# Python-chado

**May 10, 2022**

# Contents

Contents:

CHAPTER 1

# Chado Library

Test Documentation

A Python library for interacting with a Chado database.

## 1.1 Installation

```
$ pip install chado

# On first use you'll need to create a config file to connect to the database, just␣
↪run:

$ chakin init
Welcome to Chado's Chakin! ()
PGHOST: xxxx
PGDATABASE: xxxx
PGUSER: xxxx
PGPASS:
PGPORT: 5432
PGSCHEMA: public
```

This will create a chakin config file in ~/.chakin.yml

## 1.2 Examples

```
from chado import ChadoInstance
ci = ChadoInstance(dbhost="localhost", dbname="chado", dbuser="chado", dbpass="chado",
↪ dbschema="public", dbport=5432)

# Create human species
org = ci.organism.add_organism(genus="Homo", species="sapiens", common="Human", abbr=
↪"H.sapiens")
```

(continues on next page)

(continued from previous page)

```
# Then display the list of organisms
orgs = ci.organism.get_organisms()

for org in orgs:
    print('{} {}'.format(org.genus, org.species))

# Create an analysis
an = ci.analysis.add_analysis(name="My cool analysis",
                                    program="Something",
                                    programversion="1.0",
                                    algorithm="Google",
                                    sourcename="src",
                                    sourceversion="2.1beta",
                                    sourceuri="http://example.org/",
                                    date_executed="2018-02-03")

# And load some data
ci.feature.load_fasta(fasta="./test-data/genome.fa", analysis_id=an['analysis_id'],␣
↪organism_id=orgs[0]['organism_id'])
ci.feature.load_gff(gff="./test-data/annot.gff", analysis_id=an['analysis_id'],␣
↪organism_id=orgs[0]['organism_id'])
```

Or with the Chakin client:

```
$ my_org=`chakin organism add_organism --species sapiens Homo Human H.sapiens  | jq -
↪r '.organism_id'`

$ chakin organism get_organisms
[
    {
        "organism_id": 1133,
        "genus": "Homo",
        "species": "sapiens",
        "abbreviation": "H.sapiens",
        "common_name": "Human",
        "comment": null
    }
]

# Then load some data
$ my_analysis=`chakin analysis add_analysis \
    "My cool analysis" \
    "Something" \
    "v1.0" \
    "src" | jq -r '.analysis_id'`


$ chakin feature load_fasta \
    --analysis_id $my_analysis \
    --sequence_type contig \
    ./test-data/genome.fa $my_org
```

## 1.3 History

- 2.3.9
  - URL decode GFF ids when loading blast/interpro/others
- 2.3.8
  - Fix connection closed error when loading big interproscan files
- 2.3.7
  - Fix loading of expression data when first column header is not empty
- 2.3.6
  - Fix loading of GO terms from GFF
- 2.3.5
  - Fix has_table() calls with recent sqlalchemy versions
- 2.3.4
  - Now requires biopython >=1.78
  - Fixes biopython sequence usage in recent biopython
- 2.3.3
  - Now requires python >= 3.6
  - Better error reporting for blast loader
- 2.3.2
  - Fix interproscan loader only loading the first result of XML v5
  - Fix interproscan loader failing to load IPR by name
- 2.3.1
  - Fix data loading in Tripal database
- 2.3.0
  - Fix non working –re_parent option in fasta loader
  - allow connection using a preformatted url (needed by galaxy tools using pgutil)
  - added loading of Blast and InterProScan data
  - moved `chakin feature load_go` to `chakin load go`
  - fix sequence computing when landmark sequence is available in the db
  - add more options to match features in expression matrix loader (query_type, match_on_name, re_name, skip_missing)
- 2.2.6
  - fix requirement name for psycopg2 (name change for version >=2.8)
- 2.2.5
  - Added support for units in expression loaders
  - Fix error in load_gff when no source is specified
- 2.2.4

  - Fix broken –skip_missing option for load_go
- 2.2.3
  - Throw a warning instead of an exception when a GFF target feature does not exist
- 2.2.2
  - Bug fixes and improvements to the expression module
- 2.2.1
  - Minor release to fix broken package at pypi, no code change
- 2.2.0
  - Added feature.load_go() to load GO annotation (blast2go results)
  - Added feature.get_feature_analyses() to fetch the analyses associated with a feature
  - Added feature.get_feature_cvterms() to fetch the cvterms associated with a feature
  - Added support for biomaterial/expression data (as used by tripal_analysis_expression)
  - New –protein_id_attr option for feature.load_gff()
- 2.1.5
  - bugfix: fix features deletion when deleting an analysis
- 2.1.4
  - bugfix: fix sporadic errors with AnalysisFeature class declaration
- 2.1.3
  - bugfix: make –species a mandatory arg for organism creation
  - bugfix: fix features deletion when deleting an analysis or an organism
  - update chado docker image
- 2.1.2
  - skip whole database schema reflection for simple tasks (analysis and organism management)
  - fix polypeptide creation for genes beginning at position 0
  - fix various small bugs in phylogeny and featureprop loading
  - fix bug in cvterm creation
  - fix crashes in gbk/gff exporters
- 2.1.1
  - newick: remove prefix from node labels too
  - newick: fix errors with named internal nodes
- 2.1
  - auto reflect db schema
  - add phylogeny module
  - load features from fasta
  - load features from gff3
  - load featureprops from tabular file

  - make chakin util commands work when db is offline
  - add unit tests
- 2.0
  - “Chakin” CLI utility
  - Complete package restructure
  - Nearly all functions renamed

## 1.4 License

Available under the MIT License

CHAPTER 2

# Commands

Chakin is a set of wrappers for accessing Chado. Each utility is implemented as a subcommand of `chakin`. This section of the documentation describes these commands.

## 2.1 analysis

This section is auto-generated from the help text for the chakin command `analysis`.

