Feng Y, Chen G, Lian J, Li Q, Huang Z, Yao X, Lv N, Zhang G, Li Y, Wang J, Wang J, Zhu B, Kang L. The locust genome provides insight into swarm formation and long-distance flight. Nat Commun. 2014;5:2957. doi: 10.1038/ncomms3957. PubMed PMID: 24423660; PubMed Central PMCID: PMC3896762.

4.1 Examples

4.1.1 Assembly



LocustGenomeV1

View | Edit | Reload

Cross Reference
Summary
properties

properties

properties table

Total Length All	5759798599
Ungapped Length All	5759798599
Contig Count All	1397492
Contig L50 All	174483
Contig N50 All	9587
Scaffold Count All	1397492
Scaffold N50 All	9587
Scaffold L50 All	174483
NCBI Data Download FTP Link	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/516/895/GCA_000516895.1_LocustGenomeV1

Summary

Resource Type	Genome Assembly
Name	LocustGenomeV1
Program, Pipeline, Workflow or Method Name	SOAPdenovo v. 1.05
Program Version	SOAPdenovo v. 1.05
Date Performed	Wednesday, December 18, 2013 - 00:00
Data Source	Source Name: SAMN02261463

Cross Reference

NCBI GenBank:889828
NCBI WGS:AVCP01

4.1.2 BioProject

Locusta migratoria: Locusta migratoria Genome sequencing

View | Edit | Reload

Summary
Contact
Cross Reference
Publication
Relationship

Summary

Resource Type	Project
Name	Locusta migratoria: Locusta migratoria Genome sequencing
Short Description	Locusta migratoria Genome sequencing. The strain used for genome sequencing originated from the inbred laboratory strains of solitarious locusts at the Institute of Zoology, CAS, China. Both colonies were reared under a 14:10 light/dark photo regime at 30°C and on a diet of fresh greenhouse-grown wheat seedlings and wheat bran. To produce an even more inbred line, a sibling female adult and male adult mated each other and eight generations of sib mating were then followed to occur. DNA for genome sequencing was extracted from the whole body of one female adult.

