D2.1

Plugins for code interoperability

Daniele Tomerini, Alberto García, Matteo Bertocchi, Claudio Rosati,

Andrea Padovani, Pablo Ordejón, and Nicola Marzari

Document information

Project acronym:	INTERSECT
Project full title:	Interoperable Material-to-Device simulation box for disruptive electronics
Research Action Project type:	Accelerating the uptake of materials modelling software (IA)
EC Grant agreement no.:	814487
Project starting / end date:	1st January 2019 (M1) / 31st January 2022 (M37)
Website:	www.intersect-project.eu
Final version:	30/01/2020
Deliverable No.:	D2.1
Responsible participant:	EPFL (participant number 2)
Due date of deliverable:	31/01/2020
Actual submission date:	30/01/2020
Dissemination level:	PU - Public

Authors: Daniele Tomerini, Alberto García, Matteo Bertocchi, Claudio Rosati, Andrea Padovani, Pablo Ordejón and Nicola Marzari

To be cited as: D. Tomerini, A. García, M. Bertocchi, C. Rosati, A. Padovani, P. Ordejón, and N. Marzari (2020): Plugins for code interoperability. Deliverable D2.1 of the H2020 project INTERSECT (final version as of 30/01/2020). EC grant agreement no: 814487, EPFL, Lausanne, Switzerland.

Content

[1] Acronyms are marked in purple in the text and defined at the end of the document.

1 Executive Summary

This document provides a description of the status and development of AiiDA, and of the plugins necessary for the intercommunication between AiiDA, Quantum ESPRESSO, SIESTA and Ginestra™ within the upcoming Interoperable Materials to device (IM2D) framework. The AiiDA code is in charge of handling the communication with the codes of the "simulation hub" of the IM2D.

The open-source AiiDA python framework has reached version 1.0, introducing new features and improvements from the previous version. The communication with AiiDA and external codes is possible thanks to a plugin system that allows the extension of the capabilities of AiiDA by defining how to deal with external packages, i.e. how to prepare the inputs and how to retrieve information from the outputs of a calculation performed by a specific code.

The Quantum ESPRESSO plugin and SIESTA plugin have been developed and adapted through 2019, and their codes are now fully compatible with the new version of AiiDA by providing the necessary python calculation classes and output parsers.

The Ginestra™ suite can interact with AiiDA through an extension of its REST API, in order to submit workflows, retrieve data and query information from an AiiDA database not necessarily installed on the same machine where Ginestra™ is running. On the Ginestra™ side, a Java applet can handle the query of crystal structures in the AiiDA internal database and Crystallography Open Database (COD) and the submission and monitoring of workflows, as well as the retrieval of the results and general interactions with the AiiDA REST API.

2 Introduction

One of the goals behind the INTERSECT project is to setup an Interoperable Materials to device (IM2D) framework able to integrate materials modelling codes, in a way that permits mutual interaction and exchange of information; this effort is powered by the AiiDA workflow engine, as an integrated program capable of handling simulation of complex materials requiring both material and device properties.

Due to its modular design, AiiDA can be easily interfaced with any material science code, so that it can prepare, submit, retrieve and store the results of a series of calculations, and automatically keep the provenance and the logic of each step; the provenance is stored in a database for easy querying and retrieval of the data and their generations, guaranteeing automatic provenance of data long after a calculation has been completed.

The plugins for Quantum ESPRESSO and SIESTA have been developed, updated to AiiDA version 1.0 and deployed to the plugin repository. The interaction between AiiDA and Ginestra™ is ensured by an extension of the AiiDA REST API that can be called from a developers' version of Ginestra™ for the submission and retrieval of data, workflows, and information. By implementing the interconnection

among the original stand-alone codes and models, AiiDA is a key enabler for the *syntactic interoperability* stage of the development of the IM2D infrastructure.

2.1 About this document

This document describes the status of deliverable D2.1, concerning the code interoperability.

3 Codes status

3.1 AiiDA

AiiDA [1] is the python framework at the center of the upcoming interoperability hub of the IM2D platform, in charge of handling the simulation hub and the data hub, and ensure the communication between the various simulation software packages: Quantum ESPRESSO [2], SIESTA [3] and Ginestra™ [4].