### 2.1.1 `add_analysis` command

**Usage**:

```
chakin analysis add_analysis [OPTIONS] NAME PROGRAM PROGRAMVERSION
```

**Help**

Create an analysis

**Output**

Analysis information

**Options**:

```
--algorithm TEXT      analysis algorithm
--sourceversion TEXT  analysis sourceversion
--sourceuri TEXT      analysis sourceuri
--description TEXT    analysis description
--date_executed TEXT  analysis date_executed (yyyy-mm-dd)
-h, --help            Show this message and exit.
```

### 2.1.2 `delete_analyses` command

**Usage**:

```
chakin analysis delete_analyses [OPTIONS]
```

**Help**

Delete analysis

**Output**

None

**Options**:

```
--analysis_id INTEGER  analysis_id filter
--name TEXT            analysis name filter
--program TEXT         analysis program filter
--programversion TEXT  analysis programversion filter
--algorithm TEXT       analysis algorithm filter
--sourcename TEXT      analysis sourcename filter
--sourceversion TEXT   analysis sourceversion filter
--sourceuri TEXT       analysis sourceuri filter
--description TEXT     analysis description
-h, --help             Show this message and exit.
```

### 2.1.3 `get_analyses` command

**Usage**:

```
chakin analysis get_analyses [OPTIONS]
```

**Help**

Get all or some analyses

**Output**

Analysis information

**Options**:

```
--analysis_id INTEGER  analysis_id filter
--name TEXT            analysis name filter
--program TEXT         analysis program filter
--programversion TEXT  analysis programversion filter
--algorithm TEXT       analysis algorithm filter
--sourcename TEXT      analysis sourcename filter
--sourceversion TEXT   analysis sourceversion filter
--sourceuri TEXT       analysis sourceuri filter
--description TEXT     analysis description
-h, --help             Show this message and exit.
```

## 2.2 export

This section is auto-generated from the help text for the chakin command `export`.

### 2.2.1 `export_fasta` command

**Usage**:

```
chakin export export_fasta [OPTIONS] ORGANISM_ID
```

**Help**

Export reference sequences as fasta.

**Output**

None

**Options**:

```
--file      If true, write to files in CWD
-h, --help  Show this message and exit.
```

### 2.2.2 `export_gbk` command

**Usage**:

```
chakin export export_gbk [OPTIONS] ORGANISM_ID
```

**Help**

Export organism features as genbank

**Output**

None

**Options**:

```
-h, --help  Show this message and exit.
```

### 2.2.3 `export_gff3` command

**Usage**:

```
chakin export export_gff3 [OPTIONS] ORGANISM_ID
```

**Help**

Export organism features as GFF3

**Output**

None

**Options**:

```
-h, --help  Show this message and exit.
```

## 2.3 expression

This section is auto-generated from the help text for the chakin command `expression`.

### 2.3.1 `add_biomaterial` command

**Usage**:

```
chakin expression add_biomaterial [OPTIONS] BIOMATERIAL_NAME ORGANISM_ID
```

**Help**

Add a new biomaterial to the database

**Output**

Biomaterial details

**Options**:

```
--description TEXT            Description of the biomaterial
--biomaterial_provider TEXT   Biomaterial provider name
--biosample_accession TEXT    Biosample accession number
--sra_accession TEXT          SRA accession number
--bioproject_accession TEXT   Bioproject accession number
--attributes TEXT             Custom attributes (In JSON dict form)
-h, --help                    Show this message and exit.
```

### 2.3.2 `add_expression` command

**Usage**:

```
chakin expression add_expression [OPTIONS] ORGANISM_ID ANALYSIS_ID
```

**Help**

Add an expression matrix file to the database

**Output**

Number of expression data loaded

**Options**:

```
--separator TEXT   Separating character in the matrix file (ex : ','). Default
                   character is tab.  [default:        ]
--unit TEXT        The units associated with the loaded values (ie, FPKM,
                   RPKM, raw counts)
--query_type TEXT  The feature type (e.g. 'gene', 'mRNA', 'polypeptide',
                   'contig') of the query. It must be a valid Sequence
                   Ontology term.  [default: mRNA]
--match_on_name    Match features using their name instead of their uniquename
--re_name TEXT     Regular expression to extract the feature name from the
                   input file (first capturing group will be used).
--skip_missing     Skip lines with unknown features or GO id instead of
                   aborting everything.
-h, --help         Show this message and exit.
```

### 2.3.3 `delete_all_biomaterials` command

**Usage**:

```
chakin expression delete_all_biomaterials [OPTIONS]
```

**Help**

Delete all biomaterials

**Output**

None

**Options**:

```
--confirm   Confirm that you really do want to delete ALL of the biomaterials.
-h, --help  Show this message and exit.
```

### 2.3.4 `delete_biomaterial` command

**Usage**:

```
chakin expression delete_biomaterial [OPTIONS]
```

**Help**

**Output**

I have no idea

**Options**:

```
--names TEXT         JSON list of biomaterial names to delete.  [default: []]
--ids TEXT           JSON list of biomaterial ids to delete.  [default: []]
--organism_id TEXT   Delete all biomaterial associated with this organism id.
--analysis_id TEXT   Delete all biomaterial associated with this analysis id.
-h, --help           Show this message and exit.
```

### 2.3.5 `delete_biomaterials` command

**Usage**:

```
chakin expression delete_biomaterials [OPTIONS]
```

**Help**

Will delete biomaterials based on selector. Only one selector will be used.

**Output**

Number of deleted biomaterials

**Options**:

```
--names TEXT            JSON list of biomaterial names to delete.
--ids TEXT              JSON list of biomaterial ids to delete.
--organism_id INTEGER   Delete all biomaterial associated with this organism
                        id.
--analysis_id INTEGER   Delete all biomaterial associated with this analysis
                        id.
-h, --help              Show this message and exit.
```

### 2.3.6 `get_biomaterials` command

**Usage**:

```
chakin expression get_biomaterials [OPTIONS]
```

**Help**

List biomaterials in the database

**Output**

List of biomaterials

**Options**:

```
--provider_id INTEGER     Limit query to the selected provider
--biomaterial_id INTEGER  Limit query to the selected biomaterial id
--organism_id INTEGER     Limit query to the selected organism
--biomaterial_name TEXT   Limit query to the selected biomaterial name
--analysis_id INTEGER     Limit query to the selected analysis_id
-h, --help                Show this message and exit.
```

## 2.4 feature

This section is auto-generated from the help text for the chakin command `feature`.

