# Contents

1 What is bio.tools? 3
   1.1 Objectives .......................................................... 3
   1.2 Scope ............................................................... 3
   1.3 Technical components .......................................... 4
   1.4 bio.tools Tool Identifiers ....................................... 4
   1.5 Docs overview ................................................... 5
   1.6 Getting involved: a quickstart guide ......................... 5

2 Contributors Guide 7
   2.1 bio.tools community site ..................................... 7
   2.2 Feature requests & issues ..................................... 7
   2.3 Mailing list ...................................................... 7
   2.4 Hangouts .......................................................... 8
   2.5 Task management ............................................... 8
   2.6 Editing tool descriptions .................................... 8

3 Quickstart guide 9
   3.1 Create an account ............................................... 9
   3.2 Add content ....................................................... 9
   3.3 Update a resource ............................................... 11
   3.4 Remove a resource .............................................. 12
   3.5 Search for a tool ............................................... 12
   3.6 References ....................................................... 12

4 Curators Guide 13
   4.1 General guidelines ............................................. 14
   4.2 Attribute guidelines ........................................... 17
   4.3 Tool type guidelines .......................................... 38
   4.4 Further guidelines (bio.tools admin only) ................. 43

5 Community-specific guidelines 45
   5.1 IFB tools .......................................................... 45
   5.2 Workflow composition (Lorentz workshop) .................. 66

6 Editors Guide 77
   6.1 Background ...................................................... 77
   6.2 Candidate thematic editors .................................... 78
17 Installing bio.tools on your system  
  17.1 1. Download and Install Docker  
  17.2 2. Clone the repo  
  17.3 3.1 The short(er) setup  
  17.4 3.2 The longer setup  
  17.5 4. Useful information  
  17.6 4.0 Basic usage  
  17.7 4.1 Local dev  
  17.8 4.2 Update EDAM  
  17.9 4.3 Local email setup  
  17.10 4.4 Docker notes  
  17.11 API Guidelines  

18 License  

19 Publications  
  19.1 Citation  

20 Support  

21 These docs  
  21.1 reStructuredText links
This is the documentation for bio.tools.

Contents:
What is bio.tools?

bio.tools is a portal to bioinformatics resources worldwide, aimed to help bioinformaticians and scientists:

• find, understand, compare and select resources == discovery
• use and connect them in workflows == (inter)operability

Our vision is to be the sustainable primary archive for basic tool metadata, providing a persistent reference to high-quality (curated and verified) “canonical” descriptions of unique tools, with information about their provision via online services and various downloadable artefacts, and including entries for different versions of a tool, where these have major functional differences.

1.1 Objectives

Our main objectives are:

• build and maintain a comprehensive registry of high-quality software metadata / descriptions
• provide a web portal enabling registration, editing, search and discovery of the registry content
• support a community for the sustainable maintenance of the registry content and development of the portal features
• expose results of tool performance benchmarking, online service monitoring and other metrics of software and service quality
• integrate the registry with popular workbench environments in a way that improves resource interoperability
• support registry stakeholders including tool providers and end-users

1.2 Scope

bio.tools scope is application software with well-defined data processing functions (inputs, outputs and operations). bio.tools includes a broad range of software types including tools available for immediate use as online services, or
in a form which which you can download, install, configure and run yourself. This includes simple tools with one or a few closely related functions, and complex, multimodal tools with many functions. It also includes executable workflows, database portals and Web APIs.

### 1.3 Technical components

- **biotoolsSchema** is a description model for bioinformatics software. It is a formalised XML schema (XSD) which defines 50 important scientific, technical and administrative attributes. It defines what attributes may be specified in a bio.tools entry, a precise syntax for those descriptions, and controlled vocabularies for consistent description of technical aspects such as software license and software type.

- **EDAM ontology** is an ontology of well-established, familiar concepts that are prevalent within bioinformatics and computational biology, including types of data and data identifiers, data formats, operations and topics. It defines precise semantics for the scientific description of software registered in bio.tools.

- **Curation guidelines** describe how each attribute should be specified, *i.e.* concern the quality of an entry. The guidelines go beyond the syntactic and semantic constraints defined by biotoolsSchema and EDAM, and are part of broader tool information standards being adopted by bio.tools.

- **Tool Cards** e.g. https://bio.tools/signalp provide key information at a glance for registered tools. Tool cards have human-friendly, persistent URLs which include the unique tool identifier (“signalp” in this case). The identifier is assigned upon registration is a URL-safe derivative of (normally identical to) the supplied tool name.

- **Query interfaces** available at https://bio.tools help bio.tools end-users with tool discovery and include the search bar, a compact “mini-card” view and a detailed “grid” view. See the Quickstart Guide.

- **Registration interface** enables manual creation of valid registry content and editing, including graphical editing via tabbed panes and an interactive JSON editor with inline error reporting. It is available to logged-on users via “Menu ... Add content”. See the Quickstart Guide.

- **bio.tools API** provides programmatic means to query, add and edit registry content.

- **bio.tools metrics** available at https://bio.tools/stats include registry growth, contributors, annotation breakdown etc.

### 1.4 bio.tools Tool Identifiers

Each bio.tools entry is assigned a unique identifier (**biotoolsID**): a manually verified, URL-safe version of (normally identical to) the supplied tool name. The IDs are used in persistent URLs, resolving to Tool Cards of essential information. The recommended short-form is a compact URI (CURIE), which is resolvable in Identifiers.org.

<table>
<thead>
<tr>
<th><strong>biotoolsID</strong></th>
<th><strong>Example</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CURIE</td>
<td>biotools:signalp</td>
</tr>
<tr>
<td>Identifiers.org</td>
<td><a href="http://identifiers.org/biotools/signalp">http://identifiers.org/biotools/signalp</a></td>
</tr>
<tr>
<td>Tool Card URL</td>
<td><a href="https://bio.tools/signalp">https://bio.tools/signalp</a></td>
</tr>
</tbody>
</table>

Registered software which, for one reason or another, is no longer operational, retain their ID and URL but are marked as obsolete. Hence, descriptions of legacy resources are archived.
1.5 Docs overview

- Contributors Guide - how to get involved (please do!)
- Quickstart Guide - quick guide on how to use the bio.tools user interfaces.
- Curators Guide - how to create a high quality bio.tools entry.
- Editors Guide - thematic editorships to improve bio.tools in scientific areas.
- Documentors Guide - how to edit the bio.tools docs.
- API reference - bio.tools API docs.
- Hangouts - monthly coordination meetings (you’re welcome to join!)
- Roadmap - technical plans for the next year
- Studentships - bio.tools studentship scheme for curation-focussed mini-projects.
- GitHub projects - open projects of relevance to bio.tools.

bio.tools development is supported by ELIXIR - the European Research Infrastructure for life science information. bio.tools content is freely available to all under open license.

1.6 Getting involved : a quickstart guide

1. Read the docs but especially the contributors guide.
2. GitHub is used for task and issue tracking, see the bio.tools and EDAM organisations, in particular the biotools-registry and edamontology projects. Email Jon if you want to join.
3. Join the mailing lists but note that most of the discussion is done via GitHub. Important announcements and some discussion are done via registry-dev (see below)
4. Members of registry-dev share a mailing list and calendar, but there are some implications of joining. Email Jon if you want to join.
5. We run hangouts (coordination meetings) as required - mostly for technical people routinely involved with bio.tools curation or software development. To suggest or join these calls email Jon.
6. Dive in at the deep end! There are no end of ongoing sub-projects and tasks to get involved with, see GitHub (at above links) or email Registry Support in the 1st instance to get orientated.
2.1 bio.tools community site

GitHub is used for sharing code and data for all bio.tools-related projects:

- https://github.com/bio-tools/
- https://github.com/bio-tools/biotoolsschema
- https://github.com/bio-tools/biotoolsregistry
- https://github.com/edamontology/edamontology

2.2 Feature requests & issues

GitHub is used to track fine-grained issues and is the preferred way to make bio.tools feature requests, content suggestions, EDAM term requests, and bug reports:

- https://github.com/bio-tools/biotoolsregistry/issues
- https://github.com/bio-tools/biotoolsschema/issues
- https://github.com/edamontology/edamontology/issues

Note: GitHub is the primary means for technical coordination: collaborators are encouraged to browse tasks, review priorities, make comments and add new tasks.

2.3 Mailing list

Please use the appropriate mailing list:
• **registry** for general discussions on *bio.tools*
• **registry-support** for questions and help on *bio.tools*
• **edam** for general discussion and help on EDAM
• **registry-announce** or **edam-announce** for low-traffic announcements
• **registry-core** and **edam-dev** for technical discussion amongst developers

To send mail:
• registry@elixir-dk.org
• registry-support@elixir-dk.org
• edam@elixir-dk.org

## 2.4 Hangouts

Coordination meetings are organised as required. The hangouts usually have an open agenda and respond to current critical needs. Technical representatives of ELIXIR-DK institutes routinely attend and everyone is very welcome: if you’d like to join mail Jon Ison.

## 2.5 Task management

High-level project management tasks concerning *bio.tools* are managed in **sifterapp** (moderated by Jon Ison):
• https://biotools.sifterapp.com

If you’d like a sifterapp account, please email Jon.

## 2.6 Editing tool descriptions

You can contribute to *bio.tools* directly:
• register your own (or other people’s) tools
• request edit rights (via a button on the *bio.tools* Tool Cards) on entries where tools are already registered
• request ownership (again via a button) of entries, if you are the developer of the tool described but it has been registered by someone else
• ask for help via the mailing lists, especially in case you have many tools to add or edits to make

*bio.tools* benefits from the support of ELIXIR Nodes: collections of research institutes from a member country that provide the resources and services that are part of ELIXIR.
Quickstart guide

This guide aims to help you through the different steps to add entries to bio.tools.

**Note:** If you find a bug, have any questions or suggestions, please get in touch.

### 3.1 Create an account

Creating an account on bio.tools is very quick and simple. Just click on the Sign-up button at the top-right corner of the page. Then you just need to give a username, your email address and a password to get your account done.

### 3.2 Add content

Everyone is welcome to add their own and other resources to bio.tools. Once your account is created, you can start adding your content by clicking on *Menu ... Add content*.

The description of an new entry is handled by different tabs within the registration interface that are described below.

At any moment, you can check the validity of your information by clicking on Validate and save it by clicking on Save.

**Note:** Saving the entry makes it directly available online. If you want to save what you have done without publishing it, the only way currently is to go to the *JSON* tab and save the .json file locally.

**Important:** The minimum information required (name, description and homepage URL) is marked with a red asterix * in the registration interface.
It’s recommended - especially if you have many tools to add - to read the Curators Guide first.