AiiDA has a modular design and is in itself code-agnostic: python packages ("plugins") can be added to extend its functionality, in order to provide access to the full functionality of a simulation code (e.g. for handling the preparation of inputs, running a calculation, store and transform the results of a calculation in objects that AiiDA can manipulate). More generally, a plugin can extend AiiDA python data types to provide new functionalities, ways to support job schedulers for High-Performance Computing (HPC), transport mechanisms to move data to and back from HPC centers, and database import/export of data and properties.

In the context of supporting the functionality of a code within AiiDA, a plugin should define at least

- A "calculation class" [5]: a python class that handles the preparation of all the inputs required to run a calculation: writing to files all the traditional text inputs and parameters from AiiDA nodes, given the code and the computer where it is executed. In general, a calculation class should aim to support all the functionality available by running the code directly. AiiDA will handle the transport of the necessary files to the computer (local or remote), handles the execution of the job according to the computer specifications, and monitor its status. Created files are retrieved and copied back to the machine where AiiDA is installed, according to a list of files to retrieve from the calculation.
- A "parser class": as well as the local files, the parser class is in charge to *translate* the outputs in order to extract relevant information that can be stored and queried more efficiently in the AiiDA database; for example, the final structure of a relaxation or a molecular dynamics trajectory from the text outputs can be parsed into a more natural AiiDA DataStructure class, while a list of properties can be stored in a Dictionary for easy query of the desired parameters.

The input, calculation, and outputs are stored in the database; their interconnection is available as a Direct Acyclic Graph (DAG), and the provenance of the calculation is ensured (Figure 1).

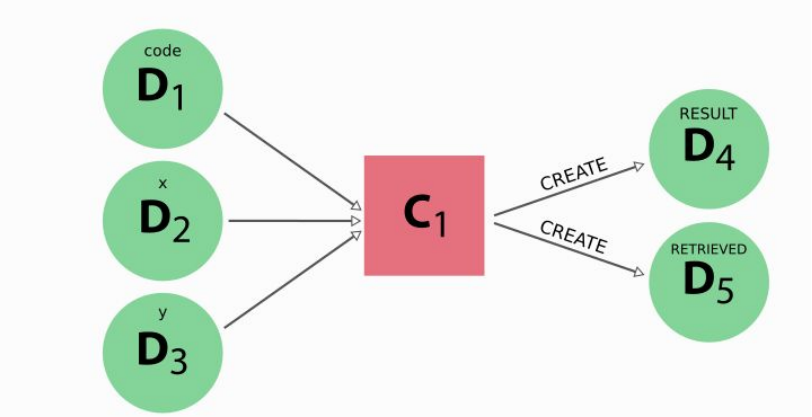

Figure 1. Example of a calculation from the AiiDA documentation: input nodes, including a code, the retrieved files and parsed results are all connected and queryable.

Plugins might also define AiiDA workflows: these provide an additional layer to handle multi-step calculations that are checkpointed and include more advanced logic [6].

AiiDA version 1.0 has been released on 1 November 2019, after a thorough beta testing. This new version includes several major changes with respect to the previous version, 0.12, including faster workflow engine, faster database access, reduced storage space and improved handling of command line interaction; see the change log [7].

The plugins developed for version 0.12 of AiiDA are incompatible with 1.0, due to breaking changes in the API; in March 2019 the AiiDA team in Lausanne hosted a plugin migration workshop, to support plugin migration to the new version.

Among the improvements for the submission and handling of calculation, there is the introduction of standardized exit codes, to provide information about the calculation failures, and an easier resubmission and pause mechanism, that reduces problematics related to the unavailability of a remote machine and provides finer control to the users.

The list of the public AiiDA plugins is available at the AiiDA plugin registry (https://aiidateam.github.io/aiida-registry/). The repository, as of January 2020, contains 88 plugins, divided into 34 packages, compatible with version 1.0 of AiiDA as shown in Figure 2.