### 2.4.1 `delete_features` command

**Usage**:

```
chakin feature delete_features [OPTIONS]
```

**Help**

Get all or some features

**Output**

Features information

**Options**:

```
--organism_id INTEGER  organism_id filter
--analysis_id INTEGER  analysis_id filter
--name TEXT            name filter
```

(continues on next page)

(continued from previous page)

```
--uniquename TEXT      uniquename filter
-h, --help             Show this message and exit.
```

### 2.4.2 `get_feature_analyses` command

**Usage**:

```
chakin feature get_feature_analyses [OPTIONS] FEATURE_ID
```

**Help**

Get analyses associated with a feature

**Output**

Feature analyses

**Options**:

```
-h, --help  Show this message and exit.
```

### 2.4.3 `get_feature_cvterms` command

**Usage**:

```
chakin feature get_feature_cvterms [OPTIONS] FEATURE_ID
```

**Help**

Get cvterms associated with a feature

**Output**

Feature cvterms

**Options**:

```
-h, --help  Show this message and exit.
```

### 2.4.4 `get_features` command

**Usage**:

```
chakin feature get_features [OPTIONS]
```

**Help**

Get all or some features

**Output**

Features information

**Options**:

```
--organism_id INTEGER  organism_id filter
--analysis_id INTEGER  analysis_id filter
--name TEXT            name filter
--uniquename TEXT      uniquename filter
-h, --help             Show this message and exit.
```

### 2.4.5 `load_fasta` command

**Usage**:

```
chakin feature load_fasta [OPTIONS] FASTA ORGANISM_ID
```

**Help**

Load features from a fasta file

**Output**

Number of inserted sequences

**Options**:

```
--sequence_type TEXT    Sequence type  [default: contig]
--analysis_id INTEGER   Analysis ID
--re_name TEXT          Regular expression to extract the feature name from
                        the fasta sequence id (first capturing group will be
                        used).
--re_uniquename TEXT    Regular expression to extract the feature name from
                        the fasta sequence id (first capturing group will be
                        used).
--match_on_name         Match existing features using their name instead of
                        their uniquename
--update                Update existing feature with new sequence instead of
                        throwing an error
--db INTEGER            External database to cross reference to.
--re_db_accession TEXT  Regular expression to extract an external database
                        accession from the fasta sequence id (first capturing
                        group will be used).
--rel_type TEXT         Relation type to parent feature ('part_of' or
                        'derives_from').
--re_parent TEXT        Regular expression to extract parent uniquename from
                        the fasta sequence id (first capturing group will be
                        used).
--parent_type TEXT      Sequence type of the parent feature
-h, --help              Show this message and exit.
```

### 2.4.6 `load_featureprops` command

**Usage**:

```
chakin feature load_featureprops [OPTIONS] TAB_FILE ANALYSIS_ID
```

**Help**

Load feature properties from a tabular file (Column1: feature name or uniquename, Column2: property value)

**Output**

Number of inserted featureprop

**Options**:

```
--feature_type TEXT  Type of the target features in sequence ontology (will
                     speed up loading if specified)
--match_on_name      Match features using their name instead of their
                     uniquename
-h, --help           Show this message and exit.
```

### 2.4.7 `load_gff` command

**Usage**:

```
chakin feature load_gff [OPTIONS] GFF ANALYSIS_ID ORGANISM_ID
```

**Help**

Load features from a gff file

**Output**

None

**Options**:

```
--landmark_type TEXT       Type of the landmarks (will speed up loading if
                           provided, e.g. contig, should be a term of the
                           Sequence ontology)
--re_protein TEXT          Replacement string for the protein name using
                           capturing groups defined by --re_protein_capture
--re_protein_capture TEXT  Regular expression to capture groups in mRNA name
                           to use in --re_protein (e.g. "^(.*?)-R([A-Z]+)$",
                           default="^(.*?)$")  [default: ^(.*?)$]
--fasta TEXT               Path to a Fasta containing sequences for some
                           features. When creating a feature, if its sequence
                           is in this fasta file it will be loaded. Otherwise
                           for mRNA and polypeptides it will be computed from
                           the genome sequence (if available), otherwise it
                           will be left empty.
--no_seq_compute           Disable the computation of mRNA and polypeptides
                           sequences based on genome sequence and positions.
--quiet                    Hide progress information
--add_only                 Use this flag if you're not updating existing
                           features, but just adding new features to the
                           selected analysis and organism. It will speedup
                           loading, and reduce memory usage, but might produce
                           errors in case of already existing feature.
--protein_id_attr TEXT     Attribute containing the protein uniquename. It is
                           searched at the mRNA level, and if not found at CDS
                           level.
-h, --help                 Show this message and exit.
```

### 2.4.8 `load_go` command

**Usage**:

```
chakin feature load_go [OPTIONS] INPUT ORGANISM_ID ANALYSIS_ID
```

**Help**

Load GO annotation from a tabular file

**Output**

Number of inserted GO terms

**Options**:

```
--query_type TEXT      The feature type (e.g. 'gene', 'mRNA', 'polypeptide',
                       'contig') of the query. It must be a valid Sequence
                       Ontology term.  [default: polypeptide]
--match_on_name        Match features using their name instead of their
                       uniquename
--name_column INTEGER  Column containing the feature identifiers (2, 3, 10 or
                       11; default=2).  [default: 2]
--go_column INTEGER    Column containing the GO id (default=5).  [default: 5]
--re_name TEXT         Regular expression to extract the feature name from the
                       input file (first capturing group will be used).
--skip_missing         Skip lines with unknown features or GO id instead of
                       aborting everything.
-h, --help             Show this message and exit.
```

## 2.5 load

This section is auto-generated from the help text for the chakin command `load`.