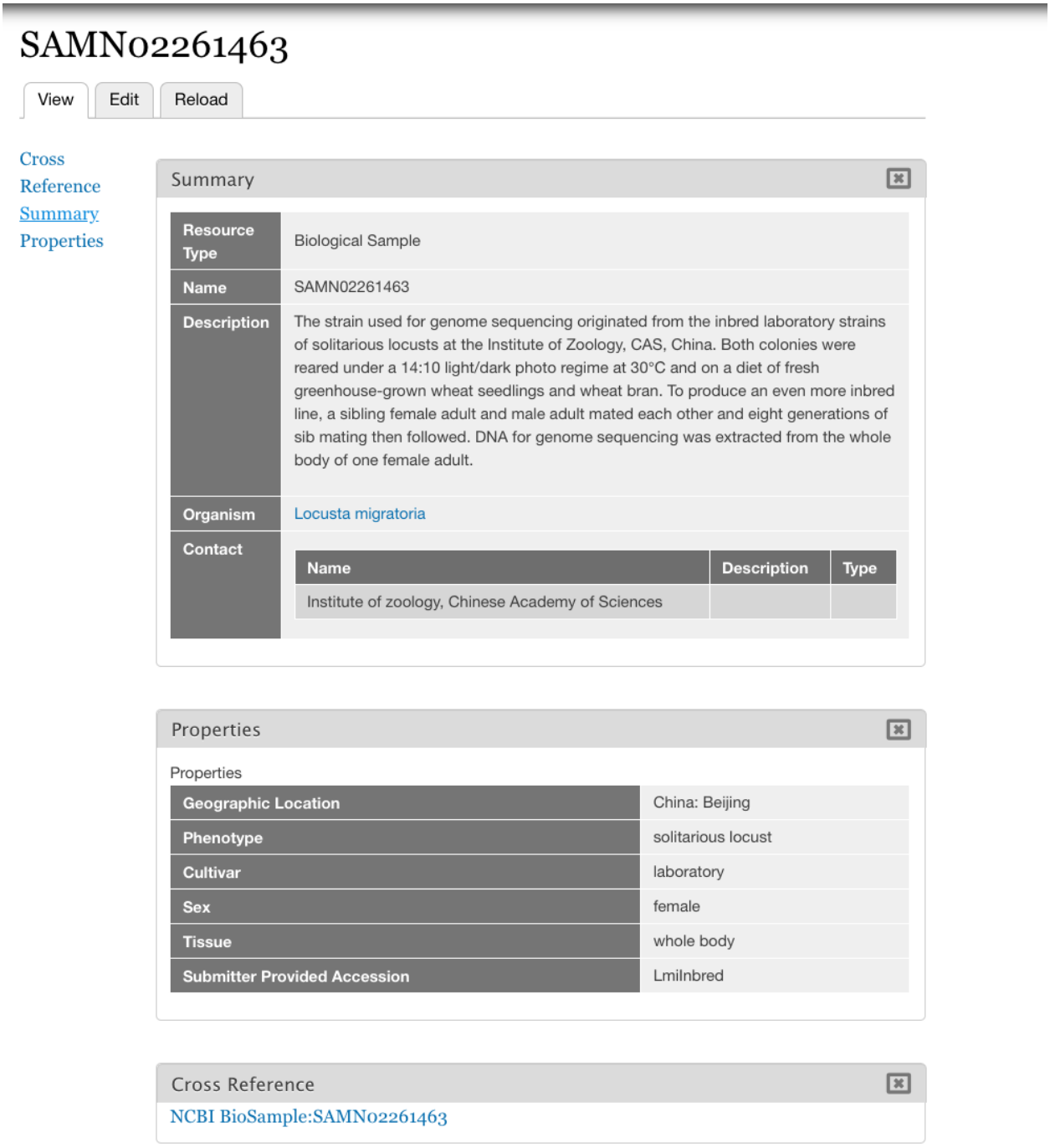
Cross Reference

NCBI BioProject:185471

Publication

Wang X, Fang X, Yang P, Jiang X, Jiang F, Zhao D, Li B, Cui F, Wei J, Ma C, Wang Y, He J, Luo Y, Wang Z, Guo X, Guo W, Wang X, Zhang Y, Yang M, Hao S, Chen B, Ma Z, Yu D, Xiong Z, Zhu Y, Fan D, Han L, Wang B, Chen Y, Wang J, Yang L, Zhao W, Feng Y, Chen G, Lian J, Li Q, Huang Z, Yao X, Lv N, Zhang G, Li Y, Wang J, Wang J, Zhu B, Kang L. The locust genome provides insight into swarm formation and long-distance flight.. Nature communications. 2014; 5:2957.

4.1.3 BioSample



4.1.4 Pubmed

The locust genome provides insight into swarm formation and long-distance flight.

