Computational Materials Science 137 (2017) 362–370

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Editor's Choice

# AFLUX: The LUX materials search API for the AFLOW data repositories

Frisco Rose [a], Cormac Toher [a], Eric Gossett [a], Corey Oses [a], Marco Buongiorno Nardelli [b], Marco Fornari [c], Stefano Curtarolo [a,*]

[a] Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA
[b] Department of Physics and Department of Chemistry, University of North Texas, Denton, TX 76203, USA
[c] Department of Physics, Central Michigan University, Mount Pleasant, MI 48858, USA

ARTICLE INFO

*Article history:*
Received 29 March 2017
Accepted 29 April 2017
Available online 20 June 2017

ABSTRACT

Automated computational materials science frameworks rapidly generate large quantities of materials data for accelerated materials design. In order to take advantage of these large databases, users should have the ability to efficiently search and extract the desired data. Therefore, we have extended the data-oriented AFLOW-repository Application-Program-Interface (API) (Comput. Mater. Sci. **93**, 178 (2014)) to enable programmatic access to search queries. A Uniform Resource Identifier (URI)-based search API is proposed for the construction of complex queries for remote creation and retrieval of customized data sets. It is expected that the new language, AFLUX, from "Automatic Flow of LUX (light)", will enable remote search operations on the AFLOW set of computational materials science data repositories. In addition, AFLUX facilitates the verification and validation of the data in the AFLOW repositories.



## 1. Introduction

Automated computational materials science frameworks such as AFLOW [1–7] rapidly generate large quantities of materials data, which can be used to identify trends, build machine learning models [8–12], formulate descriptors for synthesizability [13,14] and ultimately accelerate materials design [15]. The dissemination of such large quantities of data requires the use of advanced online databases such as AFLOW [16–18], NoMaD [19], Materials Project [20], OQMD [21] and AiiDA [22,23] to make the data accessible in an effective fashion.

The AFLOW data repositories [16–18] currently contain in excess of 1.6 million materials entries, and over 160 million computed properties which are directly accessible using the existing AFLOW data REpresentational-State-Transfer Application-Program-Interface (REST-API) [18]. The existing data API presents data effectively, but it does not enable programmatic access to the types of complex search queries available through the AFLOW online web portal. In order to address these issues, we have implemented a Uniform Resource Identifier (URI)-based search API which enables the construction of complex search queries and allows for the creation and retrieval of customized data sets.

For large, automatically generated data sets such as those contained in the AFLOW repository, automated verification procedures for calculation quality is an important issue, since it is impossible for a user to individually check the calculations for every single entry in a data set. The Automatic Flow of LUX (light) (AFLUX)-search API can be used to facilitate such verification checks. It is possible to include properties which provide information about calculation quality in the results returned by a search query, and to filter the results to remove entries which do not fulfill specified convergence criteria. To facilitate this, the AFLOW data REST-API has been extended to include additional keywords to expose information concerning the calculation settings and convergence, as listed in Appendix C. This upgraded version of the data API is termed version 1.2 as compared to version 1.0 described in Ref. 18. Even and odd version numbers correspond to stable and developmental iterations, respectively.

## 2. AFLUX search API

Access to relational information is typically a highly involved process that requires specialized knowledge not generally available or readily attainable by general users. An access mechanism that exposes the data to human-directed exploration via a graphical user interface (GUI) is typically limited by the GUI design (as imagined by the developers), and is practically impossible to access via programmatic methods. Exposing data via a procedural approach, such as through the Structured Query Language (SQL), requires knowledge of both the language in use and the (often highly convoluted) organization of the underlying data. In order

* Corresponding author.
*E-mail address:* stefano@duke.edu (S. Curtarolo).

http://dx.doi.org/10.1016/j.commatsci.2017.04.036
0927-0256/

to address these concerns, we created a domain-agnostic text-based search API named LUX. LUX is presented in the AFLOW domain context as AFLUX with the design goals of exposing an API that is human accessible, logically robust, concise and as lucid as possible.

### 2.1. Design specification

The design of this API is driven by the need for a straightforward and globally accessible representation of the AFLOW search features. The feature set should reflect and extend the capabilities found in the AFLOW online search GUI available at aflow.org. Further motivation comes from the attempt to unburden the end user from needing to understand the intricacies of SQL or a particular database schema. An additional benefit of eliminating the explicit representation of the database (DB) schema is that we are free to optimize the internal DB structure without breaking backwards compatibility in the AFLUX API. Although LUX is designed to operate in an origin-agnostic fashion, concessions must be made to older software stacks that may be unable to create an extended length URI, and thus LUX strives to balance conciseness with human readability. As a final consideration, the LUX syntax is designed to be generally extensible, both to accommodate the growth of the underlying data and to allow LUX's application to an arbitrary data store. The LUX API language construct is superficially like a C/C++ style subroutine/function call, which we term the `matchbook`. The utility of LUX is derived from the fact that the intra-`matchbook` restrictions and inter-`matchbook` declarations are logically related search criteria.

### 2.2. URI query context

This implementation of the LUX search API uses an internet URI as the submission layer. In a related effort, the AFLOW data API also uses a URI to access large data repositories [18]. Whereas in the data API the path portion of the URI is used to uniquely identify each material and navigate between materials, the query portion of the URI in LUX is used to isolate specific data values and dictate the associated data response format. Therefore, for the LUX API, the query portion of the URI is best suited to the highly dynamic nature of LUX's relational criteria.

The beginning of the query section in the URI is defined by the presence of the first question mark character "?" in the URI. The query section is terminated by either the presence of the fragment section, indicated by the first octothorpe character "#", or by the end of the URI.

Thus the standards for the query portion of a URI form the foundation of the LUX API.

### 2.3. Summons syntax

The summons syntax is broken into two segments with URI query safe characters forming the body of the summons, as shown in Fig. 1(a). The first portion contains the relational property match criterion with its respective matching restrictions – this is the `matchbook`. The `matchbook` portion is followed by the `directive` portion with its attendant attributes. While there is no *a priori* order dependency within the `matchbook` or the `directive`, the `matchbook` must come entirely before the `directive` in the submitted URI, as shown in Fig. 1.

The property `matchbook` portion is mandatory for a meaningful data response. In contrast a `directive`-only summons may return metadata or an error state, depending on the `directive` of concern and the manner in which it is summoned.

### 2.4. Character set

As one of the primary design objectives of LUX is to be human-readable, the allowed URI characters must be considered in that context. The URI generic syntax (RFC3986 [24]) defines explicit restrictions on URI query formation in ABNF notation [25]. We limit our overt use of percent (pct) encoded characters. Finally, in order to support a concise summons, we forgo verbose operators and use single character operators. What follows is a derivation of the LUX character set based on our design goals and the limitations of the standards that LUX operates under.