### 2.5.1 `blast` command

**Usage**:

```
chakin load blast [OPTIONS] ANALYSIS_ID ORGANISM_ID INPUT
```

**Help**

Load a blast analysis, in the same way as does the tripal_analysis_blast module

**Output**

Number of processed hits

**Options**:

```
--blastdb TEXT         Name of the database blasted against (must be in the
                       Chado db table)
--blastdb_id INTEGER   ID of the database blasted against (must be in the Chado
                       db table)
--re_name TEXT         Regular expression to extract the feature name from the
                       input file (first capturing group will be used).
--query_type TEXT      The feature type (e.g. 'gene', 'mRNA', 'polypeptide',
                       'contig') of the query. It must be a valid Sequence
                       Ontology term.  [default: polypeptide]
--match_on_name        Match features using their name instead of their
```

(continues on next page)

(continued from previous page)

```
                        uniquename
--skip_missing          Skip lines with unknown features or GO id instead of
                        aborting everything.
-h, --help              Show this message and exit.
```

### 2.5.2 `go` command

**Usage**:

```
chakin load go [OPTIONS] INPUT ORGANISM_ID ANALYSIS_ID
```

**Help**

Load GO annotation from a tabular file, in the same way as does the tripal_analysis_go module

**Output**

Number of inserted GO terms

**Options**:

```
--query_type TEXT       The feature type (e.g. 'gene', 'mRNA', 'polypeptide',
                        'contig') of the query. It must be a valid Sequence
                        Ontology term.  [default: polypeptide]
--match_on_name         Match features using their name instead of their
                        uniquename
--name_column INTEGER   Column containing the feature identifiers (2, 3, 10 or
                        11; default=2).  [default: 2]
--go_column INTEGER     Column containing the GO id (default=5).  [default: 5]
--re_name TEXT          Regular expression to extract the feature name from the
                        input file (first capturing group will be used).
--skip_missing          Skip lines with unknown features or GO id instead of
                        aborting everything.
-h, --help              Show this message and exit.
```

### 2.5.3 `interpro` command

**Usage**:

```
chakin load interpro [OPTIONS] ANALYSIS_ID ORGANISM_ID INPUT
```

**Help**

Load an InterProScan analysis, in the same way as does the tripal_analysis_intepro module

**Output**

Number of processed hits

**Options**:

```
--parse_go          Load GO annotation to the database
--re_name TEXT      Regular expression to extract the feature name from the
                    input file (first capturing group will be used).
--query_type TEXT   The feature type (e.g. 'gene', 'mRNA', 'polypeptide',
                    'contig') of the query. It must be a valid Sequence
```

(continues on next page)

(continued from previous page)

```
                    Ontology term.  [default: polypeptide]
--match_on_name     Match features using their name instead of their uniquename
--skip_missing      Skip lines with unknown features or GO id instead of
                    aborting everything.
-h, --help          Show this message and exit.
```

## 2.6 organism

This section is auto-generated from the help text for the chakin command `organism`.

### 2.6.1 `add_organism` command

**Usage**:

```
chakin organism add_organism [OPTIONS] GENUS SPECIES COMMON ABBR
```

**Help**

Add a new organism to the Chado database

**Output**

Organism information

**Options**:

```
--comment TEXT  A comment / description
-h, --help      Show this message and exit.
```

### 2.6.2 `delete_all_organisms` command

**Usage**:

```
chakin organism delete_all_organisms [OPTIONS]
```

**Help**

Delete all organisms

**Output**

None

**Options**:

```
--confirm   Confirm that you really do want to delete ALL of the organisms.
-h, --help  Show this message and exit.
```

### 2.6.3 `delete_organisms` command

**Usage**:

```
chakin organism delete_organisms [OPTIONS]
```

**Help**

Delete all organisms

**Output**

None

**Options**:

```
--organism_id INTEGER  organism_id filter
--genus TEXT           genus filter
--species TEXT         species filter
--common TEXT          common filter
--abbr TEXT            abbr filter
--comment TEXT         comment filter
-h, --help             Show this message and exit.
```

### 2.6.4 `get_organisms` command

**Usage**:

```
chakin organism get_organisms [OPTIONS]
```

**Help**

Get all or some organisms

**Output**

Organisms information

**Options**:

```
--organism_id INTEGER  organism_id filter
--genus TEXT           genus filter
--species TEXT         species filter
--common TEXT          common filter
--abbr TEXT            abbr filter
--comment TEXT         comment filter
-h, --help             Show this message and exit.
```

## 2.7 phylogeny

This section is auto-generated from the help text for the chakin command `phylogeny`.

### 2.7.1 `add_cvterms` command

**Usage**:

```
chakin phylogeny add_cvterms [OPTIONS]
```

**Help**

Make sure required cvterms are loaded

**Output**

created cvterms

**Options**:

```
-h, --help  Show this message and exit.
```

### 2.7.2 `gene_families` command

**Usage**:

```
chakin phylogeny gene_families [OPTIONS]
```

**Help**

Adds an entry in the featureprop table in a chado database for each family a gene belongs to (for use in https://github.com/legumeinfo/lis_context_viewer/).

**Output**

None

**Options**:

```
--family_name TEXT  Restrict to families beginning with given prefix
--nuke              Removes all previous gene families data
-h, --help          Show this message and exit.
```

### 2.7.3 `gene_order` command

**Usage**:

```
chakin phylogeny gene_order [OPTIONS]
```

**Help**

Orders all the genes in the database by their order on their respective chromosomes in the gene_order table (for use in https://github.com/legumeinfo/lis_context_viewer/).

**Output**

None

**Options**:

```
--nuke      Removes all previous gene ordering data
-h, --help  Show this message and exit.
```

### 2.7.4 `load_tree` command

**Usage**:

```
chakin phylogeny load_tree [OPTIONS] NEWICK ANALYSIS_ID
```

**Help**

Load a phylogenetic tree (Newick format) into Chado db

**Output**

Number of inserted trees

**Options**:

```
--name TEXT             The name given to the phylotree entry in the database
                        (default=<filename>)
--xref_db TEXT          The name of the db to link dbxrefs for the trees
                        (default: "null")  [default: null]
--xref_accession TEXT   The accession to use for dbxrefs for the trees (assumed
                        same as name unless otherwise specified)
--match_on_name         Match polypeptide features using their name instead of
                        their uniquename
--prefix TEXT           Comma-separated list of prefix to be removed from
                        identifiers (e.g species prefixes when loading
                        OrthoFinder output)
-h, --help              Show this message and exit.
```

## 2.8 util

This section is auto-generated from the help text for the chakin command `util`.