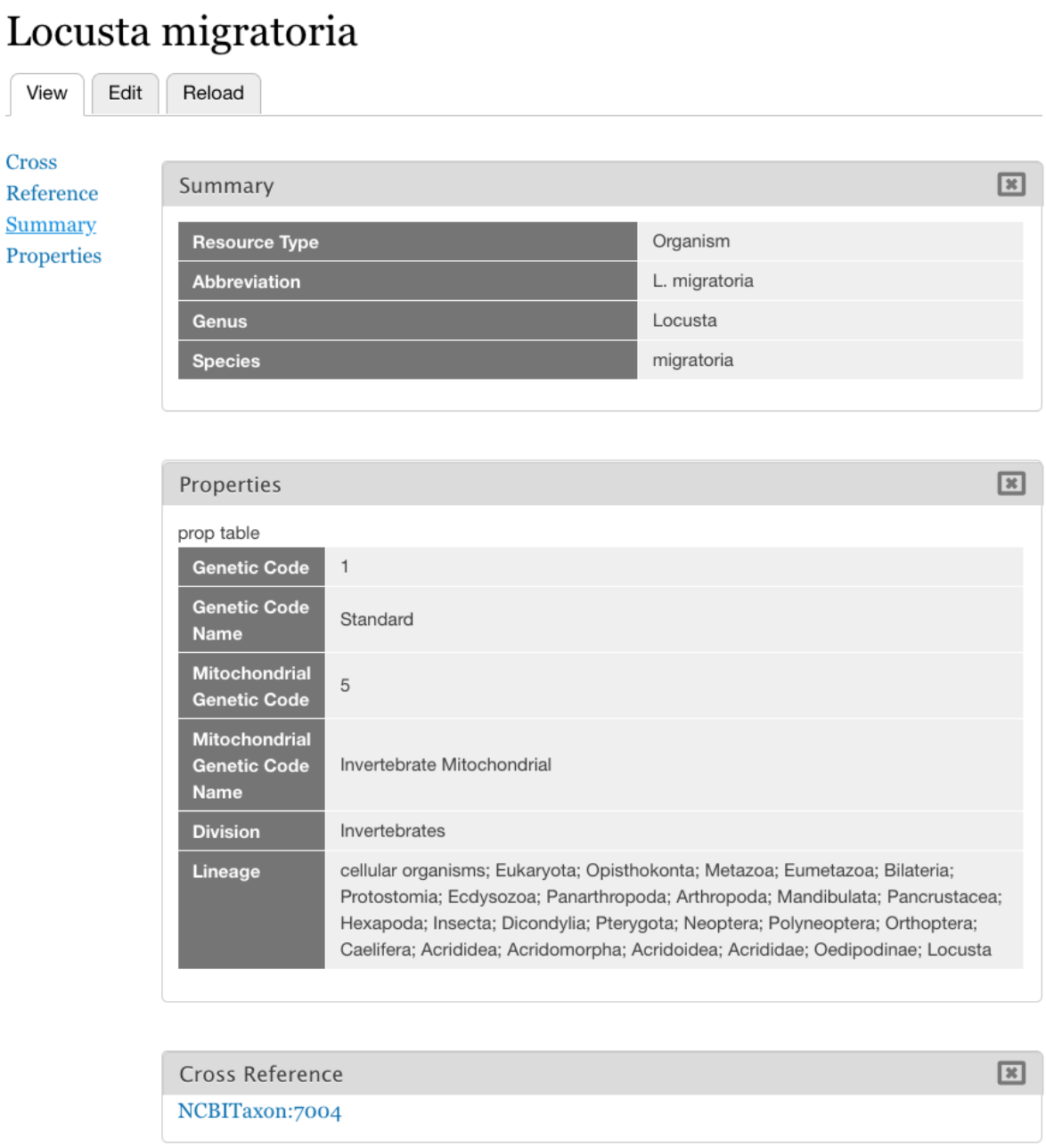
View | Edit | Reload

References
Cross
Reference
Summary

Summary

Publication Type	Journal Article
DOI	10.1038/ncomms3957
Publication Date	2014
Citation	Wang X, Fang X, Yang P, Jiang X, Jiang F, Zhao D, Li B, Cui F, Wei J, Ma C, Wang Y, He J, Luo Y, Wang Z, Guo X, Guo W, Wang X, Zhang Y, Yang M, Hao S, Chen B, Ma Z, Yu D, Xiong Z, Zhu Y, Fan D, Han L, Wang B, Chen Y, Wang J, Yang L, Zhao W, Feng Y, Chen G, Lian J, Li Q, Huang Z, Yao X, Lv N, Zhang G, Li Y, Wang J, Wang J, Zhu B, Kang L. The locust genome provides insight into swarm formation and long-distance flight.. Nature communications. 2014; 5:2957.
Abstract	Locusts are one of the world's most destructive agricultural pests and represent a useful model system in entomology. Here we present a draft 6.5Gb genome sequence of Locusta migratoria, which is the largest animal genome sequenced so far. Our findings indicate that the large genome size of L. migratoria is likely to be because of transposable element proliferation combined with slow rates of loss for these elements. Methylome and transcriptome analyses reveal complex regulatory mechanisms involved in microtubule dynamic-mediated synapse plasticity during phase change. We find significant expansion of gene families associated with energy consumption and detoxification, consistent with long-distance flight capacity and phytophagy. We report hundreds of potential insecticide target genes, including cys-loop ligand-gated ion channels, G-protein-coupled receptors and lethal genes. The L. migratoria genome sequence offers new insights into the biology and sustainable management of this pest species, and will promote its wide use as a model system.

4.1.5 Taxon



Locusta migratoria

View | Edit | Reload

Cross Reference
Summary
Properties

Summary

Resource Type	Organism
Abbreviation	L. migratoria
Genus	Locusta
Species	migratoria

Properties

prop table

Genetic Code	1
Genetic Code Name	Standard
Mitochondrial Genetic Code	5
Mitochondrial Genetic Code Name	Invertebrate Mitochondrial
Division	Invertebrates
Lineage	cellular organisms; Eukaryota; Opisthokonta; Metazoa; Eumetazoa; Bilateria; Protostomia; Ecdysozoa; Panarthropoda; Arthropoda; Mandibulata; Pancrustacea; Hexapoda; Insecta; Dicondylia; Pterygota; Neoptera; Polyneoptera; Orthoptera; Caelifera; Acrididea; Acridomorpha; Acridoidea; Acrididae; Oedipodinae; Locusta

Cross Reference

NCBITaxon:7004

CHAPTER

FIVE

MAPPING NCBI CONTENT INTO CHADO

Unfortunately it isn't always clear how NCBI data should map into Chado.

This section describes what to expect when running the EUtils importer.

5.1 NCBI to Chado

Table 1: NCBI to Chado mappings

NCBI Database	Chado Base Table
BioSample	biomaterial
BioProject	project
Assembly	analysis (but see Chado repo for discussion on this)
NCBI Taxon	organism

5.2 Database specific mappings

5.2.1 Assembly

- https://www.ncbi.nlm.nih.gov/assembly/
- https://laceysanderson.github.io/chado-docs/tables/analysis.html

Table 2: Assembly to Chado.analysis mappings

XML	Chado Base Table	Chado Column
AssemblyName	analysis	name
AssemblyDescription	analysis	description
`# Assembly method:` (from FTP)	analysis	program
`# Assembly method:` (from FTP)	analysis	programversion
N/A	analysis	algorithm
BioSampleAccn	analysis	sourcename
N/A	analysis	sourceversion
N/A	analysis	sourceuri
SubmissionDate	analysis	timeexecuted
Stats	analysisprop	type/value
FtpSites	analysisprop	type/value
SpeciesTaxid	organism	chado.analysis_organism
RsUid	dbxref	chado.analysis_dbxref
GbUid	dbxref	chado.analysis_dbxref
WGS RS_BioProjects/GB_BioProjects	project	(Chado table doesnt exist yet)
BioSampleID	biomaterial	(Chado table doesnt exist yet)
RefSeq_category	analysisprop	rdfs:type (sets the analysis type)

Note that the program and program version are not found directly in the XML. Instead they are extracted from the FTP attribute.

Analysis type

The analysis table has no *type_id* column. The type is therefore set with the *rdfs:type* property.

The *RefSeq_category* tag is used to determine the analysis type. Currently, only the value *representative genome* is supported and mapped to the bundle Genome Assembly (operation:0525), via the type value 'genome_assembly'. We have thus far come across no other values for this key in the database.