From Section 3.4 of Ref. 24 the query section of the URI allows for the use of the following characters

$$\text{query} = {}^{*}(\text{pchar} \;/\; \text{“/”} \;/\; \text{“?”})$$

which can be read as follows: allow zero or more of the characters from the conjunction of the pchar set, the solidus character "/", and the question mark character "?", where the "pchar" character set is defined in Section 3.4 of Ref. 24 as

$$\text{pchar} = \text{unreserved} \;/\; \text{pct-encoded} \;/\; \text{sub-delims} \;/\; \text{“:”} \;/\; \text{“@”}$$

"Unreserved" is defined in Section 2.3 of Ref. 24 as

$$\text{unreserved} = \text{ALPHA} \;/\; \text{DIGIT} \;/\; \text{“-”} \;/\; \text{“.”} \;/\; \text{“_”} \;/\; \text{“~”}$$

and "sub-delims" are defined in Section 2.2 of Ref. 24 as

$$\begin{aligned}\text{sub-delims} = {} & \text{“!”} \;/\; \text{“\$”} \;/\; \text{“\&”} \;/\; \text{“'”} \;/\; \text{“(”} \;/\; \text{“)”} \;/\; \text{“*”} \\ & /\; \text{“+”} \;/\; \text{“,”} \;/\; \text{“;”} \;/\; \text{“=”}\end{aligned}$$

So, in summary, a strict superset of the characters that our syntax may be constructed from is as follows:

$$\begin{aligned}&\text{ALPHA} \;/\; \text{DIGIT} \;/\; \text{“-”} \;/\; \text{“.”} \;/\; \text{“_”} \;/\; \text{“~”} \;/\; \text{“!”} \;/\; \text{“\$”} \;/\; \text{“\&”} \;/\; \text{“'”} \\ &\quad /\; \text{“(”} \;/\; \text{“)”} \;/\; \text{“*”} \;/\; \text{“+”} \;/\; \text{“,”} \;/\; \text{“;”} \;/\; \text{“=”} \;/\; \text{“:”} \;/\; \text{“@”} \;/\; \text{“/”} \;/\; \text{“?”}\end{aligned}$$

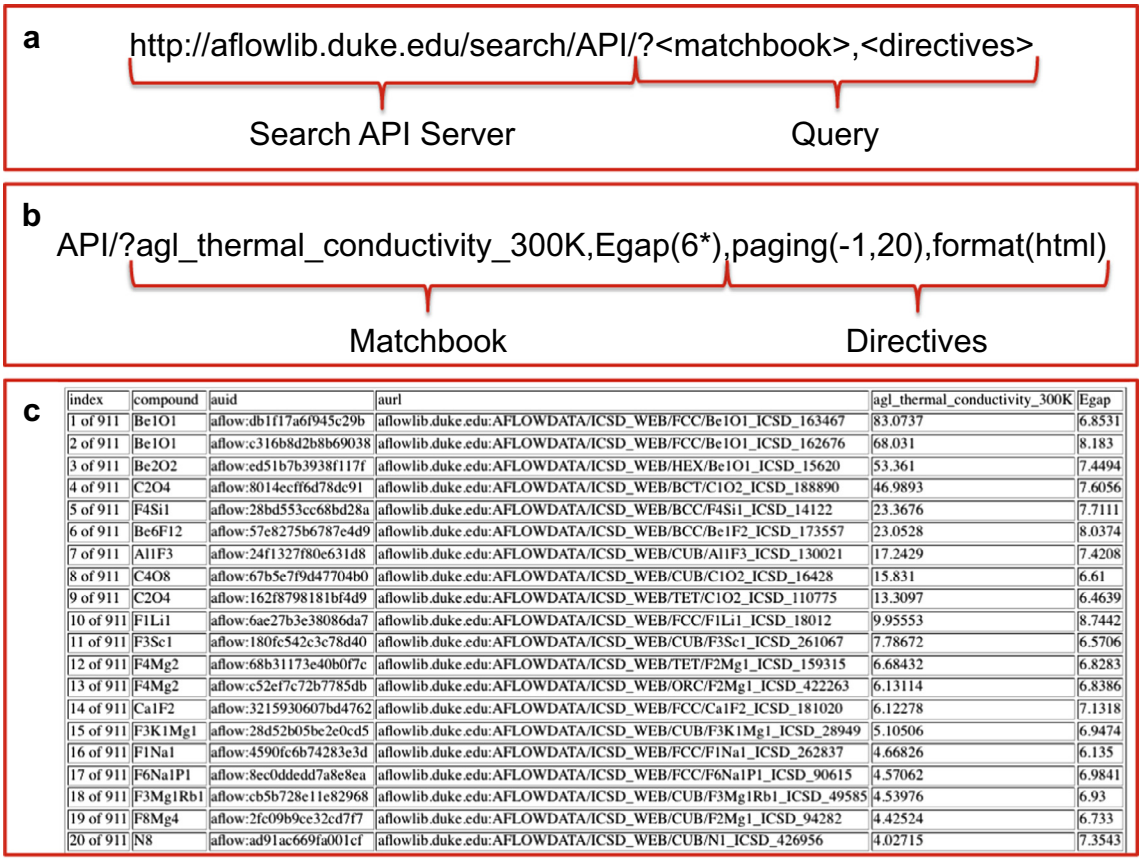
However, while this is the latest relevant RFC there are older protocols implemented by browsers/servers/languages that interfere with the query character set. For instance, RFC1630 and RFC2396 reserved the plus sign. We attempt to avoid any characters that may result in an unintended mangling of the query.

### 2.5. Operator character codex

We are not explicitly limited by any origin standards, such as HTML5 which only allows unreserved and pct-encoded, as the origin of the query is application agnostic and only requires the ability to construct a valid URI. As an aside, it is often possible to work around the limitation of origin standards by calling lower level methods in the application stack. In essence, we need only open a bi-directional communication socket to the LUX API, send an un-mangled URI with an appropriate query structure to the server, and accept the returned data set. That said, we want to make the implementation as easy, unambiguous and unimpeded as possible. As a first consideration we will avoid the use of "?", "&", "+", ";" and "=" in order to avoid confusing arbitrarily handled webserver/cgi/PHP/Web-Browser/... URI parsers. This will also help distinguish LUX from preconceived end user expectations of other, more simplistic, query based API behaviors. The characters "." and "-" are also reserved as they are inherent to the definition of a number. Furthermore, we want to use ALPHA/DIGIT as general string/number characters. Lastly "_" is reserved as it is used in the keyword (AFLOW Snake Case keywords for instance) identifier amalgamation convention. This leaves

$$\text{“~”} \;/\; \text{“!”} \;/\; \text{“\$”} \;/\; \text{“'”} \;/\; \text{“(”} \;/\; \text{“)”} \;/\; \text{“*”} \;/\; \text{“,”} \;/\; \text{“:”} \;/\; \text{“@”} \;/\; \text{“/”}$$

Thus, the LUX operator tokens are defined as shown in Table 1.