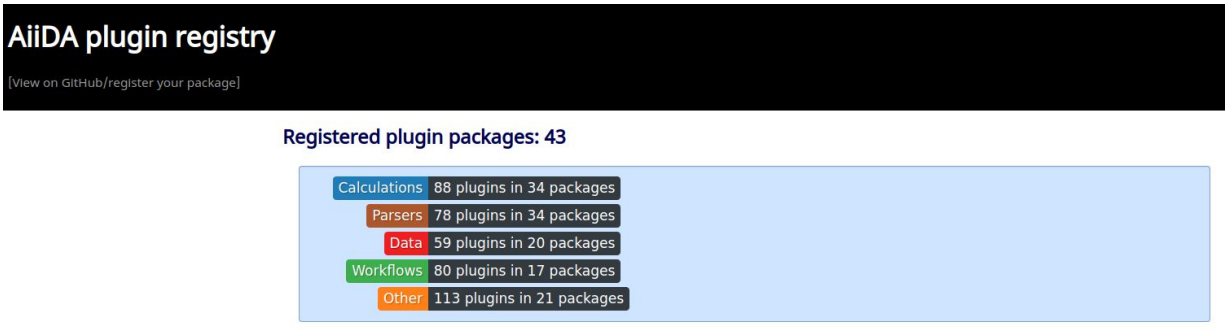

Figure 2. Welcome page of the AiiDA plugin registry as of January 2020.

3.2 AiiDA Quantum ESPRESSO interface

The AiiDA-QuantumEspresso plugin v.3.0 (Figure 3) [8] is compatible with AiiDA v.1.0 since September 2019. The plugin supports the main codes and postprocessing tools available within the Quantum ESPRESSO suite of programs, among others:

- pw.x: Ground state properties, total energy, ionic relaxation, forces
- cp.x: Car-Parrinello molecular dynamics
- ph.x: Phonons from density functional perturbation theory
- neb.x: Energy barriers and reaction pathways using the Nudged Elastic Band (NEB) method
- dos.x: Compute density of states (DOS)
- pw2wannier90.x: Interface between Quantum ESPRESSO and the Wannier90 [9] code

The plugin supports the Quantum ESPRESSO executables from versions 5.1 onwards.

The migration process from v 2.1 to v.3.0 was completed through 2019 with the migration to the new AiiDA 1.0 API. Version 2.1 of the plugin is still available and supported for users of AiiDA v.0.12.x

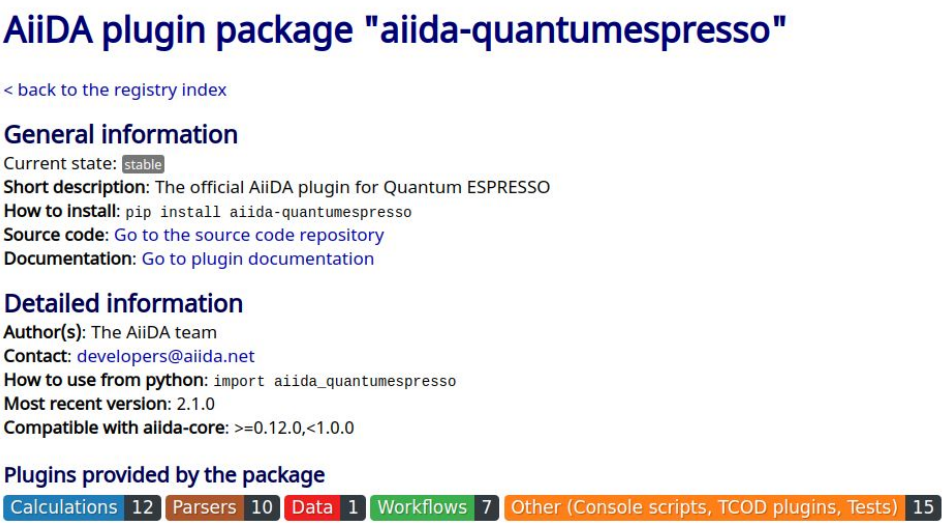

Figure 3. Detailed information about the plugins provided by the AiiDA-QuantumEspresso package.

The plugin introduces a new datatype to better deal with *forceconstant* objects, a number of command line scripts for easier submission of jobs (click scripts, their interface homogenized with Aiida's own *verdi* commands) and workflows that also take advantage of the new exponential backoff mechanism [9] implemented in AiiDA v 1.0.