### 2.8.1 `dbshell` command

**Usage**:

```
chakin util dbshell [OPTIONS]
```

**Help**

Open a psql session to the database

**Output**

None

**Options**:

```
-h, --help  Show this message and exit.
```

### 2.8.2 `launch_docker_image` command

**Usage**:

```
chakin util launch_docker_image [OPTIONS]
```

**Help**

Launch a chado docker image.

**Output**

None

**Options**:

```
--background  Launch the image in the background
--no_yeast    Disable loading of example yeast data
-h, --help    Show this message and exit.
```

CHAPTER 3

# chado package

## 3.1 Subpackages

### 3.1.1 chado.analysis package

#### Module contents

Contains possible interactions with the Chado Analysis Module

**class** chado.analysis.**AnalysisClient**(*engine*, *metadata*, *session*, *ci*)

Bases: *chado.client.Client*

Access to the chado analysis table

**add_analysis**(*name*, *program*, *programversion*, *sourcename*, *algorithm=None*, *sourceversion=None*, *sourceuri=None*, *description=None*, *date_executed=None*)

Create an analysis

**Parameters**

- **name** (*str*) – analysis name
- **program** (*str*) – analysis program
- **programversion** (*str*) – analysis programversion
- **algorithm** (*str*) – analysis algorithm
- **sourcename** (*str*) – analysis sourcename
- **sourceversion** (*str*) – analysis sourceversion
- **sourceuri** (*str*) – analysis sourceuri
- **description** (*str*) – analysis description
- **date_executed** (*str*) – analysis date_executed (yyyy-mm-dd)

**Return type** dict

**Returns** Analysis information

**delete_analyses**(*analysis_id=None*, *name=None*, *program=None*, *programversion=None*, *algorithm=None*, *sourcename=None*, *sourceversion=None*, *sourceuri=None*, *description=None*)

Delete analysis

**Parameters**

- **analysis_id** (*int*) – analysis_id filter
- **name** (*str*) – analysis name filter
- **program** (*str*) – analysis program filter
- **programversion** (*str*) – analysis programversion filter
- **algorithm** (*str*) – analysis algorithm filter
- **sourcename** (*str*) – analysis sourcename filter
- **sourceversion** (*str*) – analysis sourceversion filter
- **sourceuri** (*str*) – analysis sourceuri filter
- **description** (*str*) – analysis description

**Return type** None

**Returns** None

**get_analyses**(*analysis_id=None*, *name=None*, *program=None*, *programversion=None*, *algorithm=None*, *sourcename=None*, *sourceversion=None*, *sourceuri=None*, *description=None*)

Get all or some analyses

**Parameters**

- **analysis_id** (*int*) – analysis_id filter
- **name** (*str*) – analysis name filter
- **program** (*str*) – analysis program filter
- **programversion** (*str*) – analysis programversion filter
- **algorithm** (*str*) – analysis algorithm filter
- **sourcename** (*str*) – analysis sourcename filter
- **sourceversion** (*str*) – analysis sourceversion filter
- **sourceuri** (*str*) – analysis sourceuri filter
- **description** (*str*) – analysis description

**Return type** list of dict

**Returns** Analysis information

### 3.1.2 chado.export package

#### Module contents

Export data from chado

**class** chado.export.**ExportClient**(*engine*, *metadata*, *session*, *ci*)
Bases: *chado.client.Client*

Export data from the chado database

**export_fasta**(*organism_id*, *file=False*)
Export reference sequences as fasta.

**Parameters**

- **organism_id** (*int*) – Organism ID
- **file** (*bool*) – If true, write to files in CWD

**Return type** None

**Returns** None

**export_gbk**(*organism_id*)
Export organism features as genbank

**Parameters** **organism_id** (*int*) – Organism ID

**Return type** None

**Returns** None

**export_gff3**(*organism_id*)
Export organism features as GFF3

**Parameters** **organism_id** (*int*) – Organism ID

**Return type** None

**Returns** None

### 3.1.3 chado.expression package

#### Module contents

Contains possible interactions with the Chado base for expressions and biomaterials

**class** chado.expression.**ExpressionClient**(*engine*, *metadata*, *session*, *ci*)
Bases: *chado.client.Client*

Interact with expressions

**add_biomaterial**(*biomaterial_name*, *organism_id*, *description=''*, *biomaterial_provider=''*, *biosample_accession=''*, *sra_accession=''*, *bioproject_accession=''*, *attributes={}*)
Add a new biomaterial to the database

**Parameters**

- **biomaterial_name** (*str*) – Biomaterial name
- **organism_id** (*int*) – The id of the associated organism
- **description** (*str*) – Description of the biomaterial
- **biomaterial_provider** (*str*) – Biomaterial provider name
- **biosample_accession** (*str*) – Biosample accession number
- **sra_accession** (*str*) – SRA accession number

- **bioproject_accession** (*str*) – Bioproject accession number
- **attributes** (*str*) – Custom attributes (In JSON dict form)

**Return type** dict

**Returns** Biomaterial details

**add_expression**(*organism_id*, *analysis_id*, *file_path*, *separator='\t'*, *unit=None*, *query_type='mRNA'*, *match_on_name=False*, *re_name=None*, *skip_missing=False*)

Add an expression matrix file to the database

**Parameters**

- **organism_id** (*int*) – The id of the associated organism
- **analysis_id** (*int*) – The id of the associated analysis
- **file_path** (*str*) – File path
- **separator** (*str*) – Separating character in the matrix file (ex : ',’). Default character is tab.
- **unit** (*str*) – The units associated with the loaded values (ie, FPKM, RPKM, raw counts)
- **query_type** (*str*) – The feature type (e.g. ‘gene’, ‘mRNA’, ‘polypeptide’, ‘contig’) of the query. It must be a valid Sequence Ontology term.
- **match_on_name** (*bool*) – Match features using their name instead of their uniquename
- **re_name** (*str*) – Regular expression to extract the feature name from the input file (first capturing group will be used).
- **skip_missing** (*bool*) – Skip lines with unknown features or GO id instead of aborting everything.