### 3.2.1 Summary

For this first part, you give the basic descriptors. This includes the name of your resource with a description, its version and a homepage URL. A unique ID is automatically generated from the name.

**Note:** A unique identifier (bio.tools toolID) is a URL-safe version of the supplied resource name. It’s used in persistent URLs to bio.tools “Tool Cards”, e.g. for the tool ID of “signalp”:

- http://bio.tools/tool/signalp
- http://bio.tools/t/signalp
- http://bio.tools/signalp

Currently, if you want to change the ID you have to mail Registry Support. In future, the ID will be editable at registration time.

### 3.2.2 Function

This is where you describe the functionality of the tool based on the EDAM ontology\(^1\). The functionality is captured in a diagram on the Tool Cards that look like this:

```
Input

Data type (Data format) → Function → Output

Data type (Data format)
```

In each box, you can add as many fields as you want. You can also add a general comment about the function (*this is particularly useful when your entry has several functions*). It’s highly recommended to read up about tool functions before filling this section.

**Note:** It can be difficult to find the right terms to describe a tool’s operation(s), input(s) or output(s). You can use OLS EDAM, BioPortal and EDAM Browser to browse EDAM and find the terms you need, or request new terms via GitHub. Improvements (including term requests) to the term picker in bio.tools are planned.

### 3.2.3 Labels

In this part, you can tell more about your tool:

- What **type** of resource it is (Command-line tool, Web application etc.)
- Relevant **topic(s)** the tool fits with (from the EDAM ontology\(^1\)).
- In which **operating system** it is possible to use it.

• The language used to develop the tool, its license and maturity.
• The accessibility of your tool and its cost.

You can also assign your tool to an arbitrary collection which can be useful for grouping together related tools.

### 3.2.4 Links

It is the place where your add links that do not belong to Download or Documentation. For instance, a link to a mailing list, mirror or repository (full list available on the drop-down menu of Link type).

### 3.2.5 Download

You can here share all the different download links you want. It can be many different kind such as binaries, source code, biological data, test data etc. (see the Download type drop-down menu).

### 3.2.6 Documentation

Make your different documentations for your tool available here. Again, you can assign type of documentation using Documentation type.

### 3.2.7 Publications

Share the different publications of the tool, which can be the primary publication (the one to cite when the tool is used), but also reviews or secondary references (see Publication type). You can use either the PubMed Central ID (PMCID), the PubMed ID (PMID) or the Digital Object ID (DOI) - DOI is preferred.

### 3.2.8 Credits & Support

Credits include all type of entities that contributed to the development, maintenance or provision of the resource. Credits can have an Entity type (Person, Institute etc.) and an Entity role (Developer, Documentor etc.). Use the role of Primary contact to indicate preferred contact details.

### 3.2.9 JSON

This is all the information you gave about your tool, formatted in JSON format.

### 3.2.10 Permissions

You can decide to make the entry either editable only by yourself, a list of users or anyone.

### 3.3 Update a resource

You’ll see up to three different buttons at the bottom right of the Tool Card:
• Click on *Update this record* to edit it
• Click on *Request editing rights* if you want to get edit rights to the entry
• Click on *Request ownership* if you want to claim ownership of the entry

..Note:: *bio.tools* entries are owned by the individuals who created them. Ownerships may grant edit rights, or transfer ownership of their entries to others.

### 3.4 Remove a resource

From the tool card, click on update this record. Then you can remove the entry by clicking on the remove button.

**Warning:** Removing an entry is definitive. There’s no way back (other than emailing Registry Support).

### 3.5 Search for a tool

Coming soon…

### 3.6 References
Attention:

- guidelines for bio.tools curators, including EDAM annotation guidelines.
- to make suggestions about these guidelines please add comments via GitHub
- for curation advice mail registry-support

**bio.tools** includes all types of bioinformatics tools - application software with well-defined data processing functions (inputs, outputs and operations). This ranges from simple tools with a single primary function, to complex, multimodal tools with many distinct functions. Tools may be available for immediate use as online services, or in a form which you can download, install, configure and run yourself.

Usually, a bio.tools entry describes a discrete tool. Some entries describe collections of tools, such as software suites. The scope, *i.e.* the types of tools that may be included, and the attributes for their description, are defined in biotoolsSchema which uses the EDAM ontology as a source of terms for the tool scientific description. These curation guidelines describe how to create a high quality tool description, above and beyond the syntactic and semantic constraints that are defined in biotoolsSchema and EDAM.

- general guidelines include basic considerations, annotation of tool functions and the use of EDAM. You should read these first of all.
- guidelines on specific attributes defined in the biotoolsSchema
- guidelines specific to individual types of tools

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119:

- “MUST”, “REQUIRED” or “SHALL” mean that the guideline is an absolute requirement of the specification.
- “MUST NOT” or “SHALL NOT” mean that the guideline is an absolute prohibition of the specification.
- “SHOULD” or “RECOMMENDED” mean that there may exist valid reasons in particular circumstances to ignore a particular guideline, but the full implications must be understood and carefully weighed before doing so.
• “SHOULD NOT” or the phrase “NOT RECOMMENDED” mean that there may exist valid reasons in particular circumstances when acting contrary to the guideline is acceptable or even useful, but the full implications should be understood and the case carefully weighed before doing so.

• “MAY” or “OPTIONAL” mean that the guideline is truly optional; you can choose to follow it or not.

Note: The guidelines are a key component of an emerging information standard for tools being adopted by bio.tools, as a basis to monitor content and label bio.tools entries.

- automatically verified guidelines are (or will be) checked via automated periodic QC of the bio.tools system
- manually verified guidelines are checked via manual QC performed by trusted curators (bio.tools admin, entry owners etc.)
- advice given in boxes (notes, tips, caution etc. are not verified)

4.1 General guidelines

4.1.1 Before you start

Consider the following before creating a bio.tools entry:

1. Are one or more entries required to describe the software?
   - workbenches and other suites often require multiple entries.
   - tools with multiple interfaces (e.g. Command-line tool, Web API, Web service and Web application) SHOULD be described by a single entry unless these interfaces provide fundamental functional differences (see Tool functions below).
   - if in doubt, mail registry-support.

2. What tool types apply?
   - one or more tool types may be assigned in a single entry reflecting different facets of the software described by the entry.
   - read the tool type-specific guidelines before you create the tool.

3. What if the software is already registered?
   - if you’re the rightful owner of the entry (i.e. the tool developer or provider of an online service) then request ownership of it
   - otherwise, request edit rights

   Make these requests using the buttons at the bottom of the Tool Card (see e.g. https://bio.tools/signalp).

   If you plan to register multiple entries en masse, please discuss this first with bio.tools admin.

4. Are there version-specific considerations?
   - as a rule, a bio.tools entry SHOULD describe the latest version available at the time of registration and SHOULD be updated, as required, for subsequent releases.
   - if a new version has fundamental functional differences (see Tool functions below) it MAY be registered as an entirely new entry. In such cases, follow carefully the guidelines for tool name and version annotations.

5. Plan how to describe the tool functions.

6. Read the general EDAM annotations guidelines.
### 4.1.2 Tool functions

`bio.tools` uses a model of software (see below) defined within `biotoolsSchema`. A tool can have one or more basic functions (modes of operation), each function performing one or more specific operation (e.g., “Sequence alignment”), and may have one or more primary inputs and outputs, each of a defined type of data and listing supported format(s).

Plan how to describe the software:

- identify the distinct functions (modes of operation) and the individual operations associated with each one. Typically different functions (modes) perform different operations and for well documented tools, this is usually obvious. If in any doubt mail registry-support.

- as a general rule, if the tool allows an option between doing one thing or another, then you **MUST** annotate the operations as distinct functions. If in contrast a tool always does one or more things, then you **MUST** annotate
these as distinct operations within a single function.

- **bio.tools** aims for fairly coarse-grained description, *i.e.* you **SHOULD** only specify the primary functions and operations, from a typical end-user perspective. If a tool happens to perform some operation internally, but this is secondary to its advertised purpose, then you **SHOULD NOT** annotate it. If in doubt mail registry-support

- this holds for input and output too, *e.g.* a sequence alignment tool would be annotated as reading sequences (input), and writing a sequence alignment (output), but not with gap insertion and extension penalties, or other parameters.

- many tools allow a primary input or output to be specified in a number of alternative ways, *e.g.* a sequence input that may be specified via a sequence identifier, or as a literal sequence. In such cases, you **MAY** annotate the alternatives as distinct functions (see above). If specifying just one alternative, you **SHOULD** use the EDAM Data concept for the type of data, rather than identifier.

**Note:** A future refactoring may improve the modelling for alternative specification of inputs and outputs, by allowing multiple data+format couplets for a given input or output. If this is done, the proposed guideline would be:

- you **MAY** annotate all the commonly used alternatives and, if specifying alternatives, **MUST** annotate these as distinct data + format couplets within a single input or output.

- many inputs and outputs are complex, with individual data files containing multiple types of data. You **MUST** select the single EDAM Data term that best describes an input or output (see EDAM annotations below) and **MUST NOT** specify multiple EDAM Data terms describing different facets of the data.

Input on this issue is welcomed via GitHub.

### 4.1.3 EDAM annotations

The **EDAM ontology** is used to annotate applicable topics, operations, and the type and format of inputs and outputs. The general guidelines below apply for all EDAM annotations.

1. **MUST NOT** use “organisational” EDAM concepts *e.g.* Topic of “Topic” or Operation of “Operation” (see note below)

2. **SHOULD** use the most specific term(s) available, bearing in mind some concepts are necessarily overlapping or general. If multiple sibling terms are applicable (*i.e.* terms under a common parent), the parent term may be applicable.

3. **SHOULD NOT** use both a term and its parent or other ancestor, when annotating a single attribute. An exception would be a tool which *e.g.* performs some general Sequence analysis operations but specialises on Protein feature detection.

**Tip:** If you’re struggling to find the terms you need, or the meaning of a term is not obvious, search EDAM using the browsers below (they have different functionalities). Multiple searches using synonyms, alternative spellings *etc.* can help.

- EBI OLS browser
- NCBO BioPortal browser
- EDAM ontology browser from IFB
- EDAM Tool Annotator Demo

If you cannot find the right term, request it’s added to EDAM via GitHub but first read the guidelines on how to request a term.
Note: It currently takes some time from requesting new EDAM terms for these to be supported in bio.tools. In future, you’ll be able to request terms directly via the bio.tools registration interface and these terms will become immediately available for use, albeit subject to approval and possible change before inclusion in EDAM and bio.tools.