| index | compound | auid | aurl | agl_thermal_conductivity_300K | Egap |
|---|---|---|---|---|---|
| 1 of 911 | Be1O1 | aflow:db1f17a6f945c29b | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/FCC/Be1O1_ICSD_163467 | 83.0737 | 6.8531 |
| 2 of 911 | Be1O1 | aflow:c316b8d2b8b69038 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/FCC/Be1O1_ICSD_162676 | 68.031 | 8.183 |
| 3 of 911 | Be2O2 | aflow:ed51b7b3938f117f | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/HEX/Be1O1_ICSD_15620 | 53.361 | 7.4494 |
| 4 of 911 | C2O4 | aflow:8014ecff6d78dc91 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/BCT/C1O2_ICSD_188890 | 46.9893 | 7.6056 |
| 5 of 911 | F4Si1 | aflow:28bd553cc68bd28a | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/BCC/F4Si1_ICSD_14122 | 23.3676 | 7.7111 |
| 6 of 911 | Be6F12 | aflow:57e8275b6787e4d9 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/BCC/Be1F2_ICSD_173557 | 23.0528 | 8.0374 |
| 7 of 911 | Al1F3 | aflow:24f1327f80e631d8 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/Al1F3_ICSD_130021 | 17.2429 | 7.4208 |
| 8 of 911 | C4O8 | aflow:67b5e7f9d47704b0 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/C1O2_ICSD_16428 | 15.831 | 6.61 |
| 9 of 911 | C2O4 | aflow:162f8798181bf4d9 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/TET/C1O2_ICSD_110775 | 13.3097 | 6.4639 |
| 10 of 911 | F1Li1 | aflow:6ae27b3e38086da7 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/FCC/F1Li1_ICSD_18012 | 9.95553 | 8.7442 |
| 11 of 911 | F3Sc1 | aflow:180fc542c3c78d40 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/F3Sc1_ICSD_261067 | 7.78672 | 6.5706 |
| 12 of 911 | F4Mg2 | aflow:68b31173e40b0f7c | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/TET/F2Mg1_ICSD_159315 | 6.68432 | 6.8283 |
| 13 of 911 | F4Mg2 | aflow:c52ef7c72b77785db | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/ORC/F2Mg1_ICSD_422263 | 6.13114 | 6.8386 |
| 14 of 911 | Ca1F2 | aflow:3215930607bd4762 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/FCC/Ca1F2_ICSD_181020 | 6.12278 | 7.1318 |
| 15 of 911 | F3K1Mg1 | aflow:28d52b05be2e0cd5 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/F3K1Mg1_ICSD_28949 | 5.10506 | 6.9474 |
| 16 of 911 | F1Na1 | aflow:4590fc6b74283e3d | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/FCC/F1Na1_ICSD_262837 | 4.66826 | 6.135 |
| 17 of 911 | F6Na1P1 | aflow:8ec0ddedd7a8e8ea | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/FCC/F6Na1P1_ICSD_90615 | 4.57062 | 6.9841 |
| 18 of 911 | F3Mg1Rb1 | aflow:cb5b728e11e82968 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/F3Mg1Rb1_ICSD_49585 | 4.53976 | 6.93 |
| 19 of 911 | F8Mg4 | aflow:2fc09b9ce32cd7f7 | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/F2Mg1_ICSD_94282 | 4.42524 | 6.733 |
| 20 of 911 | N8 | aflow:ad91ac669fa001cf | aflowlib.duke.edu:AFLOWDATA/ICSD_WEB/CUB/N1_ICSD_426956 | 4.02715 | 7.3543 |

**Fig. 1.** (a) URI format for LUX search query. Note that the entire `matchbook` must be before all of the `directives` in the query. (b) Example search query for a material to act as an electrically insulating heat sink, with the materials properties of high thermal conductivity and a band gap exceeding 6 eV. **(c)** Results returned in HTML format. The best candidates found are two different structures of BeO.

**Table 1**
LUX Codex. List of the symbols used to represent logic operations in the LUX Search-API syntax. Note that three symbols are currently reserved for future use.

| Logic operation | Operator symbol | Description |
|---|---|---|
| BLOCK-NEW | ( | The new set precedence context |
| BLOCK-END | ) | The end set precedence context |
| UNARY-MUTE | $ | The property output suppression operator |
| UNARY-NOT | ! | The logical inversion unary operator |
| UNARY-LOOSE | * | The positional loose datum match unary operator |
| BINAL-AND | , | The logical conjunction binary operator/list separator |
| BINAL-OR | : | The logical disjunction binary operator/list separator |
| DATUM-STR | ' | The explicit string datum context |
| RESERVED | @ | Reserved for future use |
| RESERVED | ~ | Reserved for future use |
| RESERVED | / | Reserved for future use |

### 2.6. Properties are reserved words

In addition to the character limitations as defined by the operator character codex, there exist reserved words. These words fall into two categories: domain specific (*e.g.*, AFLOW properties) keywords and LUX `directives`. The list of the AFLOW property data keywords valid as of this draft are those keywords in the original AFLOW data REST API [18] and recently developed keywords listed in Appendix C.

### 2.7. Directives are pseudo property keywords

LUX's `directives` do not necessarily have an explicit presence in the output data. Their main purpose is to provide context and affect the way that the output is produced.

The list of `directives` is intentionally short and is as follows:

[“catalog”, “format”, “help”, “paging”, “schema”]

**Directive: catalog.** It is often helpful to provide a higher-level abstraction of the data sets available. In AFLOW we need to control which library or libraries of materials we wish to query. The internal implementation of the catalog `directive` could have been provided as a domain specific keyword if the defining property was guaranteed to be ubiquitous. However, this may not be possible for all domains and thus “catalog” has been designated in the `directive` context to allow for specialized internal handling. Within AFLOW the returned set of material properties can be restricted by the use of the “catalog” `directive`. The currently available catalogs are: [“icsd”, “lib1”, “lib2”, “lib3”].

By default all catalogs are included in a search request and hence the catalog `directive` may be omitted in many cases. The principal use-scenario for the catalog `directive` is to restrict the search to a subset of endpoint resources. The catalog `directive` is called as:

`API-Server?<matchbook>,catalog(library_a)`

for just “library_a”; or as

`API-Server?<matchbook>,catalog(library_a:library_b)`

to search inclusively in both libraries "a" and "b" (*i.e.* an entry can be in the conjunction of "a" and "b").

**Directive: format.** We need to be able to format output of the resultant data. The currently supported output modifiers are "html" and "json". The "format" keyword is used to set the output type. The default response is compact JSON, while setting "json" explicitly returns a more legible JSON response.

`API-Server?<matchbook>, format(json)`

or

`API-Server?<matchbook>, format(html)`

**Directive: help.** The help `directive` with no arguments will return a response with a summary of the proper syntax and use of LUX. If a specific help context has been internally defined for a keyword, then the keyword may be passed to the help `directive` to retrieve the associated message, otherwise the response will be the same as if no keyword had been supplied.