**Return type** str

**Returns** Number of expression data loaded

**delete_all_biomaterials**(*confirm=False*)

Delete all biomaterials

**Parameters** **confirm** (*bool*) – Confirm that you really do want to delete ALL of the biomaterials.

**Return type** None

**Returns** None

**delete_biomaterials**(*names=None*, *ids=None*, *organism_id=''*, *analysis_id=''*)

Will delete biomaterials based on selector. Only one selector will be used.

**Parameters**

- **names** (*str*) – JSON list of biomaterial names to delete.
- **ids** (*str*) – JSON list of biomaterial ids to delete.
- **organism_id** (*int*) – Delete all biomaterial associated with this organism id.
- **analysis_id** (*int*) – Delete all biomaterial associated with this analysis id.

**Return type** str

**Returns** Number of deleted biomaterials

**get_biomaterials**(*provider_id=''*, *biomaterial_id=''*, *organism_id=''*, *biomaterial_name=''*, *analysis_id=''*)
List biomaterials in the database

**Parameters**

- **organism_id** (*int*) – Limit query to the selected organism
- **biomaterial_id** (*int*) – Limit query to the selected biomaterial id
- **provider_id** (*int*) – Limit query to the selected provider
- **biomaterial_name** (*str*) – Limit query to the selected biomaterial name
- **analysis_id** (*int*) – Limit query to the selected analysis_id

**Return type** list

**Returns** List of biomaterials

### 3.1.4 chado.feature package

#### Module contents

Contains possible interactions with the Chado Features

**class** chado.feature.**FeatureClient**(*engine*, *metadata*, *session*, *ci*)
Bases: *chado.client.Client*

Access to the chado features

**delete_features**(*organism_id=None*, *analysis_id=None*, *name=None*, *uniquename=None*)
Get all or some features

**Parameters**

- **organism_id** (*int*) – organism_id filter
- **analysis_id** (*int*) – analysis_id filter
- **name** (*str*) – name filter
- **uniquename** (*str*) – uniquename filter

**Return type** list of dict

**Returns** Features information

**get_feature_analyses**(*feature_id*)
Get analyses associated with a feature

**Parameters** **feature_id** (*int*) – Id of the feature

**Return type** list

**Returns** Feature analyses

**get_feature_cvterms**(*feature_id*)
Get cvterms associated with a feature

**Parameters** **feature_id** (*int*) – Id of the feature

**Return type** list

**Returns** Feature cvterms

**get_features**(*organism_id=None*, *analysis_id=None*, *name=None*, *uniquename=None*)
Get all or some features

**Parameters**

- **organism_id** (*int*) – organism_id filter
- **analysis_id** (*int*) – analysis_id filter
- **name** (*str*) – name filter
- **uniquename** (*str*) – uniquename filter

**Return type** list of dict

**Returns** Features information

**load_fasta**(*fasta*, *organism_id*, *sequence_type='contig'*, *analysis_id=None*, *re_name=None*, *re_uniquename=None*, *match_on_name=False*, *update=False*, *db=None*, *re_db_accession=None*, *rel_type=None*, *re_parent=None*, *parent_type=None*)
Load features from a fasta file

**Parameters**

- **fasta** (*str*) – Path to the Fasta file to load
- **organism_id** (*int*) – Organism ID
- **sequence_type** (*str*) – Sequence type
- **analysis_id** (*int*) – Analysis ID
- **re_name** (*str*) – Regular expression to extract the feature name from the fasta sequence id (first capturing group will be used).
- **re_uniquename** (*str*) – Regular expression to extract the feature name from the fasta sequence id (first capturing group will be used).
- **match_on_name** (*bool*) – Match existing features using their name instead of their uniquename
- **update** (*bool*) – Update existing feature with new sequence instead of throwing an error
- **db** (*int*) – External database to cross reference to.
- **re_db_accession** (*str*) – Regular expression to extract an external database accession from the fasta sequence id (first capturing group will be used).
- **rel_type** (*str*) – Relation type to parent feature ('part_of' or 'derives_from').
- **re_parent** (*str*) – Regular expression to extract parent uniquename from the fasta sequence id (first capturing group will be used).
- **parent_type** (*str*) – Sequence type of the parent feature

**Return type** dict

**Returns** Number of inserted sequences

**load_featureprops**(*tab_file*, *analysis_id*, *organism_id*, *prop_type*, *feature_type=None*, *match_on_name=False*)
Load feature properties from a tabular file (Column1: feature name or uniquename, Column2: property value)

**Parameters**

- **tab_file** (*str*) – Path to the tabular file to load

- **analysis_id** (*int*) – Analysis ID
- **organism_id** (*int*) – Organism ID
- **prop_type** (*str*) – Type of the feature property (cvterm will be created if it doesn't exist)
- **feature_type** (*str*) – Type of the target features in sequence ontology (will speed up loading if specified)
- **match_on_name** (*bool*) – Match features using their name instead of their uniquename

**Return type** dict

**Returns** Number of inserted featureprop

**load_gff**(*gff*, *analysis_id*, *organism_id*, *landmark_type=None*, *re_protein=None*, *re_protein_capture='^(.*?)$'*, *fasta=None*, *no_seq_compute=False*, *quiet=False*, *add_only=False*, *protein_id_attr=None*)

Load features from a gff file

**Parameters**

- **gff** (*str*) – Path to the Fasta file to load
- **analysis_id** (*int*) – Analysis ID
- **organism_id** (*int*) – Organism ID
- **landmark_type** (*str*) – Type of the landmarks (will speed up loading if provided, e.g. contig, should be a term of the Sequence ontology)
- **re_protein** (*str*) – Replacement string for the protein name using capturing groups defined by –re_protein_capture
- **re_protein_capture** (*str*) – Regular expression to capture groups in mRNA name to use in –re_protein (e.g. "^(.*?)-R([A-Z]+)$", default="^(.*?)$")
- **protein_id_attr** (*str*) – Attribute containing the protein uniquename. It is searched at the mRNA level, and if not found at CDS level.
- **fasta** (*str*) – Path to a Fasta containing sequences for some features. When creating a feature, if its sequence is in this fasta file it will be loaded. Otherwise for mRNA and polypeptides it will be computed from the genome sequence (if available), otherwise it will be left empty.
- **no_seq_compute** (*bool*) – Disable the computation of mRNA and polypeptides sequences based on genome sequence and positions.
- **quiet** (*bool*) – Hide progress information
- **add_only** (*bool*) – Use this flag if you're not updating existing features, but just adding new features to the selected analysis and organism. It will speedup loading, and reduce memory usage, but might produce errors in case of already existing feature.