Is an assembly a Chado analysis or project?

This is still an **open question**. This module maps NCBI Assemblies into `chado.analysis`, but it may split the NCBI assembly record into an analysis and project in the future. This is because the current definition of a Chado analysis is a **single program run**. Assemblies are typically many programs run in a pipeline.

Undecided mappings

We don't currently know how we will map analyses to biomaterials in Chado. BioSamples that are listed in Assembly records are therefore ignored currently.

5.2.2 BioProject

- **NCBI BioProject**: https://www.ncbi.nlm.nih.gov/bioproject
- **Chado Project**: https://laceysanderson.github.io/chado-docs/tables/project.html

The Chado project table will provide many more linkers in Chado 1.4: until that discussion is resolved, this module will not take full advantage of NCBI BioProjects.

Table 3: BioProject to Chado.bioproject mappings

XML	Chado Base Table	Chado Column
ProjectDescr->Name and ProjectDescr->Title	project	name
ProjectDescr->Description	project	description
ProjectID	dbxref	project_dbxref

Notes and details

Multiple organisms

We do not insert all organisms when importing a project accession.

Sometimes, a project will specify a different species and `taxID` in the Organism tag: `<Organism species="57918" taxID="101020">`. In these cases, the actual biomaterial is derived from the `taxID`, so thats what this module imports.

Indirect mappings

NCBI Taxon (organism) is linked to BioProject (project) indirectly, via BioSamples (biomaterial).

5.2.3 BioSample

NCBI BioSamples are mapped into the Chado.biomaterial table.

- **NCBI database**: https://www.ncbi.nlm.nih.gov/biosample/
- **Chado biomaterial table**: https://laceysanderson.github.io/chado-docs/tables/biomaterial.html
- **Chado MAGE module**: http://gmod.org/wiki/Chado_Mage_Module

Table 4: BioSample to Chado.biomaterial mappings

XML	Chado Base Table	Chado Column
BioSample->accession	biomaterial	name
Owner->Name	contact	biomaterial.biosourceprovider_id
Comment->Paragraph	biomaterial	description
Attribute	biomaterialprop	type_id, value
Organism->taxonomy_id	organism_dbxref	dbxref.accession

Note: In the above table, XML tags are described as Parent_tag->Child_tag. If the value comes from the attribute of a tag, it is written lowercase, as Parent_tag->attribute.

Undecided mappings

We don't currently know how we will map analyses to biomaterials in Chado. Assemblies that are listed in BioSample records are therefore ignored currently.

Attributes

This module does not currently map attributes to ontology terms. Instead, all attributes are put into a "NCBI Property" controlled vocabulary. Suggested attribute - ontology term mappings for the Plant 1.0 and Invertebrate 1.0 BioSample packages are available here: https://data.nal.usda.gov/dataset/data-tripal-eutils-tripal-module-increase-exchange-and-reuse-genome-assembly-metadata. The full attribute set can be downloaded at NCBI.

5.2.4 Pubmed

NCBI pubmed records are mapped into `chado.pub`.

- **NCBI Pubmed**: https://www.ncbi.nlm.nih.gov/pubmed
- **Chado pub table**: https://laceysanderson.github.io/chado-docs/tables/pub.html

Note: Developer's note: publications are imported using the Tripal core *tripal_pub_PMID_parse_pubxml()* and *tripal_pub_add_publications()* functions. Any suggestions or modifications should be made at the Tripal core repo instead.

Table 5: Pubmed to Chado.pub mappings

XML	Chado Base Table	Chado Column
Journal	pub	title
NA	pub	volumetitle
Volume	pub	volume
NA	pub	series_name
Issue	pub	issue
PubDate	pub	pyear
MedlinePgn	pub	pages
NA	pub	miniref
(Citation built from multiple keys)	pub	uniquename
PublicationType	pub	type_id
NA	pub	publisher
NA	pub	pubplace
AuthorList	pubauthor	surname/givennames/suffix

Table 6: pubmed XML keys to Chado.pubprop mappings

XML Key	Property	
Journal->ISOAbbreviation	Journal Abbreviation	
Elocation		
Media Code		
Conference Name		
Keywords		
Series Name		

continues on next page

Table 6 – continued from previous page

XML Key	Property	
pISSN		
Publication Date		
Journal Code		
Journal Alias		
Journal Country		
Published Location		
Publication Model		
Language Abbr		
Alias		
Publication Dbxref		
Copyright		
Abstract		
Notes		
Citation		
Language		
URL		
eISSN		
DOI		
ISSN		
Publication Code		
Comments		
Publisher		
Media Alias		
Original Title		

5.2.5 Taxon

NCBI taxons are mapped into `chado.organism`.

- **NCBI Taxonomy**: https://www.ncbi.nlm.nih.gov/taxonomy
- **Chado Organism table**: https://laceysanderson.github.io/chado-docs/tables/organism.html

---

Note: Developer's note: Taxons are imported using the Tripal core class `TaxonomyImporter.inc`. Any suggestions or modifications should be made at the Tripal core repo instead.

