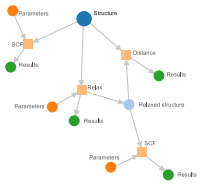
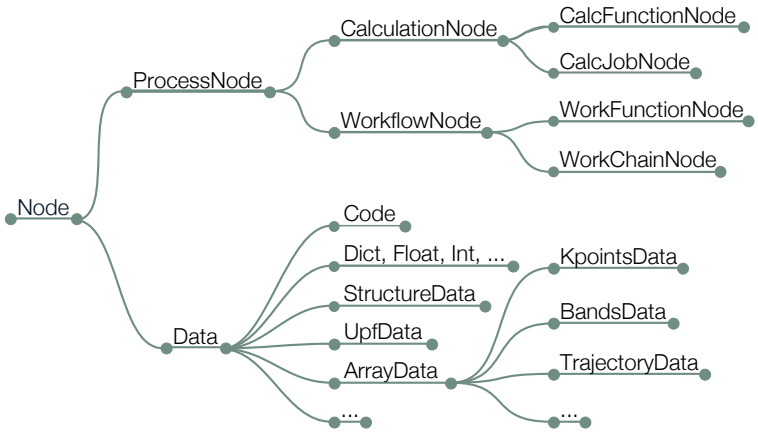
The AiiDA cheat sheet



The main AiiDA Node subclasses



To load an existing node: `load_node(<id>)`
Where `<id>` may be either the `pk`, `UUID`, or `label`

To load a class, either import it from `aiida.orm` or use the `DataFactory` (returning Data subclasses) or the `CalculationFactory` (returning CalcJobNode subclasses)

Importing classes

ORM and the Factories

Import `aiida-core` Node classes from `aiida.orm` using their class name:

```
from aiida.orm import CalcJobNode
from aiida.orm import Dict
```

Import `Data` classes via the `DataFactory` using `<label>`s::

```
KpointsData = DataFactory("array.kpoints")
MyData = DataFactory("plugin.my")
```

Other `<label>`s for `Data`:
`"upf", "array", "array.bands", "dict", ...`

Import `CalcJob` classes via the `CalculationFactory`:

```
PwCalculation =
    CalculationFactory("quantumespresso.pw")
```

Other `<label>`s for `Calculations`:
`"quantumespresso.ph", "vasp.scf", ...`

Import `WorkChain` classes via the `WorkflowFactory`.

Main attributes and methods

Note: each derived class inherits all the methods of the parent class

Node

`pk`	Node ID
`label`	Short label
`uuid`	Unique ID
`ctime`	Creation time
`mtime`	Modification time
`get_incoming()`	Get input
`get_outgoing()`	Get output
`inputs`	All inputs generator
`outputs`	All outputs generator
`attributes`	Queryable attributes
`get_attribute(k)`	Attribute 'k'
`extras`	Queryable extras
`get_extra(<k>)`	Extra 'k'
`set_extra(<k>,<v>)`	Set extra k = v
`get_comments()`	All comments
`add_comment(<c>)`	Add comment with content <c>
`store()`	Save node in DB

Code

`load_code(<id>)`	Load code using `pk`, `UUID`, or `label`
`get_builder()`	Return new builder using this code

Data

`export()`	Export to file
`_exportcontent()`	Export to string
`importfile()`	Import from file
`importstring()`	Import from string

StructureData

`cell`	Lattice vectors
`sites`	Atomic sites
`kinds`	Species with masses, symbols, ...
`pbc`	Periodic bound. cond. along each axis
`get_formula()`	Chemical formula
`get_cell_volume()`	Compute cell volume
`convert(<fmt>)`	Convert to ASE, pymatgen, ...
`set_cell(<c>)`	Set lattice vectors
`set_ase(<a>)`	Create cell from ASE
`set_pymatgen(<p>)`	Create cell from pymatgen
`append_atom(symbols=<symb>, position=<p>)`	Add atom of type 'symb' at position 'p'

Dict

`dict.<k>`	Get value for key 'k'
`keys()`	Get all keys generator
`get_dict()`	Get all key/values
`set_dict(<dict>)`	Replace all key/values

ArrayData

`get_arraynames()`	Names of all arrays
`get_array(<n>)`	Get array named 'n'
`set_array(<n>,<a>)`	Set/store array 'a' with name 'n'

KpointsData

`set_kpoints(<k>)`	Set an explicit list of kpoints 'k' (optionally with weights)
`get_kpoints()`	Get explicit list of kpts (if stored explicitly)
`set_kpoints_mesh(<m>)`	Set an implicit mesh (e.g. 'm'=3x2x5)
`get_kpoints_mesh()`	Get the implicit mesh (if stored implicitly)

CalcJobNode

`process_state`	Calc. process state
`exit_status`	Exit status or int code
`is_finished`	Has calc. finished?
`is_failed`	Has calc. failed?
`computer`	Computer where it is running
`inputs.code`	Code used to run
`get_job_id()`	Scheduler job ID
`get_options()`	Get # machines, MPI procs per machine, ...
`res.<k>`	Value of parsed output 'k'

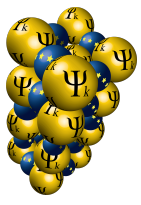
Useful links:

Tutorial website:
`aiida-tutorials.readthedocs.io`

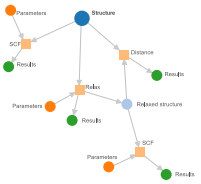
AiiDA documentation:
`aiida-core.readthedocs.io/en/latest`