`API-Server?help()`

or

`API-Server?help(<keyword>)`

**Directive: paging.** We need the ability to control the number of responses as some searches may return exceedingly large data sets (potentially exceeding tens of millions of data points). By default, AFLUX returns the first response page of $k = 64$ data sets. This return limit can be overridden with the "paging" `directive` by specifying the number of data sets $k$ per page, or the $n$th page of $k$ data sets. Paging has special functionality: asking for zero results ($k = 0$) will return the number of matching data sets; asking for the zeroth page ($n = 0$) will return all matching data sets regardless of $k$ - **be careful** as this can be a large response! Negating $n$ will change the sort order to descending. The paging `directive` is called as:

`API-Server?<matchbook>, paging(n)`

or

`API-Server?<matchbook>, paging(n, k)`

**Directive: schema.** The schema `directive` allows external access to the internal data store metadata. The returned metadata represents useful information about the internal nature of the keyword data allowing an end user to make informed decisions about how to handle the property or properties of interest. The LUX language makes no assumptions regarding the metadata content. This allows freedom in the way that LUX is implemented on the backing store. For the AFLUX instance, the AFLOW data schema is exposed to allow many forms of disambiguation in the properties. There are a few metadata values that are of special importance, specifically (in no particular order) title, type, units and verification. Of particular note is the verification metadatum, which exposes a contextually relevant set of certification criteria that lend validity to the property of interest. Each property has a potentially unique set of metadata and should be examined prior to using the property in your research. The following are examples of using the schema `directive`:

`API-Server?schema()`

or

`API-Server?schema(<keyword>)`

*Nota Bene*: **Directives are not logical.** The `directives` do not support inter-`directive` logical relations. Intra-`directive` properties are not required to be logically related. Any attempt to depend on a logical relationship involving `directive` keywords may not have the desired result. The `directive` keywords are not interpreted in blocking context. Any attempt to modify precedence by the use of blocking parentheses will likely cause a failure in the request. Using `directive` keywords as the sole operators in a request may result in an error state being returned.

### 2.8. Implicit procedures

The first match keyword in the `matchbook` is used as the ordering criteria for the response data set, so that the returned entries are listed in ascending order of the property accessed by that keyword.

In the AFLUX implementation the response always includes the compound formula, the AUID and the AURL; *i.e.* the keywords `compound`, `auid` and `aurl` are implicitly included in the query [18]. If any of these properties are undesirable then they may be suppressed by application of the suppression operator: *e.g.* the following will prevent the response from producing the `compound` property

`API-Server?<matchbook>, $compound`

The first data response is paged with 64 entries per page and retrieval of subsequent pages require an explicit use of the paging `directive`.

### 2.9. Matching criteria

The AFLUX Search-API can filter for search results that match a range of different criteria, including exact matches and value ranges for both strings and numerical data.

**Numerical.** For scalar filtering of a single element we have three basic operations and they are

< "equal" | "less" | "more" >

and their inverses, *i.e.*

< "not equal" | "not less" | "not more" >

The results returned for a numerical value associated with a particular keyword can be restricted to specific ranges using the syntax shown in Table 2. Note that the operations "less than" and "greater than" are constructed by negating the "greater than or equal to" and "less than or equal to" operations, respectively.

**Strings.** For string matching we use a similar construction, as shown in Table 3.

Note that the string operator </> is optional unless the matching string contains a reserved character.

**Multiple criteria.** The search described so far only demonstrates a single matching condition, hereinafter referred to as a

**Table 2**
Numerical filtering.

| Query syntax | Result description |
|---|---|
| <keyword>(1.234) | Return only results where value in <keyword> equals 1.234 |
| <keyword>(!1.234) | Return only results where value in <keyword> does not equal 1.234 |
| <keyword>(*1.234) | Return only results where value in <keyword> is less than or equal to 1.234 |
| <keyword>(!*1.234) | Return only results where value in <keyword> is not less than or equal to 1.234, *i.e.* is greater than 1.234 |
| <keyword>(1.234*) | Return only results where value in <keyword> is greater than or equal to 1.234 |
| <keyword>(!1.234*) | Return only results where value in <keyword> is not greater than or equal to 1.234, *i.e.* is less than 1.234 |

match. The logical construction of multiple matches, referred to as the `matchbook`, is also supported. This is done by the juxtaposition of matches with one of the two list separators, namely <:> the logical _OR_ operator and <,> the logical _AND_ operator. Precedence is maintained by using nested lists. For instance, if we want to match two criteria simultaneously, the `matchbook` might look as shown in the first row of Table 4. When matching multiple criteria, two forms are possible, as shown in the remaining rows of Table 4.

## 3. Examples

### 3.1. Search for electrically insulating heat sink material

In this example, the steps to screen for an electrically insulating heat sink material for use in nanoelectronics are introduced. For this example, the search is for a material with a band gap in excess of 6 eV and thus will be electrically insulating, that also has a high value of the lattice thermal conductivity to efficiently transport heat away from the device. Therefore, as shown in Fig. 1(b), the `matchbook` is constructed by querying two materials properties keywords. The first keyword is `agl_thermal_conductivity_300K`, which is the lattice thermal conductivity at 300 K calculated using the AGL quasi-harmonic Debye-Grüneisen model as implemented within the AFLOW framework [26–28]. The second keyword is `Egap`, the electronic structure band gap obtained by taking the difference between the conduction band minimum and the valence band maximum, using the band structure calculated along the high-symmetry paths in reciprocal space as defined by the AFLOW Standard [17,3]. In this case, the search is restricted to return only materials with a band gap calculated to be greater than 6 eV by searching for `Egap(6*)`. There are two `directives` after the `matchbook`, both related to the formatting of the returned output. The `paging(-1,20) directive` instructs the API to return the first 20 results in descending order, while the `format(html) directive` instructs the API to return the results in HTML format.

The returned results for this search are shown in Fig. 1(c). In this case, 911 entries are found in the AFLOW data repository which match the requested search criteria. The entries are sorted in descending order of the value corresponding to the first property keyword in the `matchbook`, `agl_thermal_conductivity_300K`, due to the use of a negative value for the page number in the argument to the `paging directive`. Therefore, the materials with the highest thermal conductivity which would be most suitable for this application appear at the top of the list, namely two different structural phases of the material BeO.

**Table 3**
String filtering.

| Query syntax | Result description |
|---|---|
| <keyword>('foo') | Return only results where the string in <keyword> is exactly "foo" |
| <keyword>(!'foo') | Return only results where the string in <keyword> is not "foo" |
| <keyword>(*'foo') | Return only results where the string in <keyword> ends with "foo" |
| <keyword>(!*'foo') | Return only results where the string in <keyword> does not end with "foo" |
| <keyword>('foo'*) | Return only results where the string in <keyword> starts with "foo" |
| <keyword>(!'foo'*) | Return only results where the string in <keyword> does not start with "foo" |
| <keyword>(*'foo'*) | Return only results where the string in <keyword> contains the substring "foo" |
| <keyword>(!*'foo'*) | Return only results where the string in <keyword> does not contain the substring "foo" |

**Table 4**
Multiple criteria filtering.

| Query syntax | Result description |
|---|---|
| <keyword>(1*,*1.234) | Returns only results where value in <keyword> is between 1.0 and 1.234 |
| <keyword>(1*),<keyword>(*1.234) | Logically equivalent to <keyword>(1*,*1.234) |
| <keyword_a>(1.234), <keyword_b>('foo') | Return results that match for <keyword_a>(1.234) and <keyword_b>('foo') |
| <keyword_a>(1.234) : <keyword_b>('foo') | Return results that match for <keyword_a>(1.234) or <keyword_b>('foo') |

### 3.2. Verification tests

In this example, the steps to verify convergence of the structural relaxation calculations are introduced. Such checks are important to catch entries for which the relaxation is incomplete, especially when calculations are run automatically without direct user supervision. To facilitate such verification tests, additional keywords have been implemented in the data API to expose information about the stress tensor, Pulay stress, residual external pressure on the relaxed cell and the $\delta E$ value for the final electronic convergence step. These properties are used along with other property keywords already available in the database, such as `forces` (the forces on the atoms) to check for the convergence of the relaxation of the ion positions as well as cell size and shape, or to check the convergence of the electronic self-consistent iterations. Other calculation inputs such as the **k**-point grid and the energy cutoff for the plane wave basis set are also already available via the AFLOW API using the keywords `kpoints` and `energy_cutoff`.