Table 7: Taxon to Chado.organism mappings

XML	Chado Base Table	Chado Column
Taxon->ScientificName	organism	genus
Taxon->ScientificName	organism	species
IdList->Id	organism_dbxref	dbxref.accession
Taxon->Rank	organism	type_id
Taxon->OtherNames->CommonName	organism	commonname
Taxon->ScientificName (if included)	organism	infraspecific_name
NA	organism	comment

Additionally, several properties are parsed into Chado properties for the organism record. However, these all utilize local terms.

Table 8: Taxon properties to Chado.organismprop mappings

XML	Property term
Taxon->Lineage	local:lineage
Taxon->GeneticCode->GCId	local:genetic_code
Taxon->GeneticCode->GCName	local:genetic_code_name
Taxon->MitoGeneticCode->MGCId	local:mitochondrial_genetic_code
Taxon->MitoGeneticCode->MGCName	local:mitochondrial_genetic_code_name
Taxon->Division	local:division
Taxon->OtherNames->GenbankCommonName	local:genbank_common_name
Taxon->OtherNames->Synonym	local:synonym
Taxon->OtherNames->GenbankSynonym	local:synonym
Taxon->OtherNames->Includes	local:other_name
Taxon->OtherNames->EquivalentName	local:equivalent_name
Taxon->OtherNames->Anamorph	local:anamorph

5.3 Linked content

The EUtils admin form has a checkbox to insert linked content. This will only insert content that is **directly linked** to the accession you are importing.

Consider a BioProject with many BioSamples and analyses listed. If you import that BioProject and choose to include linked records, all the directly associated BioSamples and Assemblies will also be imported.

If, however, you only wanted a subset of BioSamples in the database, you could import them individually: each BioSample would link to the BioProject, but the undesired BioSamples would not be imported into Chado. If all the BioSamples of interest were listed in an Assembly project, you could import that Assembly.

Note: Pay attention to the importer preview! The preview lets you double check the correct record will be inserted into the database. It also demonstrates which additional records will be inserted if the "Insert Linked Records" box is checked.

5.3.1 Problematic Links

In other cases, records may be linked **indirectly** via Chado. For example, project and organism cannot be directly linked, but, they are indirectly linked via biomaterials. In those cases, the linked records **will** be inserted. The user does not need to be notified at the preview step. The exception should be documented on that database's page in the documentation.

This is the case for the following links:

- BioProject and NCBI Taxon (linked via biomaterial table)
- Assembly and Biomaterial (linked via the project table).

Linking some content is problematic for the current release of Chado. In cases where a link cannot be made, and the content cannot be linked indirectly, then this module should **not** insert the content. Instead, the formatter should notify the user that those accessions should be added directly.

CHAPTER

SIX

DEVELOPER GUIDE

6.1 Adding support for a new NCBI database

- Implement an `EutilsParserInterface`
- Add the interface to `EutilsXMLParserFactory`
- Create a formatter extending `EutilsFormatter` for displaying previews to the user.
- Create a repository extending `EutilsRepository` for inserting into Chado.
- Add the formatter and repository to their respective factory classes.
- Add the database to the `tripal_eutils_import.form.inc` database list.
- Modify `tripal_eutils.install`, inserting any new databases necessary for the cross references. Also be sure to insert any new cvterms your repository will need.

6.1.1 Connecting to NCBI: Eutils

The Eutils **resource** classes (located in `includes/resources`) are for connecting to the NCBI repository. Not all databses are supported by the ESearch API: we therefore use the Eutils class to provide the correct service.

group **resources**

Some explanation.

Functions

__construct ($create_linked_records=TRUE, $job=NULL)

EUtils constructor.

Parameters

- **$create_linked_records** – Records referenced in the XML will spawn new *EUtils* to import if true.
- **$job** – Tripal Job object.

checkResponseSuccess ($response, string $db, string $accession)

Checks that the resource was found.

Parameters

- **$response** – Response object.
- **$db** – NCBI db string.
- **$accession** – Accession.

Throws

`convertAccessionsToUID (string $db, string $accession)`

Checks the accession and converts to uid accession if neccesary.

Parameters

- **$db** –
- **$accession** – The accession for the NCBI record.

Throws
Returns
bool|string

`get ($db, $accession)`

Queries and parses an NCBI record.

Parameters

- **$db** – NCBI database.
- **$accession** – Numeric only accession.

Throws
Returns
mixed Chado object record.

`getAccessionField ($db)`

`getResourceProvider ($db)`

Returns the appropriate NCBI query method given the database.

Parameters
$db – NCBI database.

Throws
Returns
|| NCBI DB query object.

`setPreview ($preview=TRUE)`

Sets the object to not insert, but only preview the XML.

Parameters
$preview – TRUE will set to preview mode, FALSE unsets.

Variables

$create_linked_records

$job

$preview = FALSE

static static $visited = []

class **BiosamplePropertyLookup**

Fetch the published biosample attributes to feed our property list.

Public Functions

lookupAll (string $url=NULL)

Looks up all attributes.

Parameters

$url – Optional URL string to lookup.

Returns

array Array of terms keyed by harmonized (machine name) with human readable label and definition.

class **EFetch** : public *EUtilsRequest*

Https://www.ncbi.nlm.nih.gov/books/NBK25499/ NCBI Efetch docs.