**Return type** None

**Returns** None

**load_go**(*input*, *organism_id*, *analysis_id*, *query_type='polypeptide'*, *match_on_name=False*, *name_column=2*, *go_column=5*, *re_name=None*, *skip_missing=False*)

Load GO annotation from a tabular file

**Parameters**

- **input** (*str*) – Path to the input tabular file to load

- **organism_id** (*int*) – Organism ID
- **analysis_id** (*int*) – Analysis ID
- **query_type** (*str*) – The feature type (e.g. 'gene', 'mRNA', 'polypeptide', 'contig') of the query. It must be a valid Sequence Ontology term.
- **match_on_name** (*bool*) – Match features using their name instead of their uniquename
- **name_column** (*int*) – Column containing the feature identifiers (2, 3, 10 or 11; default=2).
- **go_column** (*int*) – Column containing the GO id (default=5).
- **re_name** (*str*) – Regular expression to extract the feature name from the input file (first capturing group will be used).
- **skip_missing** (*bool*) – Skip lines with unknown features or GO id instead of aborting everything.

**Return type** dict

**Returns** Number of inserted GO terms

### 3.1.5 chado.load package

#### Module contents

Contains loader methods

**class** chado.load.**LoadClient**(*engine*, *metadata*, *session*, *ci*)

Bases: *chado.client.Client*

**blast**(*analysis_id*, *organism_id*, *input*, *blastdb=None*, *blastdb_id=None*, *re_name=None*, *query_type='polypeptide'*, *match_on_name=False*, *skip_missing=False*)

Load a blast analysis, in the same way as does the tripal_analysis_blast module

**Parameters**

- **analysis_id** (*int*) – Analysis ID
- **organism_id** (*int*) – Organism ID
- **input** (*str*) – Path to the Blast XML file to load
- **blastdb** (*str*) – Name of the database blasted against (must be in the Chado db table)
- **blastdb_id** (*int*) – ID of the database blasted against (must be in the Chado db table)
- **query_type** (*str*) – The feature type (e.g. 'gene', 'mRNA', 'polypeptide', 'contig') of the query. It must be a valid Sequence Ontology term.
- **match_on_name** (*bool*) – Match features using their name instead of their uniquename
- **re_name** (*str*) – Regular expression to extract the feature name from the input file (first capturing group will be used).
- **skip_missing** (*bool*) – Skip lines with unknown features or GO id instead of aborting everything.

**Return type** dict

**Returns** Number of processed hits

**go**(*input*, *organism_id*, *analysis_id*, *query_type='polypeptide'*, *match_on_name=False*, *name_column=2*, *go_column=5*, *re_name=None*, *skip_missing=False*)
Load GO annotation from a tabular file, in the same way as does the tripal_analysis_go module

**Parameters**

- **input** (*str*) – Path to the input tabular file to load
- **organism_id** (*int*) – Organism ID
- **analysis_id** (*int*) – Analysis ID
- **query_type** (*str*) – The feature type (e.g. 'gene', 'mRNA', 'polypeptide', 'contig') of the query. It must be a valid Sequence Ontology term.
- **match_on_name** (*bool*) – Match features using their name instead of their uniquename
- **name_column** (*int*) – Column containing the feature identifiers (2, 3, 10 or 11; default=2).
- **go_column** (*int*) – Column containing the GO id (default=5).
- **re_name** (*str*) – Regular expression to extract the feature name from the input file (first capturing group will be used).
- **skip_missing** (*bool*) – Skip lines with unknown features or GO id instead of aborting everything.

**Return type** dict

**Returns** Number of inserted GO terms

**interpro**(*analysis_id*, *organism_id*, *input*, *parse_go=False*, *re_name=None*, *query_type='polypeptide'*, *match_on_name=False*, *skip_missing=False*)
Load an InterProScan analysis, in the same way as does the tripal_analysis_intepro module

**Parameters**

- **analysis_id** (*int*) – Analysis ID
- **organism_id** (*int*) – Organism ID
- **input** (*str*) – Path to the InterProScan file to load
- **parse_go** (*bool*) – Load GO annotation to the database
- **query_type** (*str*) – The feature type (e.g. 'gene', 'mRNA', 'polypeptide', 'contig') of the query. It must be a valid Sequence Ontology term.
- **match_on_name** (*bool*) – Match features using their name instead of their uniquename
- **re_name** (*str*) – Regular expression to extract the feature name from the input file (first capturing group will be used).
- **skip_missing** (*bool*) – Skip lines with unknown features or GO id instead of aborting everything.