The appropriate keywords to query for verification tests on a particular property are listed in the AFLUX schema for that property, which can be retrieved using the "schema" `directive` for that keyword, *i.e.*, by querying `schema(<keyword>)` using AFLUX. Fig. 2 shows the schema for the keywords `volume_atom` and `positions_fractional`, which are used to retrieve the average volume per atom and the ionic positions (in fractional coordinates), respectively. In both cases, these properties depend on the quality of the electronic convergence, and so the keywords to query include `kpoints` and `energy_cutoff`. The relaxation convergence of the ionic positions can be verified by checking the forces on the atoms using the keyword `forces`, while the volume convergence can be validated by checking the residual pressure on the cell and the stress tensor using the keywords `pressure_residual` and `stress_tensor`, respectively.

The relaxation of the size and shape of the cell can be checked for all of the entries in the AFLOW-ICSD catalog (Inorganic Crystal Structure Database) using the AFLUX search query `API-Server?stress_tensor,catalog(icsd)`, as demonstrated in the example code shown in Appendix B. The distribution of stress tensor components with the largest absolute value for each entry is shown in the histogram in Fig. 3. This shows that approximately half of the entries in the AFLOW-ICSD catalog have been converged so that all of the components of the stress tensor have a magnitude of less than 1 kB, while all of the entries in this catalog have no stress tensor component exceeding 10 kB. Calculations resulting in stress tensor components in excess of 10 kB are automatically detected and re-run by the AFLOW workflow using increased precision and cut-offs for the plane wave basis set.

Note that comparison against the initial volume of the experimentally measured ICSD structure is not necessarily the best method for performing data verification, as reported ICSD structures are frequently measured under extreme temperature or pressure conditions which will significantly impact bond lengths [29].

**a**

```
search/API/?schema(volume_atom,positions_fractional),format(json)
```

**b**


```
[
    {
        "__comment__": [
            ""
        ],
        "description": "Returns the volume per atom in the unit cell.",
        "title": "atomic volume",
        "format": "%s",
        "class": "crystal",
        "subclass": "real space lattice",
        "type": "number",
        "units": "&Aring;<sup>3</sup>/atom",
        "inclusion": "mandatory",
        "expression": "derivative",
        "verification": [
            "energy_cutoff",
            "forces",
            "kpoints",
            "pressure_residual",
            "stress_tensor"
        ],
        "example": "volume_atom=100.984",
        "status": "production",
        "syntax": "$aurl/?volume_atom"
    },
    {
        "__comment__": [
            ""
        ],
        "description": "Final fractional positions (xi,xj,xk) with respect to the unit cell as specified in $geometry.",
        "title": "relaxed relative positions",
        "format": "%s",
        "class": "other",
        "subclass": "",
        "type": "numbers",
        "inclusion": "mandatory",
        "expression": "derivative",
        "verification": [
            "energy_cutoff",
            "forces",
            "kpoints",
            "pressure_residual",
            "stress_tensor"
        ],
        "example": "positions_fractional=0,0,0;0.25,0.25,0.25;...",
        "status": "development",
        "syntax": "$aurl/?positions_fractional"
    }
]
```


**Fig. 2.** (a) The AFLUX schema for a particular set of keywords can be retrieved using the "schema" `directive` with the appropriate keywords as arguments. (b) The schema includes information on the property type referenced by each keyword, the units that the property is provided in, and a list of other keywords which can be used to verify the convergence and quality of the calculation used to obtain that particular property.

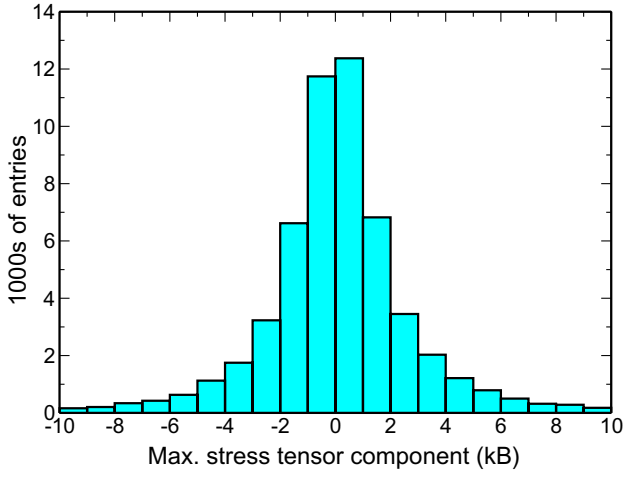



**Fig. 3.** Histogram showing the distribution of the stress tensor component with the largest absolute value for each entry in the AFLOW-ICSD catalog.

Additionally, GGA exchange correlation functionals, such as PBE [30], tend to systematically overestimate bond lengths and thus volume per atom [31], leading to a systematic shift from the experimentally measured volume per atom if the calculations have been properly converged.

## 4. Conclusion

In this article, we have presented AFLUX, a URI-based search API extending the original materials data oriented AFLOW-API. The syntax of the new materials language AFLUX (LUX for AFLOW) enables the construction of complex search queries and facilitates the creation of remote operations on the AFLOW.org repositories. The semantic of AFLUX is transparent and easy to adapt to other materials genome initiatives, such as the Materials Project, NoMaD, OQMD, and AiiDA. Implementation of an interface to the proposed Open Databases Integration for Materials Design (OPTiMaDe, www.optimade.org) common API for materials data is currently in progress.

## Acknowledgments

The authors thank Drs. O. Levy, I. Takeuchi, G. Hart, J. Carrete, J.J. Plata and N. Mingo for helpful discussions. This work is partially supported by DOD-ONR (N00014-13-1-0635, N00014-11-1-0136, N00014-09-1-0921), NIST #70NANB12H163 and by the Duke University—Center for Materials Genomics. C.T. and S.C. acknowledge partial support by DOE (DE-AC02-05CH11231), specifically the BES program under Grant #EDCBEE. C.O. acknowledges support from the National Science Foundation Graduate Research Fellowship under Grant No. DGF-1106401.