Public Functions

__construct (string $db)

EFetch constructor.

Parameters

$db – NCBI database string.

Throws

class **EFTP**

Right now responsible for getting a single value from the Assembly.

Public Functions

getField (string $field)

Find all records of line starting with a specific item.

Parameters
$field – The field is the substring to look for at the start of a line.

Returns
array

setURL ($url)

Get the contents of a file at a given URL.

Parameters
$url – The FTP URL.

Throws

class **ESearch** : public *EUtilsRequest*

Queries the *EUtils* esearch API.

Public Functions

__construct (string $db)

ESearch constructor.

Parameters
$db –

Throws

class **ESummary** : public *EUtilsRequest*

Queries the NCBI *ESummary* API.

Public Functions

__construct ($db)

ESummary constructor.

Parameters
$db –

Throws

class **EUtils**

Factory class which returns the appropriate NCBI resource provider.

Public Functions

`__construct ($create_linked_records=TRUE, $job=NULL)`

EUtils constructor.

Parameters

- **$create_linked_records** – Records referenced in the XML will spawn new *EUtils* to import if true.
- **$job** – Tripal Job object.

`convertAccessionsToUID (string $db, string $accession)`

Checks the accession and converts to uid accession if neccesary.

Parameters

- **$db** –
- **$accession** – The accession for the NCBI record.

Throws

Returns

bool|string

`get ($db, $accession)`

Queries and parses an NCBI record.

Parameters

- **$db** – NCBI database.
- **$accession** – Numeric only accession.

Throws

Returns

mixed Chado object record.

`setPreview ($preview=TRUE)`

Sets the object to not insert, but only preview the XML.

Parameters

$preview – TRUE will set to preview mode, FALSE unsets.

class **EUtilsRequest**

Builds and executes the API request to NCBI.

Subclassed by *EFetch*, *ESearch*, *ESummary*

Public Functions

`addHeader ($key, $value)`

Parameters

- **$key** –
- **$value** –

Returns

$this

`addHeaders ($headers)`

Parameters

$headers –

Returns
$this

`addParam ($key, $value)`

Add a single parameter.

Parameters
- **$key** –
- **$value** –

Returns
$this

`addParams ($params)`

Add an array of parameters.

Parameters
$params –

Returns
$this

`get ($url='')`

Send a GET request.

Parameters
$url –

Returns

`post ($url='')`

Send a POST request.

Parameters
$url –

Returns

`setBaseURL ($url)`

Parameters
$url –

Returns
$this

class **EUtilsResource**

Interacts with a response from the *EUtils* API.

Public Functions

`__construct ($response)`

EUtilsResource constructor.

See also:

drupal_http_request()

Parameters
$response – The object returned by drupal_http_request()

`dom()`

Parse response into DOMDocument.

Returns
The response as DOMDocument.

`errorMessage()`

Get the error message.

Returns
string|null The message or null if none exist.

`hasError()`

Find and set errors.

Returns
bool Whether the response has an error element.

`headers()`

Get an array of response headers.

Returns
array

`isSuccessful()`

Check if the request is successful.

Returns
bool TRUE for success.

`originalBody()`

Get raw body response.

Returns
string The raw body response string.

`originalResponseObject()`

Get the response object.

Returns
object The original response object.

`status()`

Get the response status code.

Returns
int Status code.

`xml()`

Parse response into XML.

Returns
The response in XML.

6.1.2 Formatters

Formatters take the output from an XML parser and return a Drupal-form friendly array. Typically a formatter will return a table each for:

- The base Chado record
- Properties
- DBXrefs
- New Chado records created and linked, including organisms, contacts, projects, analyses, etc

EUtilsFormatter

class **EUtilsFormatter**

Subclassed by *EUtilsAssemblyFormatter*, *EUtilsBioProjectFormatter*, *EUtilsBioSampleFormatter*

Public Functions

format (array $data)

Format a parser's output.

Parameters
$data – The data array returned by a parser.

Returns
array This function does not return anything. It directly manipulates the elements array.

getDbLink (string $accession, string $db)

Generates a URL link given a db string and accession.

Parameters

- **$accession** – Accession string.
- **$db** – Database lookup string.

Returns
mixed returns either the accession string, or the accession with a link to the xref.

getNCBIDB (string $db_name)

Fetch the DB object for an NCBI DB.

Parameters
$db_name – The DB name as passed by the parser.

Returns
bool Returns a database object or FALSE.

EUtilsFormatterFactory

class **EUtilsFormatterFactory** : public EUtilsFactoryInterface

EUtilsFormatterFactory.

Creates a formatter given the db.

Public Functions

get (string $db)

Parameters
$db – The database name.

Throws
Returns
The formatter for the given DB.

EUtilsAssemblyFormatter

class **EUtilsAssemblyFormatter** : public *EUtilsFormatter*

Parse *EUtilsAssemblyParser* output for display on a form.

Public Functions

format (array $data)

Add the formatted data into a table.

Parameters
$data – The parsed XML data.

Returns
array Drupal form elements array of each section in a fieldset.

EUtilsBioProjectFormatter

class **EUtilsBioProjectFormatter** : public *EUtilsFormatter*

Class EutilsBioProject Formatter.