Note: Some high-level “organisational” concepts defined in EDAM are intended primarily to structure the hierarchy, and are not intended for annotation in bio.tools. They are defined in EDAM.owl via “<usageGuideline>Not recommended for annotation in bio.tools.</usageGuideline>”. Such tips are visible in the OLS and BioPortal browsers.

4.2 Attribute guidelines

Guidelines below are organised into sections as they appear in the bio.tools registration user interface

4.2.1 Summary group

Basic information about the software.

Name (tool)

Canonical software name assigned by the software developer or service provider, e.g. “needle”

- 1. MUST use name in common use, e.g. in the tool homepage and publication.
- 2. MUST use short form if available e.g. ExPASy not ExPASy Bioinformatics Resource Portal.
- 3. MUST NOT include general or technical terms (“software”, “application”, “server”, “service”, “SOAP”, “REST”, “RESTful” etc.) unless these are part of the common name
- 4. MUST NOT misappropriate the names of other tools, e.g. there are many online BLAST services besides the original NCBI BLAST tool; calling any of them “BLAST” would be wrong
- 5. MUST NOT include version or status information including terms like “alpha”, “beta” etc. - unless this is part of common name
- 6. SHOULD preserve capitalisation e.g. ExPASy not expasy.
- 7. SHOULD follow the naming patterns (see below)

Note:
- see the syntax guidelines.

Note: Naming patterns

For database portals use the pattern:

name (acronym) e.g. The Protein Databank (PDB)

- a common abbreviation can be given instead of an acronym
- if no common acronym or abbreviation exists, omit this part: do not invent one!
For tools that simply wrap or provide an interface to some other tool, including Web APIs (REST), Web services (SOAP+WSDL), and web applications over command-line tools, use the pattern:

\{collectionName\} toolName \{API|WS\}\{\ (providerName)\}  \textit{e.g.}  \textit{EMBOSS water API} (ebi)

where:

- \textit{collectionName} is the name of suite, workbench or other collection the underlying tool is from (if applicable)
- \textit{toolName} is the canonical name of the underlying tool
- use \textit{API} for Web APIs or \textit{WS} for Web services
- \textit{providerName} is the name of the institute providing the online service (if applicable)

If in exceptional cases (\textit{i.e.} when registering, as separate entries, versions of a tool with fundamental differences), substitute for \textit{toolName} in the pattern above:

\textit{toolname versionID} \textit{e.g.} \textit{FindPeaks 3.1}

where \textit{versionID} is the version number.

\textbf{Tip:}

- in case of multiple related entries be consistent, \textit{e.g.} \textit{Open PHACTS} and \textit{Open PHACTS API}
- be wary of names that are very long (>25 characters). If shortening the name is necessary, don’t truncate it in a way (\textit{e.g.} within the middle of a word) that would render it meaningless or unintuitive

\textbf{Description}

\textit{Textual description of the software, e.g. “needle reads two input sequences and writes their optimal global sequence alignment to file. It uses the Needleman-Wunsch alignment algorithm to find the optimum alignment (including gaps) of two sequences along their entire length. The algorithm uses a dynamic programming method to ensure the alignment is optimum, by exploring all possible alignments and choosing the best.”}

- 1. \textbf{MUST} provide a concise summary of purpose / function of the tool
- 2. \textbf{MUST} begin with a capital letter and end with a period (‘.’)
- 3. \textbf{SHOULD NOT} include any of the following, \textit{unless} essential to distinguish the tool from other bio.tool entries:
  - provenance information \textit{e.g.} software provider, institute or person name
  - describe how good the software is (mentions of applicability are OK)
- 4. \textbf{SHOULD NOT} include URLs
- 5. \textbf{SHOULD NOT} include DOIs

\textbf{Note:}

- see the \textit{syntax guidelines}. 

---

\textbf{Chapter 4. Curators Guide}
Homepage

Homepage of the software, or some URL that best serves this purpose, e.g. “http://emboss.open-bio.org/rel/rel6/apps/needle.html”

- 1. MUST resolve to a web page from the developer / provider that most specifically describes the tool
- 2. SHOULD NOT specify an FTP site unless nothing else is available.
- 3. MAY specify a repository if no better alternative is available.

Note:
- see the syntax guidelines.

Tip: In case a tool lacks it’s own website, a URL of it’s code repository is OK. Do not use a general URL such as an institutional homepage, unless nothing better is available.

Version (tool)

Version information (typically a version number) of the software applicable to this bio.tools entry, e.g. “6.4.0.0”

- 1. MUST correctly identify the tool version as described by the other attributes (see note below)
- 2. MUST specify exactly the public version label in common use
- 3. MUST NOT include tokens such as “v”, “ver”, “version”, “rel”, “release” etc., unless these are part of the public version label
- 4. MAY identify all tool versions which are applicable to the entry
- 5. MAY specify a version for database portals and web applications, but only if this is used in the common name

Note:
- see the syntax guidelines.

Important:

Care is needed to ensure annotations correspond to the indicated tool version.
- only change the version if you’re sure there’s no fundamental change to the specified tool functions (operations, inputs and outputs)
- if there are fundamental changes, update the tool function annotation
- do not assume version “1” in case the version number is not readily findable

Tip: One or more version fields may be specified, and each - in principle - allows flexible specification of version information including single versions, ranges, lists and lists including ranges, e.g.:

- 1.1
- beta01

4.2. Attribute guidelines
• 2.0 - 2.7
• 1.1, 1.2.1, 1.4, v5
• 1.1 - 1.4, 2.0-alpha, 2.0-beta-01 - 2.0-beta-04, 2.0.0
• etc.

We recommend to keep things simple (one version label per field by default) and pragmatic (using version ranges where desirable).

**Other IDs**

A *unique identifier of the software, typically assigned by an ID-assignment authority other than bio.tools, e.g. “RRID:SCR_015644”*

• 1. MUST correctly identify the same tool as indicated by the bio.toolsID
• 2. MUST include version information if IDs for multiple different versions are specified
• 3. MAY specify the type of identifier (see below)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doi</td>
<td>Digital Object Identifier of the software assigned (typically) by the software developer or service provider.</td>
</tr>
<tr>
<td>rrid</td>
<td>Research Resource Identifier as used by the NIH-supported Resource Identification Portal (<a href="https://scicrunch.org/resources">https://scicrunch.org/resources</a>).</td>
</tr>
<tr>
<td>cpe</td>
<td>Common Platform Enumeration (CPE) identifier as listed in the CPE dictionary (<a href="https://cpe.mitre.org/dictionary/">https://cpe.mitre.org/dictionary/</a>).</td>
</tr>
<tr>
<td>biotoolsCURIE</td>
<td>bio.tools CURIE (secondary identifier).</td>
</tr>
</tbody>
</table>

**Note:**

• see the syntax guidelines.

**Attention:** Alternative IDs of type biotoolsCURIE are set (and can only be changed) by bio.tools admin. They allow bio.tools to support multiple biotoolsIDs (hence resolvable Tool Card URLs) for a single tool; this done in exceptional circumstances only, e.g. the name of a tool is changed.

**Value**

*Value of tool identifier, e.g. “RRID:SCR_001156”*

• 1. MUST specify a valid identifier for the tool.

**Type (otherID)**

*Type of tool identifier, e.g. “rrid”*

• 1. MAY specify the applicable type, in terms from a controlled vocabulary (see below) - although this should not normally be necessary
Version (otherID)

Version information (typically a version number) of the software applicable to this identifier, e.g. “1.4”

- 1. MUST correctly identify the applicable tool version
- 2. MUST follow the general guidelines for version

4.2.2 Function group

Details of a function (i.e. mode of operation) the software provides, expressed in concepts from the EDAM ontology.

Operation

The basic operation(s) performed by this software function (EDAM Operation), e.g. “‘Protein signal peptide detection’ (http://edamontology.org/operation_0418)”

- 1. MUST correctly specify operations performed by the tool, or (if version is indicated), those specific version(s) of the tool
- 2. MUST be correctly organised into multiple functions, in case the tool has multiple modes of operation (see guidelines for tool functions).
- 3. SHOULD describe all the primary operations performed by that tool and SHOULD NOT describe secondary / minor operations: if in any doubt, mail registry-support.

Attention:
- see the general guidelines for EDAM annotations.

Note:
- see the syntax guidelines.

Data type (input and output data)

Type of primary input / output data (if any) e.g. “‘Sequence’ (http://edamontology.org/data_2044)”

- 1. MUST correctly specify types of input or output data processed by the tool, or (if version is indicated), those specific version(s) of the tool
- 2. MUST be correctly associated with the operation(s); for each function in case the tool has multiple modes of operation (see guidelines for tool functions).
- 3. SHOULD describe all the primary inputs and outputs of the tool and SHOULD NOT describe secondary / minor inputs and outputs: if in any doubt, mail registry-support.

Attention:
- see the general guidelines for EDAM annotations.
Tip:
- many tools allow a primary input to be specified in a number of alternative ways, the common case being a sequence input that may be specified via a sequence identifier, or by typing in a literal sequence. In such cases, annotate the input using the EDAM Data concept for the type of data, not the identifier.

Note:
- see the syntax guidelines for input and output

Data format (input and output data)

Allowed format(s) of primary inputs/outputs e.g. “‘FASTA’ (http://edamontology.org/format_1929)”

- 1. MUST correctly specify data formats supported on input or output by the tool, or (if version) is indicated, those specific version(s) of the tool
- 2. MUST be correctly associated with the data type of an input or output (see guidelines for tool functions).
- 3. SHOULD describe the primary data formats and MAY exhaustively describe all formats: if in any doubt, mail registry-support.

Attention: see the general guidelines for EDAM annotations.

Note:
- see the syntax guidelines.

Note (function)

Concise comment about this function, if not apparent from the software description and EDAM annotations, e.g. “This option is slower, but more precise.”

- 1. MUST not duplicate what is already apparent from the EDAM annotations
- 2. SHOULD be concise and summarise only critical usage information
- 3. SHOULD NOT duplicate online documentation; give a link if necessary

Note:
- see the syntax guidelines.

Command

Relevant command, command-line fragment or option for executing this function / running the tool in this mode, e.g “-s best”

- 1. MUST specify precisely a command, command-line fragment or option specified in the tool documentation
• 2. **MUST** be correctly associated with a function (the command must be used to invoke that specific tool function)

**Note:**

• see the syntax guidelines.