## Appendix A. Syntax diagrams

In the instantiation of an AFLUX process we accept summons that conform to the following syntax. The LUX syntax is described in ABNF (https://tools.ietf.org/html/rfc5234 and https://tools.ietf.org/html/rfc7405) (see Fig. 4).

```
Summons =
    (matchbook/directive) *(Binal Directive)
Matchbook =
    [Unary-Not]([Unary-Mute]Datum-string"
  ("Match")")/"("Matchbook")"/(Matchbook
  Binal Matchbook)
Match =
    [Unary-Not] ([Unary-Loose](Datum-string/
  Datum-number)[Unary-Loose]/"("Match")")/
  (Match Binal Match)
Directive =
    [Unary-Mute]Datum-string"("(Datum-string/Da
  tum-number) *(Binal(Datum-string/Datum-num
  ber))")"
```

## Appendix B. Example code for AFLUX programmatic access

This section includes an example code written in the python programming language which can be used to perform programmatic searches using AFLUX. This code was used to retrieve the data used to generate the histogram shown in Fig. 3.

```
#!/usr/bin/python3

import json, sys, os
from urllib.request import urlopen

SERVER="http://aflowlib.duke.edu"
API="/search/API/?"
SUMMONS="stress_tensor,$compound, $aurl,catalog
  ('icsd'),$paging(0)"

response=json.loads(urlopen(SERVER+API+SUM
  MONS).read().decode("utf-8"))
for datum in response:
    s_vals=[float(x) for x in datum
  ['stress_tensor'].split(",")]
    s_max=max(s_vals)
    s_min=min(s_vals)
    if(s_max >= abs(s_min)):
        mag=s_max
    else:
        mag=s_min
    print ("{0}, {1}".format(datum['auid'], mag))
```

## Appendix C. Table of properties and API keywords

This section includes the new keywords added to the AFLOW data API since the release of version 1.0 as described in Ref. 18. These include keywords which were added to facilitate verification as well as keywords that were added to provide access to data calculated using the AEL-AGL methodology for thermomechanical properties [26,27]. The `kpoints` keyword has also been upgraded from version 1.0 to include the number of kpoints per line segment for the electronic band structure calculation, and the new format is also described below. For each keyword, we list the description, type and the retrieval syntax. Full current information for any keyword in the AFLOW data API can be retrieved using the AFLUX `schema directive` described in Section 2.7 and Fig. 2.

### C.1. Optional materials keywords (alphabetic order)

- `ael_bulk_modulus_reuss`
  - *Description.* Returns AEL bulk modulus as calculated using the Reuss average.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `ael_bulk_modulus_reuss=105.315`
  - *Request syntax.* $`aurl/?ael_bulk_modulus_reuss`
- `ael_bulk_modulus_voigt`
  - *Description.* Returns AEL bulk modulus as calculated using the Voigt average.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `ael_bulk_modulus_voigt=105.315`
  - *Request syntax.* $`aurl/?ael_bulk_modulus_voigt`
- `ael_bulk_modulus_vrh`
  - *Description.* Returns AEL bulk modulus as calculated using the Voigt-Reuss-Hill (VRH) average.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `ael_bulk_modulus_vrh=105.315`
  - *Request syntax.* $`aurl/?ael_bulk_modulus_vrh`
- `ael_elastic_anisotropy`
  - *Description.* Returns AEL elastic anisotropy.
  - *Type.* `number`.
  - *Units.* dimensionless.
  - *Example.* `ael_elastic_anisotropy=0.000816153`
  - *Request syntax.* $`aurl/?ael_elastic_anisotropy`
- `ael_poisson_ratio`
  - *Description.* Returns AEL Poisson ratio.
  - *Type.* `number`.
  - *Units.* dimensionless.