Users can also decide to import already executed calculations so that they can take advantage of the functionalities in AiiDA.

Workflows are available for pw.x and ph.x. The former is the main DFT engine for electronic and structural calculations; the latter is devoted to the evaluation of vibrational properties (e.g. phonons, dielectric tensor, Born Charges, etc) based on Density Functional Perturbation Theory (DFPT). In particular, the pw.x workchains are modularly built on top of each other: the base scf workchain

handles restart upon, e.g., running out of walltime, or out of steps; there is also internal logic to deal with specific errors to improve the chances of recovery of a resubmitted calculation; the relax workchain builds upon it, adding handling of restarts and errors based on the relaxation behavior of the structure; the bands workflow further adds upon it for the calculation of the band structures upon specific reciprocal k-point paths. Finally, the band_structure workchain wraps the majority of parameters into a number of standard, optimized "protocol" parameters, so that the final user does not have to worry about providing most of the technical details of a calculation, that can, therefore, be run in high-throughput mode, with minimal intervention.

3.3 AiiDA-SIESTA interface

The AiiDA-SIESTA package is fully compatible with AiiDA 1.0 since its latest beta release (1.0.0). Its code base and the documentation [11] are accessible from the AiiDA plugin registry (Figure 4).

The SIESTA program is able to perform, in a single run, the computation of the electronic structure, the optional relaxation of the input structure, and a final analysis step in which a variety of magnitudes can be computed: band structures, projected densities of states, etc. The operations to be carried out are specified in a very flexible input format. The SIESTA plugin has been designed to be able to run the most general SIESTA calculation, with support for most of the available options (limited only by corresponding support in the parser).

The AiiDA-SIESTA package supports calculations executed from SIESTA versions 4.0.1 and later. It also offers a plugin for an STM simulation utility and introduces data types dealing with the norm-conserving-pseudopotential file formats used by SIESTA (PsfData and PsmlData). The PsmlData objects have the potential to also improve the interoperability between SIESTA and Quantum ESPRESSO, as SIESTA (meta)data is a superset of that in the corresponding Upf objects used by QE.

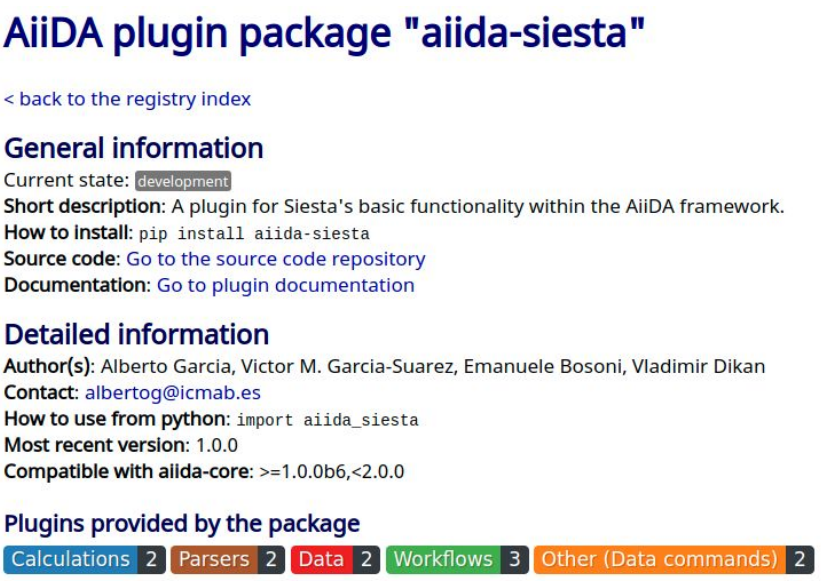

Figure 4. Detailed information about the plugins provided by the AiiDA-SIESTA package.

Calculations include error handling warnings and timings. The base workflow handles restart of the workflow in case of incomplete calculations, until convergence is reached (e.g., a relaxation not converged in the maximum allowed number of steps for a single calculation, or maximum time reached). The bands' workflow can be used to obtain and visualize the electronic band structure and makes use of protocols, collections of pre-set parameters that offer a balance of accuracy and efficiency. The choice of a workflow rather than a single SIESTA calculation also allows the use of different levels of accuracy (if required) in different steps of a workflow. As an example of an electronic-structure-analysis workflow, the AiiDA-SIESTA package offers an STM-image simulation workchain that harnesses the base workflow, also using protocol dictionaries to consolidate user-level options.