### 4.2.3 Labels group

*Miscellaneous scientific, technical and administrative details of the software, expressed in terms from controlled vocabularies.*

**Tool type**

*The type of application software: a discrete software entity can have more than one type, e.g. “Command-line tool, Web application”*

• 1. **MUST** specify all types that are applicable, in terms from a controlled vocabulary (see below)
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bioinformatics portal</td>
<td>web site providing a platform/portal to multiple resources used for research in a focused area, including biological databases, web applications, training resources and so on.</td>
</tr>
<tr>
<td>Command-line tool</td>
<td>A tool with a text-based (command-line) interface.</td>
</tr>
<tr>
<td>Database portal</td>
<td>A Web application, suite or workbench providing a portal to a biological database.</td>
</tr>
<tr>
<td>Desktop application</td>
<td>A tool with a graphical user interface that runs on your desktop environment, e.g. on a PC or mobile device.</td>
</tr>
<tr>
<td>Library</td>
<td>A collection of components that are used to construct other tools. bio.tools scope includes component libraries performing high-level bioinformatics functions but excludes lower-level programming libraries.</td>
</tr>
<tr>
<td>Ontology</td>
<td>A collection of information about concepts, including terms, synonyms, descriptions etc.</td>
</tr>
<tr>
<td>Plug-in</td>
<td>A software component encapsulating a set of related functions, which are not standalone, i.e. depend upon other software for its use, e.g. a Javascript widget, or a plug-in, extension add-on etc. that extends the function of some existing tool.</td>
</tr>
<tr>
<td>Script</td>
<td>A tool written for some run-time environment (e.g. other applications or an OS shell) that automates the execution of tasks. Often a small program written in a general-purpose languages (e.g. Perl, Python) or some domain-specific languages (e.g. sed).</td>
</tr>
<tr>
<td>SPARQL endpoint</td>
<td>A service that provides queries over an RDF knowledge base via the SPARQL query language and protocol, and returns results via HTTP.</td>
</tr>
<tr>
<td>Suite</td>
<td>A collection of tools which are bundled together into a convenient toolkit. Such tools typically share related functionality, a common user interface and can exchange data conveniently. This includes collections of stand-alone command-line tools, or Web applications within a common portal.</td>
</tr>
<tr>
<td>Web application</td>
<td>A tool with a graphical user interface that runs in your Web browser.</td>
</tr>
<tr>
<td>Web API</td>
<td>An application programming interface (API) consisting of endpoints to a request-response message system accessible via HTTP. Includes everything from simple data-access URLs to RESTful APIs.</td>
</tr>
<tr>
<td>Web service</td>
<td>An API described in a machine readable form (typically WSDL) providing programmatic access via SOAP over HTTP.</td>
</tr>
<tr>
<td>Workbench</td>
<td>An application or suite with a graphical user interface, providing an integrated environment for data analysis which includes or may be extended with any number of functions or tools. Includes workflow systems, platforms, frameworks etc.</td>
</tr>
<tr>
<td>Workflow</td>
<td>A set of tools which have been composed together into a pipeline of some sort. Such tools are (typically) standalone, but are composed for convenience, for instance for batch execution via some workflow engine or script.</td>
</tr>
</tbody>
</table>

Tip:

- in cases where a given software is described by more than one entry (e.g. a web application and its API are described separately) then assign only the types that are applicable to that entry.

Note:

- bio.tools includes all types of bioinformatics tools: application software with well-defined data processing functions (inputs, outputs and operations). When registering a tool, one or more tool types may be assigned, reflecting the different facets of the software being described.
- see the syntax guidelines.
**Topic**

*General scientific domain the software serves or other general category (EDAM Topic), e.g. ‘Protein sites, features and motifs’ (http://edamontology.org/topic_3510)*

- **1. MUST** specify the single most important and relevant scientific topic
- **2. MAY** specify all highly relevant scientific topics
- **3. SHOULD NOT** exhaustively specify all the topics of lower or secondary relevance

**Attention:**

- see the general guidelines for EDAM annotations.

**Note:**

- see the syntax guidelines.

---

**Operating system**

*The operating system supported by a downloadable software package, e.g. “Linux”*

- **1. MUST** specify all operating systems that are applicable, in terms from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>Maturity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>All flavours of Linux/UNIX operating systems.</td>
</tr>
<tr>
<td>Windows</td>
<td>All flavours of Microsoft Windows operating system.</td>
</tr>
<tr>
<td>Mac</td>
<td>All flavours of Apple Macintosh operating systems (primarily Mac OS X).</td>
</tr>
</tbody>
</table>

**Note:**

- see the syntax guidelines.

---

**Programming language**

*Name of programming language the software source code was written in, e.g. “C”*

- **1. MUST** specify the primary language used, in terms from a controlled vocabulary (see below)
- **2. MAY** exhaustively specify other languages used

<table>
<thead>
<tr>
<th>Programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionScript</td>
</tr>
<tr>
<td>Ada</td>
</tr>
<tr>
<td>AppleScript</td>
</tr>
<tr>
<td>Assembly language</td>
</tr>
<tr>
<td>AWK</td>
</tr>
<tr>
<td>Bash</td>
</tr>
</tbody>
</table>

Continued on next page

---

4.2. Attribute guidelines
<table>
<thead>
<tr>
<th>Programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
</tr>
<tr>
<td>C#</td>
</tr>
<tr>
<td>C++</td>
</tr>
<tr>
<td>COBOL</td>
</tr>
<tr>
<td>ColdFusion</td>
</tr>
<tr>
<td>CWL</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>Delphi</td>
</tr>
<tr>
<td>Dylan</td>
</tr>
<tr>
<td>Eiffel</td>
</tr>
<tr>
<td>Forth</td>
</tr>
<tr>
<td>Fortran</td>
</tr>
<tr>
<td>Groovy</td>
</tr>
<tr>
<td>Haskell</td>
</tr>
<tr>
<td>Icarus</td>
</tr>
<tr>
<td>Java</td>
</tr>
<tr>
<td>JavaScript</td>
</tr>
<tr>
<td>JSP</td>
</tr>
<tr>
<td>LabVIEW</td>
</tr>
<tr>
<td>Lisp</td>
</tr>
<tr>
<td>Lua</td>
</tr>
<tr>
<td>Maple</td>
</tr>
<tr>
<td>Mathematica</td>
</tr>
<tr>
<td>MATLAB</td>
</tr>
<tr>
<td>MLXTRAN</td>
</tr>
<tr>
<td>NMTRAN</td>
</tr>
<tr>
<td>OCaml</td>
</tr>
<tr>
<td>Pascal</td>
</tr>
<tr>
<td>Perl</td>
</tr>
<tr>
<td>PHP</td>
</tr>
<tr>
<td>Prolog</td>
</tr>
<tr>
<td>PyMOL</td>
</tr>
<tr>
<td>Python</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>Racket</td>
</tr>
<tr>
<td>REXX</td>
</tr>
<tr>
<td>Ruby</td>
</tr>
<tr>
<td>SAS</td>
</tr>
<tr>
<td>Scala</td>
</tr>
<tr>
<td>Scheme</td>
</tr>
<tr>
<td>Shell</td>
</tr>
<tr>
<td>Smalltalk</td>
</tr>
<tr>
<td>SQL</td>
</tr>
<tr>
<td>Turing</td>
</tr>
<tr>
<td>Verilog</td>
</tr>
<tr>
<td>VHDL</td>
</tr>
<tr>
<td>Visual Basic</td>
</tr>
<tr>
<td>XAML</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>
Note:

- see the syntax guidelines.

License

Software or data usage license, e.g. “GPL-3.0”

1. **MUST** accurately describe the license used.

2. **SHOULD** use “Proprietary” in cases where the software is under license (not defined in biotoolsSchema) whereby it can be obtained from the provider (*e.g.* for money), and then owned, *i.e.* definitely not an open-source or free software license.

3. **SHOULD** use “Freeware” for software that is available for use at no monetary cost. In other words, freeware may be used without payment but may usually not be modified, re-distributed or reverse-engineered without the author’s permission.

4. **SHOULD** use “Not licensed” for software which is not licensed and is not “Proprietary”.

5. **SHOULD** use “Other” if the software is available under a license not listed by biotoolsSchema and which is not “Proprietary”.
   - a controlled vocabulary of valid terms is defined in biotoolsSchema.
   - see the syntax guidelines.

Tip:

- Use the “Other” license for custom institutional licenses that are out of scope of biotoolsSchema. If you’ve found a license that you think should be included in biotoolsSchema please report it via GitHub.

Note: Most permissible values are identifiers from the SPDX license list ([https://spdx.org/licenses/](https://spdx.org/licenses/)). In future, based on the specified license a label (*e.g.* “Open-source”) may be attached to the bio.tools entry (see table below)
Table 2: Labelling based on license (future work)

<table>
<thead>
<tr>
<th>License</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Software is made available under a license approved by the Open Source Initiative (OSI). The software is distributed in a way that satisfies the 10 criteria of the Open Source Definition maintained by OSI (see <a href="https://opensource.org/docs/osd">https://opensource.org/docs/osd</a>). The source code is available to others.</td>
</tr>
<tr>
<td>Free software</td>
<td>Free as in ‘freedom’ not necessarily free of charge. Software is made available under a license approved by the Free Software Foundation (FSF). The software satisfies the criteria of the Free Software Definition maintained by FSF (see <a href="http://www.gnu.org/philosophy/free-sw.html">http://www.gnu.org/philosophy/free-sw.html</a>). The source code is available to others.</td>
</tr>
<tr>
<td>Free and open source</td>
<td>Software is made available under a license approved by both the Open Source Initiative (OSI) and the Free Software Foundation (FSF), and satisfies the criteria of the OSI Open Source Definition maintained (<a href="https://opensource.org/docs/osd">https://opensource.org/docs/osd</a>) and the FSF Free Software Definition (<a href="http://www.gnu.org/philosophy/free-sw.html">http://www.gnu.org/philosophy/free-sw.html</a>). Such software ensures users have the freedom to run, copy, distribute, study, change and improve the software. The source code is available to others.</td>
</tr>
<tr>
<td>Copyleft</td>
<td>Software is made available under a license designated as ‘copyleft’ by the Free Software Foundation (FSF). The license ensures such software is free and that all modified and extended versions of the program are free as well. Free as in ‘freedom’ not necessarily free of charge, as per the Free Software Definition maintained by FSF (see <a href="http://www.gnu.org/philosophy/free-sw.html">http://www.gnu.org/philosophy/free-sw.html</a>).</td>
</tr>
</tbody>
</table>

Collection

*Unique ID of a collection that the software has been assigned to within bio.tools, e.g. “CBS*

- **1. SHOULD** be short and intuitive

Tip:

- collections may be created for any arbitrary purpose

Note:

- see the syntax guidelines.