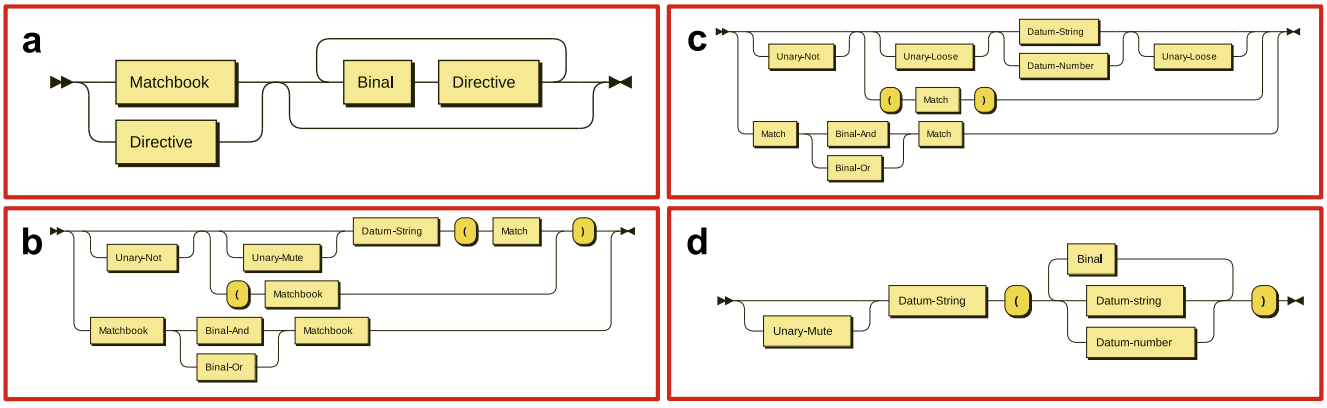



**Fig. 4.** Syntax diagrams for the LUX search-API: (a) Summons, (b) `Matchbook`, (c) Match, and (d) `Directive`.

  - *Example.* `ael_poisson_ratio=0.21599`
  - *Request syntax.* $`aurl/?ael_poisson_ratio`
- `ael_shear_modulus_reuss`
  - *Description.* Returns AEL shear modulus as calculated using the Reuss average.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `ael_shear_modulus_reuss=73.7868`
  - *Request syntax.* $`aurl/?ael_shear_modulus_reuss`
- `ael_shear_modulus_voigt`
  - *Description.* Returns AEL shear modulus as calculated using the Voigt average.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `ael_shear_modulus_voigt=73.7989`
  - *Request syntax.* $`aurl/?ael_shear_modulus_voigt`
- `ael_shear_modulus_vrh`
  - *Description.* Returns AEL shear modulus as calculated using the Voigt-Reuss-Hill (VRH) average.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `ael_shear_modulus_vrh=73.7929`
  - *Request syntax.* $`aurl/?ael_shear_modulus_vrh`
- `ael_speed_of_sound_average`
  - *Description.* Returns AEL average speed of sound calculated from the transverse and longitudinal speeds of sound.
  - *Type.* `number`.
  - *Units.* m/s.
  - *Example.* `ael_speed_of_sound_average=500.0`
  - *Request syntax.* $`aurl/?ael_speed_of_sound_average`
- `ael_speed_of_sound_longitudinal`
  - *Description.* Returns AEL speed of sound in the longitudinal direction.
  - *Type.* `number`.
  - *Units.* m/s.
  - *Example.* `ael_speed_of_sound_longitudinal=500.0`
  - *Request syntax.* $`aurl/?ael_speed_of_sound_longitudinal`
- `ael_speed_of_sound_transverse`
  - *Description.* Returns AEL speed of sound in the transverse direction.
  - *Type.* `number`.
  - *Units.* m/s.
  - *Example.* `ael_speed_of_sound_transverse=500.0`
  - *Request syntax.* $`aurl/?ael_speed_of_sound_transverse`
- `agl_acoustic_debye`
  - *Description.* Returns AGL acoustic Debye temperature.
  - *Type.* `number`.
  - *Units.* K.
  - *Example.* `agl_acoustic_debye=492`
  - *Request syntax.* $`aurl/?agl_acoustic_debye`
- `agl_bulk_modulus_isothermal_300K`
  - *Description.* Returns AGL isothermal bulk modulus at 300 K and zero pressure.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `agl_bulk_modulus_isothermal_300K=96.6`
  - *Request syntax.* $`aurl/?agl_bulk_modulus_isothermal_300K`
- `agl_bulk_modulus_static_300K`
  - *Description.* Returns AGL static bulk modulus at 300K and zero pressure.
  - *Type.* `number`.
  - *Units.* GPa.
  - *Example.* `agl_bulk_modulus_static_300K=99.59`
  - *Request syntax.* $`aurl/?agl_bulk_modulus_static_300K`
- `agl_debye`
  - *Description.* Returns AGL Debye temperature.
  - *Type.* `number`.
  - *Units.* K.
  - *Example.* `agl_debye=620`
  - *Request syntax.* $`aurl/?agl_debye`
- `agl_gruneisen`
  - *Description.* Returns AGL Grüneisen parameter.
  - *Type.* `number`.
  - *Units.* dimensionless.
  - *Example.* `agl_gruneisen=2.06`
  - *Request syntax.* $`aurl/?agl_gruneisen`
- `agl_heat_capacity_Cv_300K`
  - *Description.* Returns AGL heat capacity at constant volume ($C_V$) at 300 K and zero pressure.
  - *Type.* `number`.
  - *Units.* $k_B$/cell.
  - *Example.* `agl_heat_capacity_Cv_300K=4.901`
  - *Request syntax.* $`aurl/?agl_heat_capacity_Cv_300K`
- `agl_heat_capacity_Cp_300K`
  - *Description.* Returns AGL heat capacity at constant pressure ($C_p$) at 300 K and zero pressure.
  - *Type.* `number`.
  - *Units.* $k_B$/cell.
  - *Example.* `agl_heat_capacity_Cp_300K=5.502`
  - *Request syntax.* $`aurl/?agl_heat_capacity_Cp_300K`
- `agl_poisson_ratio_source`
  - *Description.* Returns source of Poisson ratio used to calculate Debye temperature in AGL. Possible sources include `ael_poisson_ratio_<value>`, in which case the Poisson ratio was calculated from first principles using AEL; `empirical_ratio_<value>`, in which case the value was taken from the literature; and `Cauchy_ratio_0.25`, in which case the default value of 0.25 of the Poisson ratio of a Cauchy solid was used.
  - *Type.* `string`.
  - *Example.* `agl_poisson_ratio_source=ael_poisson_ratio_0.193802`
  - *Request syntax.* $`aurl/?agl_poisson_ratio_source`
- `agl_thermal_conductivity_300K`
  - *Description.* Returns AGL thermal conductivity at 300 K.
  - *Type.* `number`.
  - *Units.* W/(m K).
  - *Example.* `agl_thermal_conductivity_300K=24.41`
  - *Request syntax.* $`aurl/?agl_thermal_conductivity_300K`
- `agl_thermal_expansion_300K`
  - *Description.* Returns AGL thermal expansion at 300 K and zero pressure.
  - *Type.* `number`.
  - *Units.* 1/K.
  - *Example.* `agl_thermal_expansion_300K=4.997e-05`
  - *Request syntax.* $`aurl/?agl_thermal_expansion_300K`
- `delta_electronic_energy_convergence`
  - *Description.* Convergence energy difference for the final electronic SCF step.
  - *Type.* `number`.
  - *Example.* `delta_electronic_energy_convergence=0.000071416`
  - *Request syntax.* $ `aurl/?delta_electronic_energy_convergence`
- `delta_electronic_energy_threshold`
  - *Description.* Convergence energy threshold for the electronic SCF loop.
  - *Type.* `number`.

  - *Example.* `delta_electronic_energy_threshold=0.0001`
  - *Request syntax.* `$aurl/?delta_electronic_energy_threshold`
- `kpoints`
  - *Description.* Set of **k**-point meshes uniquely identifying the various steps of the calculations, *e.g.*, relaxation, static and electronic band structure (specifying the **k**-space symmetry points of the structure and the number of points per path segment).
  - *Type.* Set of `numbers` and `strings` separated by "," and ";".
  - *Example.* `kpoints=10,10,10;16,16,16;Γ-X,X-W,W-K,K-Γ,Γ-L,L-U,U-W,W-L,L-K,U-X;20`
  - *Request syntax.* `$aurl/?kpoints`
- `pressure_residual`
  - *Description.* Returns the residual pressure for the simulation, *i.e.* the difference between the pressure specified in the input and the actual pressure achieved for the relaxation.
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, *e.g.*, kB or a.u. (Ry/Bohr) if the calculations were performed with VASP [32] or QE [33], respectively.
  - *Example.* `pressure_residual=10.0`
  - *Request syntax.* `$aurl/?pressure_residual`
- `Pulay_stress`
  - *Description.* Returns the Pulay stress for the simulation.
  - *Type.* `number`.
  - *Units.* Natural units of the `$code`, *e.g.*, kB or a.u. (Ry/Bohr) if the calculations were performed with VASP [32] or QE [33], respectively.
  - *Example.* `Pulay_stress=10.0`
  - *Request syntax.* `$aurl/?Pulay_stress`
- `stress_tensor`
  - *Description.* Returns the stress tensor obtained for the simulation.
  - *Type.* List of 9 `numbers` separated by commas, giving the elements of the stress tensor in the form $S_{xx}, S_{xy}, S_{xz}, S_{yx}, S_{yy}, S_{yz}, S_{zx}, S_{zy}, S_{zz}$.
  - *Units.* Natural units of the `$code`, *e.g.*, kB or a.u. (Ry/Bohr) if the calculations were performed with VASP [32] or QE [33], respectively.
  - *Example.* `stress_tensor=0.74,0,0,0,0.74,−0,0,−0,−4.42`
  - *Request syntax.* `$aurl/?stress_tensor`

## References


[1] S. Curtarolo, W. Setyawan, G.L.W. Hart, M. Jahnátek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M.J. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, AFLOW: an automatic framework for high-throughput materials discovery, Comput. Mater. Sci. 58 (2012) 218–226.

[2] K. Yang, C. Oses, S. Curtarolo, Modeling off-stoichiometry materials with a high-throughput ab-initio approach, Chem. Mater. 28 (2016) 6484–6492.