3.4 AiiDA-Ginestra™ interface

The AiiDA-Ginestra™ interface has been handled in a different way from the plugin system, in anticipation of the possibility of Ginestra™ directly querying the AiiDA database, submitting workflows and retrieving results.

The AiiDA package adopts a web interface that conforms to the Representational State Transfer (REST) architecture. The REST API allows for querying the database through a web service, exposing the underlying database where AiiDA is running, for their node properties and relationships.

The choice of interconnection between AiiDA and Ginestra™ has therefore been based on an extension of the AiiDA REST api, dealing with the submission of jobs (rather than just the retrieval of node data) on the AiiDA side, and a Java applet to allow Ginestra™ to query and submit data.

The package (aiida-post) is hosted at the Fraunhofer Gitlab repository created as of Deliverable D1.2, and is available to all the INTERSECT partners (Figure 5). As of now, any workflow contained in an installed AiiDA plugin is supported for submission, through the entrypoint mechanism provided by the *pkg_resources* python package. Further workflows can be added to the plugin directly by updating its setup.json file. The plugin contains detailed information about installation and examples of use [12].

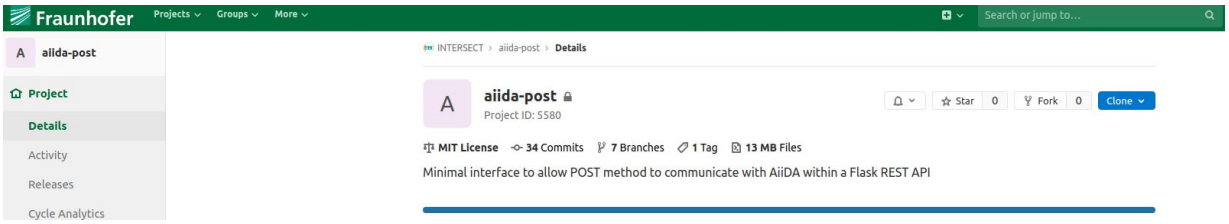

Figure 5. Snapshot of the aiida-post plugin on the Fraunhofer GitLab repository.

Upon HTTP queries, the plugin always returns a JavaScript Object Notation file (JSON), a simple and well-supported interchange file type format, that can be easily parsed and used. The json file provides information about the request, and additionally error messages to deal with an incorrect request.

For easier submission and query, rather than submitting a workflow, the user can define a “property” keyword directly related to a workflow, and ties it to one of the outputs of the workflow for retrieval.

The submission of a workflow can be easily performed with a “POST” HTTP request, by providing a json dictionary of the required and optional inputs of a workflow. On successful submission, the request json is stored in the database, and linked to the inputs of the workflow; the unique UUID identifier of the workflow is returned to the user, and it can be used to query the AiiDA database for the status of the calculation, the output nodes, and the required property. The provenance of the calculation therefore directly ties the json request to the final results, connecting all the in-between steps of the workflow.

The AiiDA-post plugin also provides utilities to query for a list of the available plugins, their input/output, and supported properties. All the information that the user can retrieve from the underlying REST API is also supported.

The Java applet is operative in version 3.2 of Ginestra™, at present used internally for testing.

The general flux that describes how a property can be retrieved or computed from the Ginestra™ Java applet is summarized by the schema (Figure 6):

1. definition of the chemical formula that represents the material or structure of interest
2. the structure is searched in the AiiDA database, all the matching structures are returned. In case the desired structure is not found it can be run a search and import workflow from an external database
3. a structure is selected
4. a physical property is selected
5. the property is searched in the AiiDA database. If it does not exist, the corresponding workflow for its computation can be submitted
6. the property value is retrieved and copied in the clipboard and may be imported in Ginestra™