Maturity

*How mature the software product is, e.g. “Mature”*

- **1. MUST** accurately reflect the software maturity, in terms from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>Maturity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emerging</td>
<td>Nascent or early release software that may not yet be fully featured or stable.</td>
</tr>
<tr>
<td>Mature</td>
<td>Software that is generally considered to fulfill several of the following: secure, reliable, actively maintained, fully featured, proven in production environments, has an active community, and is described or cited in the scientific literature.</td>
</tr>
<tr>
<td>Legacy</td>
<td>Software which is no longer in common use, deprecated by the provider, superseded by other software, replaced by a newer version, is obsolete etc.</td>
</tr>
</tbody>
</table>
Attention:
  • normally only the developer or provider of a tool is sure of its maturity. If you are not sure, then do not complete this field.

Note:
  • see the syntax guidelines.

Cost

Monetary cost of acquiring the software, e.g. “Free of charge (with restrictions)”
  • 1. MUST accurately describe the monetary cost of acquiring the software, in terms from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free of charge</td>
<td>Software which available for use by all, with full functionality, at no monetary cost to the user.</td>
</tr>
<tr>
<td>Free of charge (with restrictions)</td>
<td>Software which is available for use at no monetary cost to the user, but possibly with limited functionality, usage restrictions, or other limitations.</td>
</tr>
<tr>
<td>Commercial</td>
<td>Software which you have to pay to access.</td>
</tr>
</tbody>
</table>

Note:
  • see the syntax guidelines.

Accessibility

Whether the software is freely available for use, e.g. “Open access”
  • 1. MUST accurately describe the accessibility conditions that apply, in terms from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>Accessibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open access</td>
<td>An online service which is available for use to all, but possibly requiring user accounts / authentication.</td>
</tr>
<tr>
<td>Restricted access</td>
<td>An online service which is available for use to a restricted audience, e.g. members of a specific institute.</td>
</tr>
</tbody>
</table>

Note:
  • see the syntax guidelines.
ELIXIR Platform

Name of the ELIXIR Platform that is credited, e.g. “Tools”

- **1. MUST** only credit the ELIXIR Platform if directly contributing to the work, using a term from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>ELIXIR Platform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>ELIXIR Data Platform</td>
</tr>
<tr>
<td>Tools</td>
<td>ELIXIR Tools Platform</td>
</tr>
<tr>
<td>Compute</td>
<td>ELIXIR Compute Platform</td>
</tr>
<tr>
<td>Interoperability</td>
<td>ELIXIR Interoperability Platform</td>
</tr>
<tr>
<td>Training</td>
<td>ELIXIR Training Platform</td>
</tr>
</tbody>
</table>

ELIXIR Node

Name of the ELIXIR Node that is credited, e.g. “Norway”

- **1. MUST** only credit the ELIXIR Node if directly contributing to the work, using a term from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>ELIXIR Node</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Belgium</td>
<td></td>
</tr>
<tr>
<td>Czech Republic</td>
<td></td>
</tr>
<tr>
<td>Denmark</td>
<td></td>
</tr>
<tr>
<td>EMBL</td>
<td></td>
</tr>
<tr>
<td>Estonia</td>
<td></td>
</tr>
<tr>
<td>Finland</td>
<td></td>
</tr>
<tr>
<td>France</td>
<td></td>
</tr>
<tr>
<td>Germany</td>
<td></td>
</tr>
<tr>
<td>Greece</td>
<td></td>
</tr>
<tr>
<td>Hungary</td>
<td></td>
</tr>
<tr>
<td>Ireland</td>
<td></td>
</tr>
<tr>
<td>Israel</td>
<td></td>
</tr>
<tr>
<td>Italy</td>
<td></td>
</tr>
<tr>
<td>Luxembourg</td>
<td></td>
</tr>
<tr>
<td>Netherlands</td>
<td></td>
</tr>
<tr>
<td>Norway</td>
<td></td>
</tr>
<tr>
<td>Portugal</td>
<td></td>
</tr>
<tr>
<td>Slovenia</td>
<td></td>
</tr>
<tr>
<td>Spain</td>
<td></td>
</tr>
<tr>
<td>Sweden</td>
<td></td>
</tr>
<tr>
<td>Switzerland</td>
<td></td>
</tr>
<tr>
<td>UK</td>
<td></td>
</tr>
</tbody>
</table>

**4.2.4 Link group**

Miscellaneous links for the software e.g. repository, issue tracker or mailing list.

**Note:**
• the bio.tools registration interface & API allows a curator to record when a link of a certain type is known to not be available
• see the syntax guidelines.

URL (link)

A link of some relevance to the software (URL), e.g. “https://github.com/pharmbio/sciluigi/issues”

• 1. MUST resolve to a page of the indicated link type
• 2. MUST NOT give a general link (e.g. homepage URL) if a more specific link is available

Link type

The type of data, information or system that is obtained when the link is resolved, e.g. “Mailing list”

• 1. MUST accurately specify the type of information available at the link, in terms from a controlled vocabulary (see below)
• 2. MUST use type “Other” if another, more specific type is not available
• 3. SHOULD specify all the types that are applicable

<table>
<thead>
<tr>
<th>Link type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion forum</td>
<td>Online forum for user discussions about the software.</td>
</tr>
<tr>
<td>Galaxy service</td>
<td>An online service providing the tool through the Galaxy platform.</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>A phone line, web site or email-based system providing help to the end-user of the software.</td>
</tr>
<tr>
<td>Issue tracker</td>
<td>Tracker for software issues, bug reports, feature requests etc.</td>
</tr>
<tr>
<td>Mailing list</td>
<td>Mailing list for the software announcements, discussions, support etc.</td>
</tr>
<tr>
<td>Mirror</td>
<td>Mirror of an (identical) online service.</td>
</tr>
<tr>
<td>Software catalogue</td>
<td>Some registry, catalogue etc. other than bio.tools where the tool is also described.</td>
</tr>
<tr>
<td>Repository</td>
<td>A place where source code, data and other files can be retrieved from, typically via platforms like GitHub which provide version control and other features, or something simpler, e.g. an FTP site.</td>
</tr>
<tr>
<td>Social media</td>
<td>A website used by the software community including social networking sites, discussion and support fora, WIKIs etc.</td>
</tr>
<tr>
<td>Service</td>
<td>An online service (other than Galaxy) that provides access (an interface) to the software.</td>
</tr>
<tr>
<td>Technical monitoring</td>
<td>Information about the technical status of a tool.</td>
</tr>
<tr>
<td>Other</td>
<td>Other type of link for software - the default if a more specific type is not available.</td>
</tr>
</tbody>
</table>

Note (link)

Comment about the link, e.g. “Please use the issue tracker for reporting bugs and making feature requests.”

• 1. SHOULD be a concise summary of practical information

4.2.5 Download group

Links to downloads for the software, e.g. source code, virtual machine image or container.
Note:

- the `bio.tools` registration interface & API allows a curator to record when a documentation link of a certain type is known to not be available
- see the syntax guidelines.

**URL (download)**

Link to download (or repo providing a download) for the software, e.g. “http://bioconductor/packages/release/bioc/src/contrib/VanillaICE_1.36.0.tar.gz”

- 1. **MUST** resolve to a page providing either an immediately download, or links for a download of the indicated link type
- 2. **MUST NOT** give a general link (e.g. homepage URL) if a more specific link is available

**Download type**

Type of download that is linked to, e.g. “Binaries”

- 1. **MUST** accurately specify the type of download available at the link, in terms from a controlled vocabulary (see below)
- 2. **MUST** use type “Other” if another, more specific type is not available
- 3. **SHOULD** use type “Downloads page” for links to general downloads pages (i.e. one which includes details about multiple types of download)
<table>
<thead>
<tr>
<th>Download type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API specification</td>
<td>File providing an API specification for the software, e.g. Swagger/OpenAPI, WSDL or RAML file.</td>
</tr>
<tr>
<td>Biological data</td>
<td>Biological data, or a web page on a database portal where such data may be downloaded.</td>
</tr>
<tr>
<td>Binaries</td>
<td>Binaries for the software; compiled code that allow a program to be installed without having to compile the source code.</td>
</tr>
<tr>
<td>Command-line specification</td>
<td>File providing a command line specification for the software.</td>
</tr>
<tr>
<td>Container file</td>
<td>Container file including the software.</td>
</tr>
<tr>
<td>Icon</td>
<td>Icon of the software.</td>
</tr>
<tr>
<td>Screenshot</td>
<td>Screenshot of the software.</td>
</tr>
<tr>
<td>Source code</td>
<td>The source code for the software, that can be compiled or assembled into an executable computer program.</td>
</tr>
<tr>
<td>Software package</td>
<td>A software package; a bundle of files and information about those files, typically including source code and / or binaries.</td>
</tr>
<tr>
<td>Test data</td>
<td>Data for testing the software is working correctly.</td>
</tr>
<tr>
<td>Test script</td>
<td>Script used for testing whether the software is working correctly.</td>
</tr>
<tr>
<td>Tool wrapper (CWL)</td>
<td>Tool wrapper in Common Workflow Language (CWL) format for the software.</td>
</tr>
<tr>
<td>Tool wrapper (galaxy)</td>
<td>Galaxy tool configuration file (wrapper) for the software.</td>
</tr>
<tr>
<td>Tool wrapper (taverna)</td>
<td>Taverna configuration file for the software.</td>
</tr>
<tr>
<td>Tool wrapper (other)</td>
<td>Workbench configuration file (other than taverna, galaxy or CWL wrapper) for the software.</td>
</tr>
<tr>
<td>VM image</td>
<td>Virtual machine (VM) image for the software.</td>
</tr>
<tr>
<td>Downloads page</td>
<td>Web page summarising general downloads available for the software.</td>
</tr>
<tr>
<td>Other</td>
<td>Other type of download for software - the default if a more specific type is not available.</td>
</tr>
</tbody>
</table>

**Note (download)**

*Comment about the download, e.g. “Complete distribution”*

- **1. SHOULD** be concise and summarise only practical information about the link

**Version (download)**

*Version information (typically a version number) of the software applicable to this download.*

- **1. MUST** correctly identify the applicable tool version
- **2. MUST** follow the general guidelines for version

### 4.2.6 Documentation group

*Links to documentation about the software e.g. manual, API specification or training material.*

**Note:**

- the **bio.tools** registration interace & API allows a curator to record when a documentation link of a certain type is known to *not* be available
see the syntax guidelines.