[3] C.E. Calderon, J.J. Plata, C. Toher, C. Oses, O. Levy, M. Fornari, A. Natan, M.J. Mehl, G.L.W. Hart, M. Buongiorno Nardelli, S. Curtarolo, The AFLOW standard for high-throughput materials science calculations, Comput. Mater. Sci. 108 (Part A) (2015) 233–238.

[4] O. Levy, M. Jahnátek, R.V. Chepulskii, G.L.W. Hart, S. Curtarolo, Ordered structures in Rhenium binary alloys from first-principles calculations, J. Am. Chem. Soc. 133 (2011) 158–163.

[5] O. Levy, G.L.W. Hart, S. Curtarolo, Structure maps for hcp metals from first-principles calculations, Phys. Rev. B 81 (2010) 174106.

[6] O. Levy, G.L.W. Hart, S. Curtarolo, Uncovering compounds by synergy of cluster expansion and high-throughput methods, J. Am. Chem. Soc. 132 (2010) 4830–4833.

[7] G.L.W. Hart, S. Curtarolo, T.B. Massalski, O. Levy, Comprehensive search for new phases and compounds in binary alloy systems based on platinum-group metals, using a computational first-principles approach, Phys. Rev. X 3 (2013) 041035.

[8] J. Carrete, W. Li, N. Mingo, S. Wang, S. Curtarolo, Finding unprecedentedly low-thermal-conductivity half-Heusler semiconductors via high-throughput materials modeling, Phys. Rev. X 4 (2014) 011019.

[9] J. Carrete, N. Mingo, S. Wang, S. Curtarolo, Nanograined half-Heusler semiconductors as advanced thermoelectrics: an ab initio high-throughput statistical study, Adv. Func. Mater. 24 (2014) 7427–7432.

[10] O. Isayev, D. Fourches, E.N. Muratov, C. Oses, K. Rasch, A. Tropsha, S. Curtarolo, Materials cartography: representing and mining materials space using structural and electronic fingerprints, Chem. Mater. 27 (2015) 735–743.

[11] A. van Roekeghem, J. Carrete, C. Oses, S. Curtarolo, N. Mingo, High-throughput computation of thermal conductivity of high-temperature solid phases: the case of oxide and fluoride perovskites, Phys. Rev. X 6 (2016) 041061.

[12] O. Isayev, C. Oses, C. Toher, E. Gossett, S. Curtarolo, A. Tropsha, Universal fragment descriptors for predicting properties of inorganic crystals, Nature Commun. 8 (2017) 15679.

[13] J. Yong, Y. Jiang, D. Usanmaz, S. Curtarolo, X. Zhang, L. Li, X. Pan, J. Shin, I. Takeuchi, R.L. Greene, Robust topological surface state of Kondo insulator $SmB_6$ thin films, Appl. Phys. Lett. 105 (2014) 222403.

[14] E. Perim, D. Lee, Y. Liu, C. Toher, P. Gong, Y. Li, W.N. Simmons, O. Levy, J.J. Vlassak, J. Schroers, S. Curtarolo, Spectral descriptors for bulk metallic glasses based on the thermodynamics of competing crystalline phases, Nature Commun. 7 (2016) 12315.

[15] S. Curtarolo, G.L.W. Hart, M. Buongiorno Nardelli, N. Mingo, S. Sanvito, O. Levy, The high-throughput highway to computational materials design, Nature Mater. 12 (2013) 191–201.

[16] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R.H. Taylor, L.J. Nelson, G.L.W. Hart, S. Sanvito, M. Buongiorno Nardelli, N. Mingo, O. Levy, AFLOWLIB.ORG: a distributed materials properties repository from high-throughput ab initio calculations, Comput. Mater. Sci. 58 (2012) 227–235.

[17] W. Setyawan, S. Curtarolo, High-throughput electronic band structure calculations: challenges and tools, Comput. Mater. Sci. 49 (2010) 299–312.

[18] R.H. Taylor, F. Rose, C. Toher, O. Levy, K. Yang, M. Buongiorno Nardelli, S. Curtarolo, A RESTful API for exchanging materials data in the AFLOWLIB.org consortium, Comput. Mater. Sci. 93 (2014) 178–192.

[19] M. Scheffler, C. Draxl, Computer Center of the Max-Planck Society, Garching, The NoMaD Repository, 2014 <http://nomad-repository.eu>.

[20] A. Jain, G. Hautier, C.J. Moore, S.P. Ong, C.C. Fischer, T. Mueller, K.A. Persson, G. Ceder, A high-throughput infrastructure for density functional theory calculations, Comput. Mater. Sci. 50 (2011) 2295–2310.

[21] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD), JOM 65 (2013) 1501–1509.

[22] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, AiiDA, 2016 <http://www.aiida.net>.

[23] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, B. Kozinsky, AiiDA: automated interactive infrastructure and database for computational science, Comput. Mater. Sci. 111 (2016) 218–230.

[24] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, 2005 <https://tools.ietf.org/html/rfc3986>.

[25] D. Crocker, P. Overell, Augmented BNF for Syntax Specifications: ABNF, 2008 <https://tools.ietf.org/html/rfc5234>.

[26] C. Toher, J.J. Plata, O. Levy, M. de Jong, M.D. Asta, M. Buongiorno Nardelli, S. Curtarolo, High-throughput computational screening of thermal conductivity, Debye temperature, and Grüneisen parameter using a quasiharmonic Debye model, Phys. Rev. B 90 (2014) 174107.

[27] C. Toher, C. Oses, J.J. Plata, D. Hicks, F. Rose, O. Levy, M. de Jong, M.D. Asta, M. Fornari, M. Buongiorno Nardelli, S. Curtarolo, Combining the AFLOW GIBBS and Elastic Libraries to efficiently and robustly screen thermo-mechanical properties of solids, Phys. Rev. Mater. (in press), arXiv:1611.05714.

[28] M.A. Blanco, E. Francisco, V. Luaña, GIBBS: isothermal-isobaric thermodynamics of solids from energy curves using a quasi-harmonic Debye model, Comput. Phys. Commun. 158 (2004) 57–72.

[29] G. Bergerhoff, R. Hundt, R. Sievers, I.D. Brown, The inorganic crystal structure data base, J. Chem. Inf. Comput. Sci. 23 (1983) 66–69.

[30] J.P. Perdew, K. Burke, M. Ernzerhof, Generalized gradient approximation made simple, Phys. Rev. Lett. 77 (1996) 3865–3868.

[31] L. He, F. Liu, G. Hautier, M.J.T. Oliveira, M.A.L. Marques, F.D. Vila, J.J. Rehr, G.-M. Rignanese, A. Zhou, Accuracy of generalized gradient approximation functionals for density-functional perturbation theory calculations, Phys. Rev. B 89 (2014) 064305.

[32] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (1996) 11169–11186.

[33] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A.P. Seitsonen, A. Smogunov, P. Umari, R.M. Wentzcovitch, QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials, J. Phys.: Condens. Matter 21 (2009) 395502.