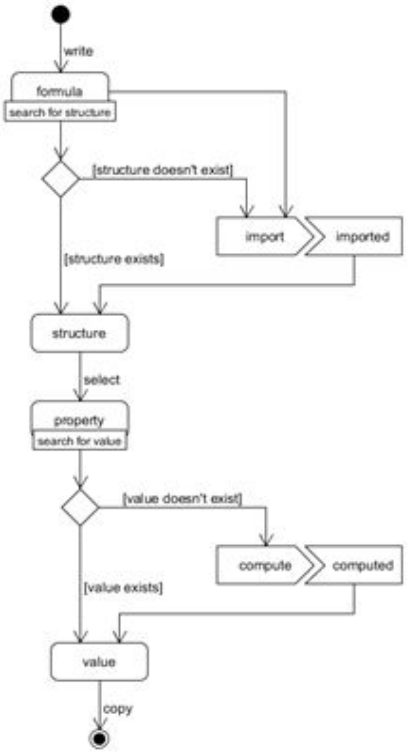

Figure 6. Schema represents the Ginestra™ Java applet interaction with the AiiDA interoperability plugin.

The Graphical User Interface (GUI) provides a simplified interaction with the intercommunication plugin, that allows to search data, submit workflows, monitor the status of the running simulation and retrieve computed data.

At each step the UUID is of the corresponding node is provided so that the structures and the workflow inputs and outputs may be inspected directly in the AiiDA database or by using other tools as Materials Cloud (www.materialscloud.org).

The GUI workflows submission is automated to create the input json files with the minimum data definition by the user side. Most of the parameters are set by default. It is provided an advanced interaction where the json input files may be edited and customized by the user.

Step 1: Chemical formula definition

The formula is checked and validated by the GUI according to the Hill and Hill-compact notation. Errors are highlighted while typing and a tooltip will explain the error. Only allowed characters are enabled. A Periodic Table of Elements dialog is provided to help filling the field by clicking on the corresponding element to insert it into the formula (Figure 7).