**URL (documentation)**

`Link to documentation on the web for the tool, e.g. “http://bioconductor.org/packages/release/bioc/html/VanillaICE.html”`

- **1. MUST** resolve to a page of the indicated documentation type
- **2. MUST NOT** give a general link (e.g. homepage URL) if a more specific link is available

**Documentation type**

`Type of documentation that is linked to, e.g. “Citation instructions”`

- **1. MUST** accurately specify the type of documentation available at the link, in terms from a controlled vocabulary (see below)
- **2. MUST** use type “Other” if another, more specific type is not available
- **3. SHOULD** specify all the types that are applicable

<table>
<thead>
<tr>
<th>Documentation type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API documentation</td>
<td>Human-readable API documentation.</td>
</tr>
<tr>
<td>Citation instructions</td>
<td>Information on how to correctly cite use of the software; typically which publication(s) to cite, or something more general, e.g. a form of words to use.</td>
</tr>
<tr>
<td>Code of conduct</td>
<td>A set of guidelines or rules outlining the norms, expectations, responsibilities and proper practice for individuals working within the software project.</td>
</tr>
<tr>
<td>Command-line options</td>
<td>Information about the command-line interface to a tool.</td>
</tr>
<tr>
<td>Contributions policy</td>
<td>Information about policy for making contributions to the software project.</td>
</tr>
<tr>
<td>FAQ</td>
<td>Frequently Asked Questions (and answers) about the software.</td>
</tr>
<tr>
<td>General</td>
<td>General documentation.</td>
</tr>
<tr>
<td>Governance</td>
<td>Information about the software governance model.</td>
</tr>
<tr>
<td>Installation instructions</td>
<td>Instructions how to install the software.</td>
</tr>
<tr>
<td>User manual</td>
<td>Information on how to use the software, tailored to the end-user.</td>
</tr>
<tr>
<td>Release notes</td>
<td>Notes about a software release or changes to the software; a change log.</td>
</tr>
<tr>
<td>Terms of use</td>
<td>Rules that one must agree to abide by in order to use a service.</td>
</tr>
<tr>
<td>Training material</td>
<td>Online training material such as a tutorial, a presentation, video etc.</td>
</tr>
<tr>
<td>Other</td>
<td>Some other type of documentation not listed in biotoolsSchema.</td>
</tr>
</tbody>
</table>

**Note (documentation)**

`Comment about the documentation, e.g. “Comprehensive usage information suitable for biologist end-users.”`

- **1. SHOULD** be concise and summarise only practical information about the link

### 4.2.7 Relation group

`Details of a relationship this software shares with other software registered in bio.tools.`
• 1. **MUST** correctly identify a relationship between two *bio.tools* entries
• 2. **MUST NOT** not contradict a relationship that is already specified in *bio.tools*
• 3. **MUST** specify a valid biotoolsID (of a tool that’s registerd in *bio.tools*)

**Note:**
- see the syntax guidelines.

### biotoolsID (relation)

*bio.tools ID of an existing bio.tools entry which this software is related to, e.g. “needle”*

#### Relation type

Type of relation between this and another registered software, e.g. “isNewVersionOf”

<table>
<thead>
<tr>
<th>Relation type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isNewVersionOf</td>
<td>The software is a new version of an existing software, typically providing new or improved functionality.</td>
</tr>
<tr>
<td>hasNewVersion</td>
<td>(inverse of above)</td>
</tr>
<tr>
<td>uses</td>
<td>The software provides an interface to or in some other way uses the functions of other software under the hood, e.g. invoking a command-line tool or calling a Web API, Web service or SPARQL endpoint to perform its function.</td>
</tr>
<tr>
<td>usedBy</td>
<td>(inverse of above)</td>
</tr>
<tr>
<td>includes</td>
<td>A workbench, toolkit or workflow includes some other, independently available, software.</td>
</tr>
<tr>
<td>includedIn</td>
<td>(inverse of above)</td>
</tr>
</tbody>
</table>

### 4.2.8 Publication group

Publications about the software

• 1. **MUST** correctly identify a relevant publication
• 2. **MUST** specify multiple IDs for a single publication within a single publication group
• 3. **SHOULD** specify a DOI (if available) (in preference to PMID and PMCID)
• 4. **MAY** specify one or more types that match the publication

**Note:**
- see the syntax guidelines.

### PubMed Central ID

*PubMed Central Identifier (PMCID) of a publication about the software, e.g. “PMC4343077”*
PubMed ID

*PubMed Identifier (PMID) of a publication about the software, e.g. “21959131”*

Digital Object ID

*Digital Object Identifier (DOI) of a publication about the software, e.g. “10.1038/nmeth.1701”*

Publication type

*Type of publication, e.g. “Primary”*

- **1. MUST** accurately specify the type of publication, in terms from a controlled vocabulary (see below)
- **2. SHOULD** specify all the types that are applicable

<table>
<thead>
<tr>
<th>Publication type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>The principal publication about the tool itself; the article to cite when acknowledging use of the tool.</td>
</tr>
<tr>
<td>Method</td>
<td>A publication describing a scientific method or algorithm implemented by the tool.</td>
</tr>
<tr>
<td>Usage</td>
<td>A publication describing the application of the tool to scientific research, a particular task or dataset.</td>
</tr>
<tr>
<td>Benchmarking study</td>
<td>A publication which assessed the performance of the tool.</td>
</tr>
<tr>
<td>Review</td>
<td>A publication where the tool was reviewed.</td>
</tr>
<tr>
<td>Other</td>
<td>A publication of relevance to the tool but not fitting the other categories.</td>
</tr>
</tbody>
</table>

Note (publication)

*A comment about the publication, e.g. “A comparison of the software to others performing a similar function.”*

- **1. SHOULD** be concise and accurate, elaborating on the motivation, purpose etc. of the publication
- **2. SHOULD NOT** duplicate information that is, or can, be provided via the type or other attributes, *i.e.* do not specify “Review article”, “Cite this where the software is used” etc.

Version (publication)

*Version information (typically a version number) of the software applicable to this publication.*

- **1. MUST** correctly identify the applicable tool version
- **2. MUST** follow the general guidelines for version

4.2.9 Credit group

*Individuals or organisations that should be credited, or may be contacted about the software.*

- **1. SHOULD** provide contact details for the first port-of-call when seeking help with the software, and **SHOULD** annotate the role of this entity as “Primary contact”
- **2. MAY** specify one or more other credits
Note:
• a credit consists of the name, email and/or URL of some entity that is credited, with other associated metadata
• see the syntax guidelines.

Name (credit)

Name of the entity that is credited, e.g. “EMBL EBI”

• 1. MUST give the first and last names of a person, or the correct name of some other entity.
• 2. MUST NOT give a redirect, e.g. “See publication”, a URL, or any information other than the name of the entity that is credited.

ORCID ID

Unique identifier (ORCID iD) of a person that is credited, e.g. “http://orcid.org/0000-0002-1825-0097”

• 1. MUST correctly identify a credited person

Note: Open Researcher and Contributor IDs (ORCID IDs) provide a persistent reference to information on a researcher, see http://orcid.org/.

GRID ID

Unique identifier (GRID ID) of an organisation that is credited, e.g. “grid.5170.3”

• 1. MUST correctly identify a credited organisation

Note: Global Research Identifier Database IDs (GRID IDs) provide a persistent reference to information on an organisation, see https://www.grid.ac/.

Email

Email address of the entity that is credited e.g. “hnielsen@cbs.dtu.dk”

• 1. MUST specify a syntactically valid email address
• 2. MUST NOT specify an email address that is not publicly acknowledged as credit for the software, e.g. on a webpage or in a publication
• 3. MUST NOT specify a stale (obsolete) email address

URL (credit)

URL for the entity that is credited, e.g. homepage of an institute, e.g. “http://www.ebi.ac.uk/”

• 1. MUST resolve to a page of information directly relevant to the credited entity

4.2. Attribute guidelines
**Entity type**

*Type of entity that is credited, e.g. “Person”*

- **1. MUST** accurately specify the type of entity that is credited, in terms from a controlled vocabulary (see below)

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Credit of an individual.</td>
</tr>
<tr>
<td>Project</td>
<td>Credit of a community software project not formally associated with any single institute.</td>
</tr>
<tr>
<td>Division</td>
<td>Credit of or a formal part of an institutional organisation, e.g. a department, research group, team, etc</td>
</tr>
<tr>
<td>Institute</td>
<td>Credit of an organisation such as a university, hospital, research institute, service center, unit etc.</td>
</tr>
<tr>
<td>Consortium</td>
<td>Credit of an association of two or more institutes or other legal entities which have joined forces for some common purpose. Includes Research Infrastructures (RIs) such as ELIXIR.</td>
</tr>
<tr>
<td>Funding agency</td>
<td>Credit of a legal entity providing funding for development of the software or provision of an online service.</td>
</tr>
</tbody>
</table>

**Entity role**

*Role performed by entity that is credited, e.g. “Developer”*

- **1. MUST** accurately specify the primary role of credited entity, in terms from a controlled vocabulary (see below)
- **2. MAY** exhaustively specify all the roles of the credited entity

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>Author of the original software source code.</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Maintainer of a mature software providing packaging, patching, distribution etc.</td>
</tr>
<tr>
<td>Provider</td>
<td>Institutional provider of an online service.</td>
</tr>
<tr>
<td>Documentor</td>
<td>Author of software documentation including making edits to a bio.tools entry.</td>
</tr>
<tr>
<td>Contributor</td>
<td>Some other role in software production or service delivery including design, deployment, system administration, evaluation, testing, documentation, training, user support etc.</td>
</tr>
<tr>
<td>Support</td>
<td>Provider of support in using the software.</td>
</tr>
<tr>
<td>Primary contact</td>
<td>The primary point of contact for the software.</td>
</tr>
</tbody>
</table>

**Note (credit)**

*A comment about the credit, e.g. “Wrote the user manual.”*

- **1. SHOULD** be concise and accurate, elaborating on the contribution of the credited entity
- **2. MUST NOT** duplicate information that is, or can, be provided via the *role* attribute, *i.e.* do not specify only “Developer”, “Support” *etc.*

### 4.3 Tool type guidelines

#### 4.3.1 Bioinformatics portal

A web site providing a platform/portal to multiple resources used for research in a focused area, including biological databases, web applications, training resources and so on.
• pick one or more topics that best describe the portal content.

• consider carefully whether the portal will be described by a single, or more than one bio.tools entry (see Before you start). Where the portal aggregates one or more discrete tools (web applications), databases etc., it is recommended to register these as separate entries.

4.3.2 Command-line tool

A tool with a text-based (command-line) interface.

• carefully identify the major functions (modes of operation) performed by the tool (see Tool functions) and annotate the major operation(s) associated with each function, in turn.

4.3.3 Database portal

A Web application, suite or workbench providing a portal to a biological database.

• pick one or more topics that best describe the database content. See also the specialised Data management concepts.

• consider carefully whether the database portal will be described by a single, or more than one bio.tools entry (see Before you start). In case the portal contains one or more discrete tools (web applications), it is recommended to register these as separate entries.