MARVEL
NATIONAL CENTRE OF COMPETENCE IN RESEARCH

MAX DRIVING THE EXASCALE TRANSITION

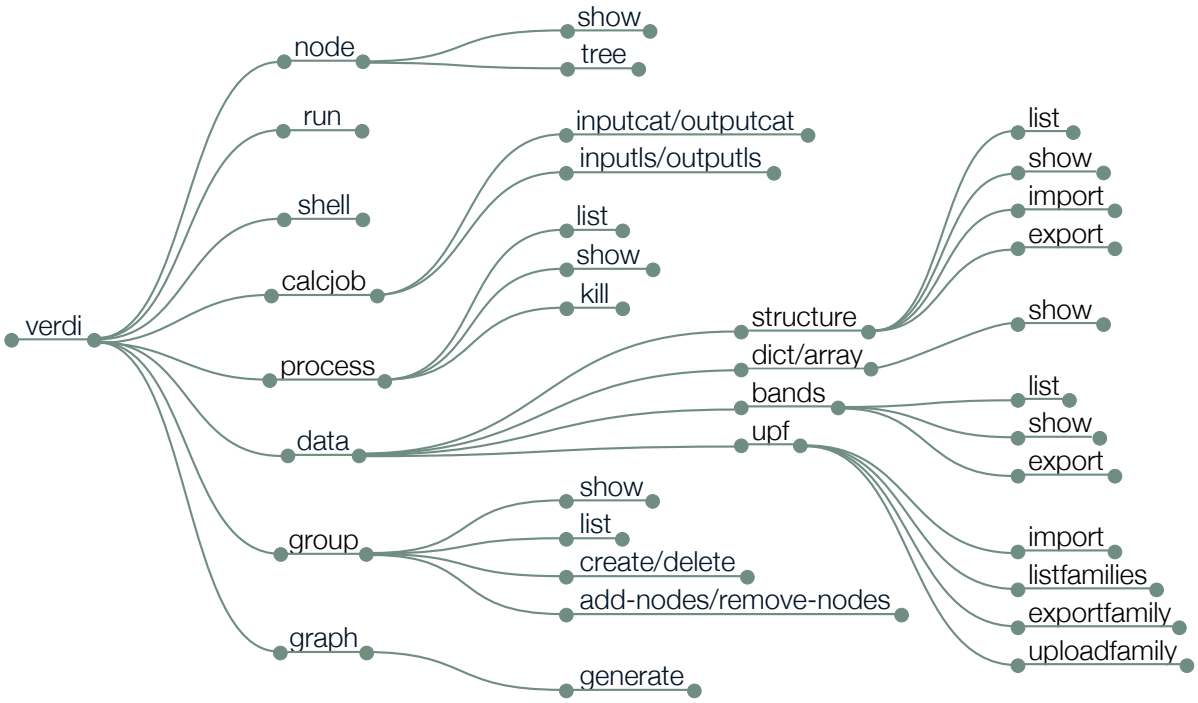


AiiDA



The `verdi` command-line tool

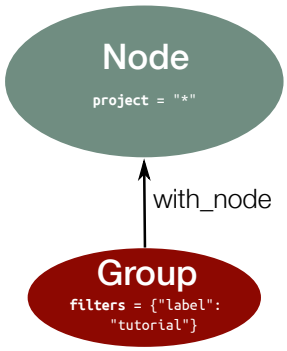
This list shows only the most common commands. Use TAB auto-completion at any level for full list



The QueryBuilder

To import: `from aiida.orm import QueryBuilder`

Fetch all nodes of group "tutorial"




```
qb = QueryBuilder()
qb.append(Node,
          tag="nodes",
          project="*")

qb.append(Group,
          with_node="nodes",
          filters={"label": "tutorial"})

qb.all()
```

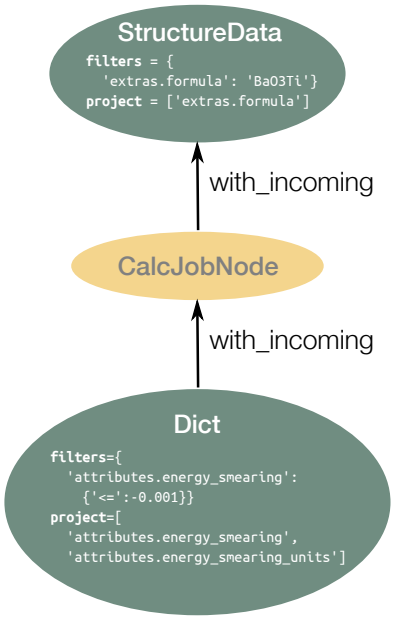
Print the smearing energy calculated for BaO_3Ti if it is smaller than 10^{-4} eV

```
qb = QueryBuilder()
qb.append(
    StructureData,
    project=["extras.formula"],
    filters={"extras.formula":"BaO3Ti"},
    tag="structure")

qb.append(
    CalcJobNode,
    tag="calculation",
    with_incoming="structure")

qb.append(
    Dict,
    tag="results",
    filters={"attributes.energy_smearing":
    {"<=":-0.0001}},
    project=[
        "attributes.energy_smearing",
        "attributes.energy_smearing_units"
    ],
    with_incoming="calculation")

qb.all()
```



EPFL swissuniversities INTERSECT