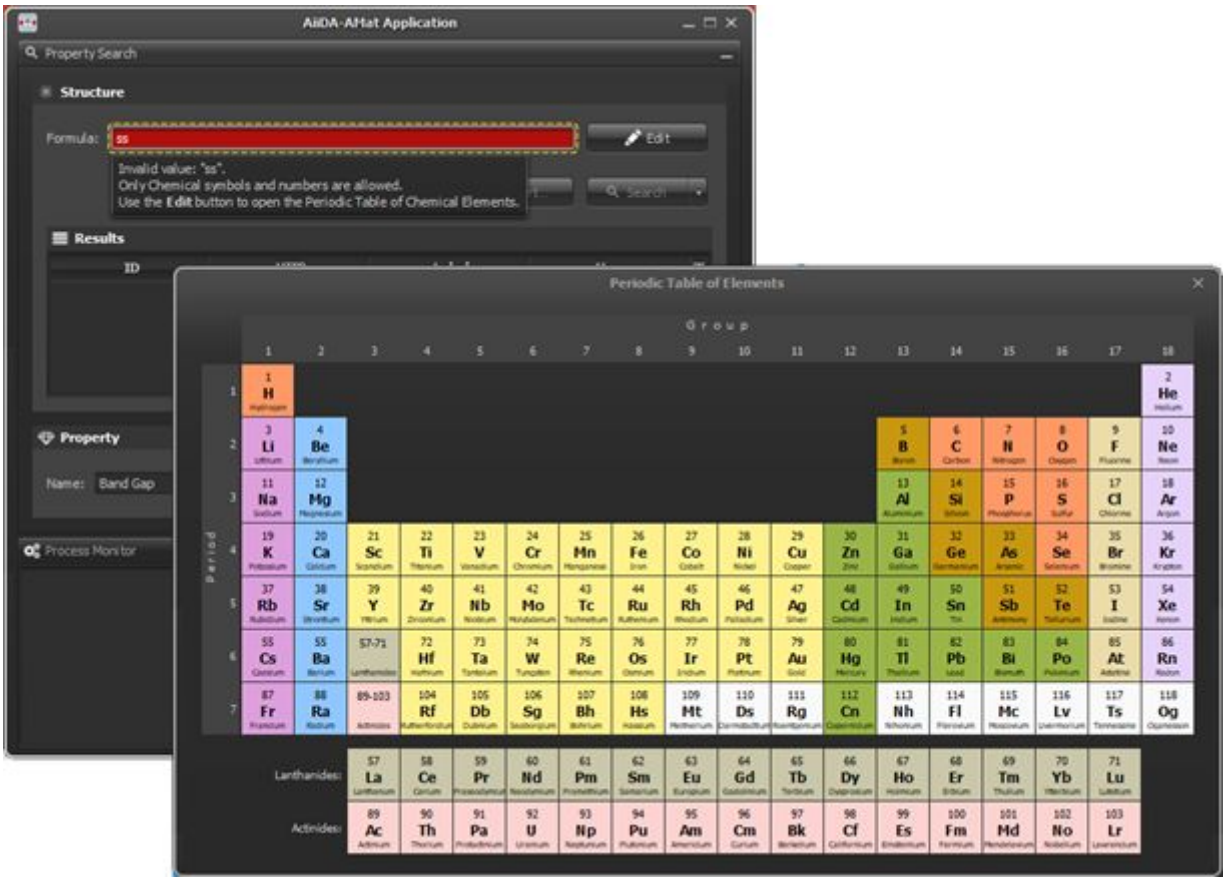

Figure 7. Snapshot of the chemical formula definition. The red background color of the "Formula" field shows that the current one is not a valid formula.

Step 2: Structure search/import

The GUI starts the search in the AiiDA database by pressing the *Search* button. The search progress is indicated by the progress bar with info about the percentage and estimated time (Figure 8).

All the structures that match the formula in the database are displayed as soon as they are found and are made immediately available to the user.

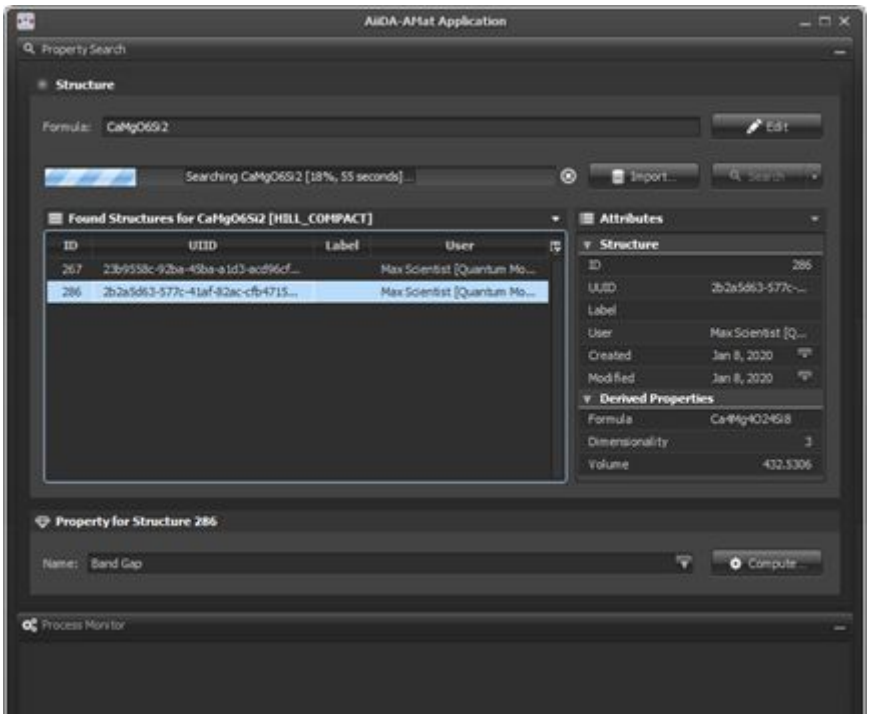

Figure 8. Snapshot of a search workflow execution while running. Progress and matching results are shown.

In case no structure is found, it can be retrieved (if existing) from an external database using the import workflow. Now the COD database (http://crystallography.net/cod/index.php) may be consulted.

Steps 3-6: From structure to property

Once the structure is found on the AiiDA database and selected in the *Results* table, all the existing properties associated with the structure are shown in the right panel. If the desired property (e.g. the band gap) is present in the list it can be copied. In case there are several nodes that match the query (e.g. several band gap values corresponding to computations with different convergence parameters) all of them are printed with the corresponding UUID so that one can inspect the results and choose the desired one.