• consider an operation of Database search (or its children)

4.3.4 Desktop application

A tool with a graphical user interface that runs on your desktop environment, e.g. on a PC or mobile device.

• desktop applications often have complex functionality: carefully identify the major functions (modes of operation) performed by the application (see Tool functions) and annotate the major operation(s) associated with each function, in turn.

• consider an operation of Visualisation (or its children) - typical of desktop apps.

4.3.5 Library

A collection of components that are used to construct other tools. bio.tools scope includes component libraries performing high-level bioinformatics functions but excludes lower-level programming libraries.

• in case the library includes just a few components, each should (typically) be modelled as a distinct function (see Tool functions); annotate the major operation(s) associated with each component (function) in turn.

• in case the library includes very many components, model the whole library as having a single function (see Tool functions); and annotate only the major operation(s) (do not try to be exhaustive).

4.3.6 Ontology

A collection of information about concepts, including terms, synonyms, descriptions etc.

• pick Ontology and terminology and one or more most relevant topics describing the scope of the ontology.

• do not annotate the function (operations, or type / format of the input and output data)
4.3.7 Plug-in

A software component encapsulating a set of related functions, which are not standalone, *i.e.* depend upon other software for its use, e.g. a Javascript widget, or a plug-in, extension add-on etc. that extends the function of some existing tool.

- when annotating the plug-in function(s), be careful to not duplicate the description of the tool which plug-in plugs into
- carefully identify the major new functions (modes of operation) which the plug-in provides, and annotate the major operation(s) associated with each function, in turn.

4.3.8 Script

A tool written for some run-time environment (e.g. other applications or an OS shell) that automates the execution of tasks. Often a small program written in a general-purpose languages (e.g. Perl, Python) or some domain-specific languages (e.g. sed).

- scripts typically have a single function (mode of operation) (see Tool functions), however, in case of complex scripts, carefully identify the major functions (modes of operation) performed by the script, and annotate the major operation(s) associated with each function, in turn.
- pick one or more most relevant topics

4.3.9 SPARQL endpoint

A service that provides queries over an RDF knowledge base via the SPARQL query language and protocol, and returns results via HTTP.

- pick the operation of “Query and retrieval” (http://edamontology.org/operation_0224)
- do not annotate the type or format of the input and output data

4.3.10 Suite

A collection of tools which are bundled together into a convenient toolkit. Such tools typically share related functionality, a common user interface and can exchange data conveniently. This includes collections of stand-alone command-line tools, or Web applications within a common portal.

- pick one or more most relevant topics that describe the workbench as a whole (don’t try to be exhaustive)
- describe the attributes that are common to the suite as a whole, not (typically) attributes of individual tools
- individual tools included in the suite should be registered as separate entries
- when annotating the operation of the suite, select operations that are core function of the suite itself / common to all tools in the suite. Alternatively pick one or two of the primary operation(s) of the included tools
- entries for the suite itself and it’s component tools can be associated by annotatong them as part of a common collection

Tip: If you are considering to register a suite with many tools, it is a good idea to discuss this first with the bio.tools admin.
**4.3.11 Web application**

A tool with a graphical user interface that runs in your Web browser.

- pick one or more most relevant topics

**Note:**

- for software that essentially just wraps or provides an interface to some other tool, e.g. a web application or web service over an existing tool, use the pattern `toolName providerName` where `providerName` is a name (without spaces) of some institute, workbench, collection *etc.*, e.g. cufflinks cloudIFB. **Do not** misappropriate the original name!

**4.3.12 Web API**

An application programming interface (API) consisting of endpoints to a request-response message system accessible via HTTP. Includes everything from simple data-access URLs to RESTful APIs.

- pick one or more most relevant topics
- in general, describe the attributes of the API as a whole, not individual endpoint of the API (see note below)
- in case the API has a single endpoint only, the input(s), operation(s) and output(s) may be annotated
- in case the API has many endpoints, annotate the primary operation(s), but **not** the inputs and outputs
- annotate the location of machine-readable API specification (e.g. openAPI file) using the `download` attribute with `download type of API specification` - annotate the location of any human-readable documentation using the `documentation` attribute with `documentation type of API specification`
- when assigning the `name`, use the pattern `name API` e.g. Open PHACTS API
- in case the web service provides an interface to an existing tool registered in `bio.tools`, try to ensure the relevant annotations are consistent

**Note:**

- `biotoolsSchema` includes a basic model of an API specification including endpoints however this is not yet supported in `bio.tools`

**4.3.13 Web service**

An API described in a machine readable form (typically WSDL) providing programmatic access via SOAP over HTTP.

- pick one or more most relevant topics
- in general, describe the attributes of the web service as a whole, not individual endpoint of the service (see note below)
- in case the web service has a single endpoint only, the input(s), operation(s) and output(s) may be annotated
• in case the web service has many endpoints, annotate the primary operation(s), but not the inputs and outputs
• annotate the location of the WSDL file using the download attribute with download type of API specification
• annotate the location of any human-readable documentation using the documentation attribute with documentation type of API specification
• when assigning the name, use the pattern name WS e.g. EMMA WS
• in case the web service provides an interface to an existing tool registered in bio.tools, try to ensure the relevant annotations are consistent

Note:
• biotoolsSchema includes a basic model of an API specification including endpoints however this is not yet supported in bio.tools

4.3.14 Workbench

An application or suite with a graphical user interface, providing an integrated environment for data analysis which includes or may be extended with any number of functions or tools. Includes workflow systems, platforms, frameworks etc.

• pick one or more most relevant topics that best describe the workbench as a whole (don’t try to be exhaustive)
• describe the attributes of the workbench as a whole, not (typically) individual tools or functions provided by it
• individual tools included in the workbench, especially where these tools are independently available, should be registered as separate entries
• individual functions provided by the workbench, especially where these are not independently available, should each be described in their own function
• entries for the workbench itself and it’s component tools can be associated by annotating them as part of a common collection

Tip: If you are considering to register a complicated workbench with many tools or functions, it is a good idea to discuss this first with the *bio.tools* admin.

4.3.15 Workflow

A set of tools which have been composed together into a pipeline of some sort. Such tools are (typically) standalone, but are composed for convenience, for instance for batch execution via some workflow engine or script.

• pick one or more most relevant topics that best describe the workflow as a whole (don’t try to be exhaustive)
• when deciding how to annotate a workflow inputs, operations and outputs, consider the workflow as a “black box” i.e. annotate the input(s) to, output(s) from and primary operation(s) of the workflow as a whole

Note:
• bio.tools does not currently contain many examples of workflows. We welcome input on how to describe workflows and ensure good coverage: please get in touch with us.
Important: workflows can contain many tools; **do not** list all the operations performed by these tools, just the main operation(s) of the workflow as a whole.

### 4.4 Further guidelines (bio.tools admin only)

| Attention: | The guidelines that follow are for attributes and other aspects under the control of bio.tools admin. If you’re not a bio.tools admin you can ignore this section. |

#### 4.4.1 summary->biotoolsID

*Unique ID (case insensitive) of the tool that is assigned upon registration of the software in bio.tools, normally identical to tool name, e.g. “needle”.*

<table>
<thead>
<tr>
<th>Attention:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• the ID by default is a URL-safe version of the tool name, and is set (and can only be changed) by bio.tools admin.</td>
</tr>
</tbody>
</table>

• MUST use the default value where possible

• MUST be clean and intuitive (in case use of default is not possible)

• MUST NOT truncate the name (in the middle of a word, or at all) if this renders the ID ugly or meaningless

<table>
<thead>
<tr>
<th>Note: Transformation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following rules apply when transforming the supplied tool name:</td>
</tr>
</tbody>
</table>

• replace ‘ ‘ (spaces) in the name with underscores (a single underscore in case of multiple spaces)

• preserve all reserved characters (uppercase and lowercase letters, decimal digits, hyphen, period, underscore, and tilde), but remove other characters

• use ‘_’ to delimit parts of names but only if these are not already truncated in the original name

• adhere to the same patterns for tool name, e.g. EMBOSS_water_API_ebi

#### 4.4.2 summary->biotoolsCURIE

*bio.tools CURIE (compact URI) based on the unique bio.tools ID of the tool, e.g. “biotools:needle”*

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• identical to biotoolsID but with the prefix biotools:</td>
</tr>
</tbody>
</table>
4.4.3 credit->elixirNode

*ELIXIR node credited for developing or providing the software - the software is in Node Service Delivery Plan, e.g. “Denmark”*

- 1. **MUST** accurately specify an ELIXIR Node that is credited, in terms from a controlled vocabulary
- 2. **MUST** only be credited on tools that are in a Node’s Service Delivery Plan.
- 3. **MUST** only be set by a ELIXIR Node manager or ELIXIR Hub.

4.4.4 credit->elixirPlatform

*ELIXIR platform credited for developing or providing the software, e.g. “Tools”*

- 1. **MUST** accurately specify an ELIXIR Platform that is credited, in terms from a controlled vocabulary
- 2. **MUST** only be set by a ELIXIR Node manager or ELIXIR Hub.
Community-specific guidelines

Tool information profiles and tutorials tailored to specific communities

Different communities - whether national, scientific or ones focused on a specific project - have different perspectives on the information they wish to disseminate about the tools and services within their portfolios.

Tool information profiles are a formal way to define which tool attributes - supported by the bio.tools registry - should be specified within a set of tool descriptions registered in bio.tools.

Accompanying the profiles are tutorials which tool developers and service providers can follow to improve the description of their resources in bio.tools. The aim is to improve the quality of the community tools portfolio, by improving the bio.tools entries, and by highlighting areas where the service around those tools can be improved.

Note: This page currently describes the profile and tutorial developed for IFB, but the expectation is that other communities will follow.

5.1 IFB tools

Caution: This tutorial describes some features of bio.tools which have not yet been released in the public version available at https://bio.tools. These features - and what to do - are indicated in an “Caution” box (like this one).

These instructions will guide you through the steps to register your tools and databases in bio.tools and describe them to the standard required for inclusion in the next iteration of the IFB Catalogue - the French national catalogue of bioinformatics resources.

There are various sources of information and help:

- This document provides guidelines tailored to the IFB catalogue, highlighting key information and common pitfalls.
bio.tools Documentation, Release latest

• The bio.tools Curators Guide provides in-depth curation guidelines. You might need to refer to it, wherever you see the {learn more} links.
• The biotoolsSchema documentation summarises the attributes, information model and controlled vocabularies - including the EDAM ontology - used by bio.tools. Links to these docs are provided where needed.
• To get help using bio.tools, or for general curation advice, please mail registry-support.
• If you have questions specifically about the IFB catalogue curation process, you can mail Jon Ison directly.