**Return type** dict

**Returns** Number of processed hits

### 3.1.6 chado.organism package

### Module contents

Contains possible interactions with the Chado Organisms Module

**class** chado.organism.**OrganismClient**(*engine*, *metadata*, *session*, *ci*)

Bases: *chado.client.Client*

Access to the chado organism table

**add_organism**(*genus*, *species*, *common*, *abbr*, *comment=None*)

Add a new organism to the Chado database

**Parameters**

- **genus** (*str*) – The genus of the organism
- **species** (*str*) – The species of the organism
- **common** (*str*) – The common name of the organism
- **abbr** (*str*) – The abbreviation of the organism
- **comment** (*str*) – A comment / description

**Return type** dict

**Returns** Organism information

**delete_all_organisms**(*confirm=False*)

Delete all organisms

**Parameters** **confirm** (*bool*) – Confirm that you really do want to delete ALL of the organisms.

**Return type** None

**Returns** None

**delete_organisms**(*organism_id=None*, *genus=None*, *species=None*, *common=None*, *abbr=None*, *comment=None*)

Delete all organisms

**Parameters**

- **organism_id** (*int*) – organism_id filter
- **genus** (*str*) – genus filter
- **species** (*str*) – species filter
- **common** (*str*) – common filter
- **abbr** (*str*) – abbr filter
- **comment** (*str*) – comment filter

**Return type** None

**Returns** None

**get_organisms**(*organism_id=None*, *genus=None*, *species=None*, *common=None*, *abbr=None*, *comment=None*)

Get all or some organisms

**Parameters**

- **organism_id** (*int*) – organism_id filter
- **genus** (*str*) – genus filter

- **species** (*str*) – species filter
- **common** (*str*) – common filter
- **abbr** (*str*) – abbr filter
- **comment** (*str*) – comment filter

**Return type** list of dict

**Returns** Organisms information

### 3.1.7 chado.phylogeny package

#### Module contents

Contains possible interactions with the Chado Phylogeny Module http://gmod.org/wiki/Chado_Phylogeny_Module As implemented in https://github.com/legumeinfo/tripal_phylotree and https://github.com/legumeinfo/lis_context_viewer/

**class** chado.phylogeny.**PhylogenyClient**(*engine*, *metadata*, *session*, *ci*)

Bases: *chado.client.Client*

Access to the chado phylogeny content

**add_cvterms**()

Make sure required cvterms are loaded

**Return type** list of dict

**Returns** created cvterms

**gene_families**(*family_name=''*, *nuke=False*)

Adds an entry in the featureprop table in a chado database for each family a gene belongs to (for use in https://github.com/legumeinfo/lis_context_viewer/).

**Parameters**

- **family_name** (*str*) – Restrict to families beginning with given prefix
- **nuke** (*bool*) – Removes all previous gene families data

**Return type** None

**Returns** None

**gene_order**(*nuke=False*)

Orders all the genes in the database by their order on their respective chromosomes in the gene_order table (for use in https://github.com/legumeinfo/lis_context_viewer/).

**Parameters** **nuke** (*bool*) – Removes all previous gene ordering data

**Return type** None

**Returns** None

**load_tree**(*newick*, *analysis_id*, *name=None*, *xref_db='null'*, *xref_accession=None*, *match_on_name=False*, *prefix=''*)

Load a phylogenetic tree (Newick format) into Chado db

**Parameters**

- **newick** (*str*) – Newick file to load (or directory containing multiple newick files to load)

- **analysis_id** (*int*) – Analysis ID
- **name** (*str*) – The name given to the phylotree entry in the database (default=<filename>)
- **xref_db** (*str*) – The name of the db to link dbxrefs for the trees (default: "null")
- **xref_accession** (*str*) – The accession to use for dbxrefs for the trees (assumed same as name unless otherwise specified)
- **match_on_name** (*bool*) – Match polypeptide features using their name instead of their uniquename
- **prefix** (*str*) – Comma-separated list of prefix to be removed from identifiers (e.g species prefixes when loading OrthoFinder output)

**Return type** dict

**Returns** Number of inserted trees

### 3.1.8 chado.util package

#### Module contents

**class** chado.util.**UtilClient**(*engine*, *metadata*, *session*, *ci*)

Bases: *chado.client.Client*

Some chado utilities

**dbshell**()

Open a psql session to the database

**Return type** None

**Returns** None

**launch_docker_image**(*background=False*, *no_yeast=False*)

Launch a chado docker image.

**Parameters**

- **background** (*bool*) – Launch the image in the background
- **no_yeast** (*bool*) – Disable loading of example yeast data

**Return type** None

**Returns** None

## 3.2 Submodules

## 3.3 chado.client module

Base chado client

**class** chado.client.**Client**(*engine*, *metadata*, *session*, *ci*)

Bases: object

Base client class implementing methods to make queries to the server

## 3.4 chado.exceptions module

**exception** chado.exceptions.**RecordNotFoundError**
Bases: Exception

Raised when a db select failed.

## 3.5 chado.util module

**class** chado.util.**UtilClient**(*engine*, *metadata*, *session*, *ci*)
Bases: *chado.client.Client*

Some chado utilities

**dbshell**()
Open a psql session to the database

**Return type** None

**Returns** None

**launch_docker_image**(*background=False*, *no_yeast=False*)
Launch a chado docker image.

**Parameters**

- **background** (*bool*) – Launch the image in the background
- **no_yeast** (*bool*) – Disable loading of example yeast data

**Return type** None

**Returns** None

## 3.6 Module contents

**class** chado.**ChadoInstance**(*dbhost='localhost'*, *dbname='chado'*, *dbuser='chado'*, *dbpass='chado'*, *dbschema='public'*, *dbport=5432*, *dburl=None*, *offline=False*, *no_reflect=False*, *reflect_tripal_tables=False*, *pool_connections=True*, ***kwargs*)
Bases: object

**create_cvterm**(*term*, *cv_name*, *db_name*, *term_definition=''*, *cv_definition=''*, *db_definition=''*, *accession=''*)

**get_cvterm_id**(*name*, *cv*, *allow_synonyms=False*)
get_cvterm_id allows lookup of CV terms by their name. This method caches the result in order to not hit the DB for every query. Maybe should investigate pre-loading popular terms? (E.g. gene, mRNA, etc)

**get_cvterm_name**(*cv_id*)
get_cvterm_name allows lookup of CV terms by their ID. This method caches the result in order to not hit the DB for every query. Maybe should investigate pre-loading popular terms? (E.g. gene, mRNA, etc)

**get_pub_id**(*name*)
Allows lookup of publication by their uniquename. This method caches the result in order to not hit the DB for every query.

**class** chado.**ChadoModel**
Bases: object

CHAPTER 4

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## C

# Index

## I



## L



## O



## P



## R



## U