Public Functions

`format (array $data)`

Add the formatted data into a table.

Parameters
`$data` – The parsed XML data.

Returns
array Drupal form elements array of each section in a fieldset.

EUtilsBioSampleFormatter

class `EUtilsBioSampleFormatter` : public *EUtilsFormatter*

Class *EUtilsBioSampleFormatter*.

Public Functions

`format (array $data)`

Add the formatted data into a table.

Parameters
`$data` – The parsed XML data.

Returns
array Drupal form elements array of each section in a fieldset.

EUtilsPubmedFormatter

Pubmed records are not directly imported via this module, as this functionality is already provided via Tripal core.

6.1.3 Repositories

Repositories take the output from an XML parser and insert the record into Chado.

For linked records, repositories will spawn new EUtils objects and insert the linked records into Chado.

EUtilsRepository

class `EUtilsRepository`

Subclassed by *EUtilsAssemblyRepository*, *EUtilsBioProjectRepository*, *EUtilsBioSampleRepository*, *EUtilsPubmedRepository*

Public Functions

__construct ($create_linked_records=TRUE)

EUtilsRepository constructor.

Parameters
$create_linked_records – Whether to create linked records.

create (array $data)

Create a new resource.

Parameters
$data – Formatted data returned from the parser.

Returns
object The chado base record object.

createAccession (array $accession)

Create a dbxref record.

Creates a new accession record if it does not exist, and attaches it to the given record.

Parameters
$accession – Expected keys: db and value, where the full accession is db:value.

Throws
Returns
mixed An accession object

createContact ($contact_name)

Get contact name.

Parameters
$contact_name – The contact name.

Throws
Returns
mixed contact record

createProperty ($cvterm_id, $value)

Inserts a property associated with the interface using the tripal API.

Parameters

- **$cvterm_id** –
- **$value** –

Throws
Returns
bool

createXMLProp ($xml)

Associates the XML with the record via the local:full_ncbi_xml term.

Parameters
$xml – string as returned by SimpleXMLElement.

Throws
Returns
bool True on creation

`getAccessionByID ($id)`

Get accession by dbxref id.

Parameters
`$id` –

Returns
mixed

`getAccessionByName ($name, $db_id)`

Search for accession by name.

Look up an accession in chado.dbxref. Retrieves record from cache if predetermined.

Parameters

- `$name` – The accession identifier (dbxref.accession).
- `$db_id` – Name of the DB ID.

Returns
object Accession record.

`getDB ($name)`

Get chado.db record by name. Retrieves data from cache if predetermined.

Parameters
`$name` – The name of database.

Returns
mixed The database object

`getNCBIRecord ($db, array $accessions)`

Fetch and create NCBI records of the type DB.

Parameters

- `$db` – The db name.
- `$accessions` – The accessions for this db.

Throws
Returns
array An array of chado base records, as returned by a repository.

`getOrganism ($accession)`

Given an ncbi taxon organism, return the organism (and create if necessary).

Parameters
`$accession` – NCBITaxon accession for organism.

Throws
Returns
mixed

`linkProjects ($projects)`

Links project to the record, assuming a project_ linker table.

Parameters
`$projects` – Array of base chado record project objects.

`lookupNcbiInChado (string $db, string $accession)`

Looks up a base record based on the accession.

The dbxref is the only reliable way to look up a record since each repository uses different parts of the XML for the base record name, etc.

Parameters

- `$db` – NCBI (not Chado) db name.
- `$accession` – NCBI accession. This might be uid, or long form.

Returns
mixed returns the chado object or FALSE.

`setBaseRecordId ($id)`

Set the Chado record id.

Parameters
`$id` –

Returns
$this

`setBaseTable ($table)`

Set the Chado base table.

Parameters
`$table` – Valid examples include 'organism' , 'biomaterial', 'project'.

Returns
$this

`setJob (TripalJob $job=NULL)`

Sets the TripalJob for error logging.

Parameters
`$job` – Tripal Job object.

`validateFields (array $data)`

Determine whether required fields are provided.

Parameters
`$data` – Formatted data from parser.

Throws

EUtilsRepositoryFactory

class **EUtilsRepositoryFactory** : public EUtilsFactoryInterface

Class *EUtilsRepositoryFactory*.

Public Functions

__construct ($create_linked_records=TRUE)

EUtilsRepositoryFactory constructor.

Parameters
$create_linked_records – Whether to create all linked records.

get (string $db)

Get a repository for a given DB.

Parameters
$db – The database name.

Throws
Returns
An initialized instance of the appropriate repository.

EUtilsAssemblyRepository

class **EUtilsAssemblyRepository** : public *EUtilsRepository*

Maps NCBI Assemblies into a Chado analysis.

Public Functions

addFTPLinks ($ftps)

Associates FTPs as properties.

Parameters
$ftps – Array of key value pars, where the key is the XML FTP type, the value is the FTP address.

static create (array $data)

Create assembly (chado.analysis) record.

Parameters
$data – The data returned by *EUtilsBioProjectParser*.

Throws
Returns
object The created bioProject.

createAnalysis()

Gets/creates this analysis record.