If the property is not available in the existing properties list, it can be computed in the Property panel by selection of the corresponding property name and pushing the button *Compute*.

The status of a workflow submission is shown by a colored icon in the *Process Monitor list* (Figure 9). Here they are collected all the submitted and executed workflows. By right-clicking on a specific entry a menu allows the user to perform specific actions such as copying the workflow UUID or viewing details about the selected job.

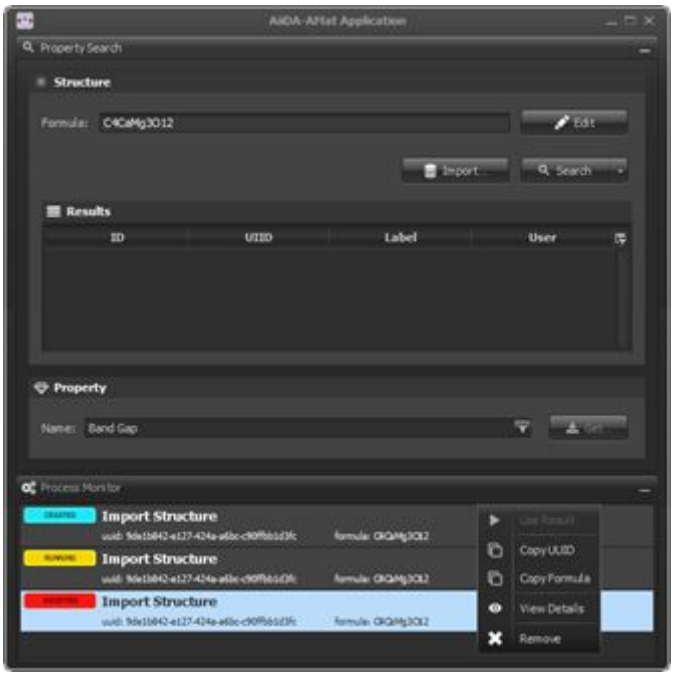

Figure 9. Process Monitor list: example of the import workflow from COD database. For completeness, all the available statuses have been printed.

Once the monitored element is in *Finished* status, in the menu it will be available the action *Use Result*. The action will copy the computed value in the system clipboard and add it to the existing properties associated with the structure.

Acronyms

API - Application Program Interface

COD - Crystallography Open Database

DAG - Direct Acyclic Graph

GUI - Graphical User Interface

HPC -High-Performance Computing

JSON - JavaScript Object Notation

REST - Representational State Transfer

UUID - Universally unique identifier

References

[1] G. Pizzi et al. Comp. Mat. Sci. 111, 218-230 (2016)

[2] P. Giannozzi *et al.*, J.Phys.:Condens.Matter **21**, 395502 (2009); P. Giannozzi *et al.*, J.Phys.:Condens.Matter **29**, 465901 (2017)

[3] José M Soler *et al* 2002 *J. Phys.: Condens. Matter* **14** 2745

[4] Ginestra™ website: https://www.mdlsoft.com/

[5] For more information about AiiDA calculations: https://aiida-core.readthedocs.io/en/latest/working/calculations.html#calculation-functions

[6] For more information about AiiDA workflows: https://aiida-core.readthedocs.io/en/latest/reference/index.html#workflows

[7] AiiDA changelog https://github.com/aiidateam/aiida-core/blob/v1.0.0/CHANGELOG.md

[8] AiiDA-QuantumEspresso plugin https://github.com/aiidateam/aiida-quantumespresso

[9]Giovanni Pizzi *et al, Wannier90 as a community code: new features and applications* 2020 *J. Phys.: Condens. Matter* **32** 165902

[10] For more information https://aiida.readthedocs.io/projects/aiida-core/en/latest/concepts/calculations.html#exponential-backoff-mechanism

[11] AiiDA-SIESTA documentation https://aiida-siesta-plugin.readthedocs.io

[12] AiiDA-post plugin https://gitlab.cc-asp.fraunhofer.de/intersect/ext_to_aiida