`createLinkedRecords (array $accessions)`

Creates dbxrefs and linked Chado records.

Parameters
`$accessions` – Array of other records indexed type => value.

Throws

`getAnalysis()`

Get analysis from db or cache.

Parameters
`$name` –

Returns
null

`linkOrganism ($organism)`

Insert into organism_analysis, or return existing link.

Parameters
`$organism` – Full chado.organism record.

Throws
Returns
mixed

EUtilsBioProjectRepository

class `EUtilsBioProjectRepository` : public *EUtilsRepository*

Takes parsed bioproject XMLs and creates chado.projects.

Public Functions

`static create (array $data)`

Creates a project and linked records.

Parameters
`$data` – Data from bioproject parser.

Throws
Returns
object chado project record.

`createAccessions (array $accessions)`

Creates a set of accessions attaches them with the given project.

Parameters
`$accessions` –

Returns
array

`createLinkedRecords (array $records, string $type)`

Links this base record to various other records.

Parameters

- **$records** – Array of record ids.
- **$type** – The NCBI record type.

Throws

`createProject (array $data)`

Create a project record.

Parameters
$data – See chado.project schema.

Throws
Returns
mixed

`createProps (array $properties)`

Iterate through the properties and insert.

TODO: How do we get the accessions from what we have here? What we probably have for project is a set of XML attributes or tags…

Parameters
$properties – Properties in form machine name => value.

Throws
Returns
bool True if successful.

`getProject ($name)`

Get project from db or cache.

Parameters
$name –

Returns
null

`linkBiomaterial ($record)`

Links this record to a biosample/biomaterial.

Parameters
$record – A record object returned from getNCBIRecord.

EUtilsBioSampleRepository

class **EUtilsBioSampleRepository** : public *EUtilsRepository*

Class *EUtilsBioSampleRepository*.

Public Functions

static create (array $data)

Takes data from the *EUtilsBioSampleParser* and creates the chado records needed including biosample, accessions and props.

Parameters
$data –

Throws
Returns
object

createAccessions (array $accessions)

Creates a set of accessions attaches them with the given biosample.

Parameters
$accessions –

Returns
array

createBioSample (array $data)

Create a bio sample record.

Parameters
$data – See chado.biomaterial schema.

Throws
Returns
mixed

createProps (array $attributes)

Iterates through the attributes array and creates properties.

Parameters
$attributes – CVterm info from the Attributes area.

Throws

getBioSample ($name)

Get biosample from db or cache.

Parameters
$name –

Returns
null

EUtilsPubmedRepository

Pubmed records are imported via the Tripal Core API.

class **EUtilsPubmedRepository** : public *EUtilsRepository*

Takes parsed pubmed XMLs and creates chado.pub. Uses core API.

Public Functions

create (array $data)

Creates a publication using the core API.

Parameters
$data – Data from bioproject parser.

Returns
pub A Chado publication record object.

6.1.4 XML Parsers

XML parsers take the response from the EUtils resources and extract information from the returned XML.

EUtilsXMLParserFactory

class **EUtilsXMLParserFactory** : public EUtilsFactoryInterface

Class *EUtilsXMLParserFactory*.

This is the base EUTILS XML parser class. The plan is to extend this base class to be specific for each DB type.

Public Functions

get (string $db)

Get the appropriate XML parser.

Parameters
$db – The name of the DB.

Throws
Returns

EUtilsAssemblyParser

class **EUtilsAssemblyParser** : public EUtilsParserInterface

Parser for NCBI Assembly https://www.ncbi.nlm.nih.gov/assembly/ XML.

Public Functions

getFTPData ($url)

Get the fields the assembly object will need from the FTP.

Parameters
$url –

- the ftp site url extracted form the metadata

Returns
array

parse (SimpleXMLElement $xml)

Parameters
$xml –

Returns
array

parseMeta ($x)

Parse the <Meta> tag, which containts CDATA but all of the Assembly props.

Parameters
$x –

Returns
array

EUtilsBioProjectParser

class **EUtilsBioProjectParser** : public EUtilsParserInterface

Class *EUtilsBioProjectParser*.

Note that projects don't have reliable attribute listings.

Public Functions

bioProjectSubmission (SimpleXMLElement $xml)

Parameters
$xml –

Returns
array

parse (SimpleXMLElement $xml)

Parse an NCBI BioProject XML.

Parameters
$xml – Simple XML Element.

Throws
Returns
array|mixed Array.

EUtilsBioSampleParser

class **EUtilsBioSampleParser** : public EUtilsParserInterface

Parses BioSample XML.

Public Functions

parse (SimpleXMLElement $xml)

Parse the XML into an array.

Parameters
$xml –

Throws
Returns
array An array of parsed data

EUtilsPubmedParser

Pubmed records are parsed via the Tripal Core API.

class **EUtilsPubmedParser** : public EUtilsParserInterface

Class *EUtilsPubmedParser*.

Public Functions

parse (SimpleXMLElement $xml)

Parse an NCBI Pubmed XML. Uses the core parser code.

Parameters
$xml – Simple XML Element.

Throws
Returns
array|mixed Array.


4
Tripal EUtilities
Import metadata from NCBI.
1
4

INDEX

B

E