Note: These instructions are tailored to the needs of IFB tool providers. If you find a bug, or have any questions or suggestions, please post them on GitHub.

5.1.1 1. Get a bio.tools account

You’ll need an account to create bio.tools entries or edit existing ones. Creating an account is simple: just go to bio.tools and click on Sign-up at the top-right corner of the page.

You’ll be asked for a username, email address and password. Your account will be setup immediately.

Note: bio.tools entries are owned by the individuals who created them. Owners may grant edit rights, or transfer ownership of their entries to other registered users. The rightful owner of a bio.tools entry is usually the person who developed the tool, or provides an online service, but it can be some other responsible person, e.g. a dedicated curator.

5.1.2 2. Claim your bio.tools entries

As a software developer or service provider, you should own the bio.tools entries describing your tools, by claiming ownership of existing entries or creating new ones.

You’ll need to login first, by clicking on Login at the top-right corner of the page.

To see whether a tool is already registered, search for it by its name. Simply type the name in the search box. You may need to click on the Name facet to narrow-down the search:
Once you find your entry you can go ahead and update it. If you can’t find the entry, you’ll need to create it.

**Important:** All tools that were submitted for consideration in the ELIXIR FR Service Delivery Plan should already be registered, but may have only very basic details. You will need to take ownership and improve the entries. Before starting work, please ensure you understand the information requirement and follow the guidelines below.

### Updating entries

To edit an existing entry, you need to click through to the Tool Card for the tool in question, e.g. [https://bio.tools/signalp](https://bio.tools/signalp). You’ll see a one or two buttons at the bottom right of the Tool Card, depending on whether you’re logged in, and own the entry or not.

- Click on **Request ownership** if you want to claim ownership of the entry
- Click on **Request editing rights** if you want to edit rights on the entry, but not own it
- Click on **Update this record** to edit the entry (visible only if you own the entry or have editing rights)

**Note:** It can take a little while for other users to respond to requests for edit rights or ownership. If these are not granted within a day or two, please mail registry-support.
Creating entries

To create new entries you’ll need to be logged onto bio.tools. Click on Menu → Add content:

![Menu and jjson drop-down menu](image)

### bio.tools editing interface

The bio.tools editing interface helps you to create valid tool descriptions. It’s organised into different tabs (Summary, Function, Labels etc.):

![Editing interface with tabs](image)

The editing interface provides some hints, and ensures that the information you set is in the right format. At any moment, you can click on ![Save button](image) to save your edits, and immediately publish the changes online. The information you specified is checked to ensure it’s in the right syntax. To (optionally) force a manual syntax check, click on ![Validate button](image).

**Important:** The attributes required by bio.tools (tool name, description and homepage URL) are marked with a red asterix * in the editing interface, and must be given before an entry can be saved. Much more information is required.
for the IFB catalogue, but this is not enforced by bio.tools!

**Note:** It’s possible to create tool descriptions in JSON format directly in a text editor, and either paste these into the editing interface (“JSON” tab) or use the bio.tools API. For guidance on using the API, see the API Reference and the API Usage Guide.

### Removing entries

To remove an entry, click on *Update this record* button (bottom right of the Tool Card). Then you can remove the entry by clicking on ![Remove](remove.png).

**Warning:** It shouldn’t normally be necessary to remove a bio.tools entry, and you should try to avoid needing to do so! Although deleted entries are actually just hidden, not really deleted, removing an entry is definitive. There’s no way back (other than emailing Registry Support).

### 5.1.3 3. Understand the information requirement

**bio.tools**

bio.tools requires only the name, description and homepage URL for a tool registration, but supports a comprehensive set of attributes for rich tool descriptions.

**Note:** The attributes supported by bio.tools, their structure and their syntax are defined in formalised XML schema called biotoolsSchema. You don’t need to look at the schema, because everything is handled through bio.tools. If you’d like to learn more or contribute to this project, please head over to GitHub.

**The IFB catalogue**

The information requirement of the IFB catalogue is more demanding than bio.tools, and depends upon the type of tool (command-line tool, database *etc.*) that is being registered. A given tool attribute is *Mandatory*, *Recommended* or *Optional* for a given type of tool:

- **Mandatory** attributes MUST be specified.
- **Recommended** attributes SHOULD be specified, but are not strictly required.
- **Optional** attributes CAN be specified, to produce a rich tool description.
<table>
<thead>
<tr>
<th></th>
<th>Bioinformatics portal</th>
<th>Database portal</th>
<th>Web application</th>
<th>Desktop application</th>
<th>Command-line tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Description</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Homepage</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Unique ID</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Tool type</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Scientific topics</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Scientific operations</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Type of input &amp; output data</td>
<td>-</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Supported data formats</td>
<td>-</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Primary publication</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Citation instructions</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Credited developer</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Credited maintainer</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Credited institute</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Credited funding agency</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Maturity</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Accessibility

Cost
The above diagram is intended to give a quick overview of the information requirement. Only the main types of tool and most important attributes are shown. The guidelines below cover exactly what’s needed for each type of tool, and go through the curation process in a step-by-step way.

**Important:** All tools in the IFB catalogue must have at least a minimal description, *i.e.* all mandatory attributes are specified. Tool providers are encouraged to provide an enhanced description which also includes all of the recommended attributes.

### 5.1.4 4. Plan your curation work

**bio.tools entries**

**Important:** Before you use bio.tools to create and edit tool descriptions, it’s important to plan carefully the entries with respect to the types of tool and the functions they perform. Be sure to understand:

1. The type of tool being described - this determines the information requirement - and is covered in the section below on *tool type*.
2. The tool functionality and how it should be described - covered in the section on *tool functions*.
3. Whether one or more entries are needed (see below).

Plan what new entries (if any) are required to describe your tools:

- A discrete tool - one which is clearly an individual, distinct entity - should have it’s own entry. This is the case for most *command-line tools* and *desktop applications*.

- bio.tools aims to catalogue unique tool functionality. Different implementations but with essentially the same functionality can be described by a single entry, *e.g.* a command-line tool that is later adapted into an R package for the Bioconductor suite, or which is served online via a Galaxy server.

- In some cases, *e.g.* complex software packages, it’s not obvious whether to have one or multiple entries. Pick the option which most clearly illustrates the tool’s functionality to end-users.

- Tool collections should be described by multiple entries. For example, one entry to describe a *suite*, and multiple other entries to describe the individual tools within that suite.

- Software with multiple interfaces should be described by a single entry, assuming these interfaces have essentially the same functionality. For example, a *command-line tool* whose functionality is also available via a *web application*, or a *database portal* with a *web API*.

- Many *database portals* provide the typical database functions (browse, deposit, search, visualise, analyse and download), often in different interface components. Usually one entry will suffice, but sometimes multiple entries are better, especially where the portal provides multiple analytical functions under different interfaces.

- For very complex entities such as *Bioinformatics portals*, do not try to describe everything in a single entry. Use a single entry for the portal, and multiple other entries for the things aggregated by the portal.

**Familiarise yourself with EDAM**

The EDAM ontology provides bio.tools with a controlled vocabulary to describe the scientific function of a tool, including the general scientific domain, specific operations it performs, types of input and output data, and supported data formats.
Chapter 5. Community-specific guidelines
The EDAM ontology includes four main types of concept (or subontologies), shown in boxes above. The concepts are Topic, Operation, Data and Format, with Identifier being a specialisation of Data. Relationships between EDAM concepts are defined internally within the ontology. You don’t need to worry about these details, as it’s all handled by bio.tools.

### Picking EDAM terms

Three EDAM browsers, each with different functionality, can be used to find EDAM terms:

- OLS
- BioPortal
- EDAM Browser

**Tip:** The EDAM term picker currently implemented in bio.tools is not very powerful. It’s strongly recommended to use the browsers above. If you can’t find exactly the terms you need, multiple searches using synonyms, alternative spellings etc. can help.

A much better term picker is on the way, and while not yet fully integrated into bio.tools is already very useful:

- EDAM Tool Annotator
You can use this to pick relevant topics and define the function of your tools. The output (in the bottom pane of the window) is a JSON object that can be copy-pasted into the JSON tab of the bio.tools editing interface, when editing a tool description.

If you can’t find the right term, please request that it’s added to EDAM via GitHub but first read the guidelines on how to request a term. It takes some time for new terms to be supported in bio.tools, so if you need many new terms, please plan ahead and contact the EDAM developers if you need help.

5.1.5 5. Describe your tools

The sections below match the tabs in the bio.tools editing interface.

Note: Only those tool attributes that are Mandatory or Recommended are described below, but you can of course also specify the Optional ones.

The [learn more] links take you to more detailed guidelines in the bio.tools Curators Guide. Follow these links whenever you’re not sure about what information is needed.

**Summary**

In the Summary tab you specify basic information about the software:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Description</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Homepage URL</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

- **Name** is the short-form name by which the tool is commonly known, *e.g.* “BLAST” not “Basic Local Alignment Search Tool”. Database names should follow a pattern where the name and abbreviation are given *e.g.* “The Protein Databank (PDB)” [learn more].

- **Description** is a concise textual summary of the tool function or purpose. It can usually be copy-pasted from the tool homepage. Do not include statements about performance, provenance, governance *etc.* [learn more].

- **Homepage URL** is the tool’s homepage, or some URL that best serves this purpose [learn more].

**Important**: A unique identifier - the bio.tools toolID - is created for a tool when a new entry is created. The ID value is a URL-safe version of the supplied tool name. The ID provides a persistent reference to the tool, used by bio.tools and other systems. toolIDs are used in the Tool Card URLs, which can be represented in a short form as a “compact URI” or “CURIE”:

- **toolID**: signalp
- **CURIE**: biotools:signalp
- **Tool Card URL**: https://biotools/signalp

The ID should be sensible and intuitive. For databases, or tools with long names, the abbreviation should be used. For example, the GnpIS tool has the ID “gnpis” and not “Genetic and Genomic Information System”.

**Tip**: The toolID is not currently editable, so if you want the ID to differ from the name (*e.g.* an ID of “PDB” for the tool name “Protein databank (PDB)”), you have to apply a workaround:
1) create the entry giving a value for “Name” which is the desired ID value, *e.g.* “PDB”

2) Save the entry

3) Edit the entry, resetting the name, *e.g.* to “Protein Databank (PDB)”

To request an ID change post-registration (to be avoided!) you have to mail Registry Support.

---

### Labels

In the *Labels* tab you specify miscellaneous scientific, technical and administrative details, expressed in terms from controlled vocabularies:

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool type</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Topic</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Cost</td>
<td>Mandatory</td>
</tr>
<tr>
<td>License</td>
<td>Mandatory <em>(Desktop application, Command-line tool)</em></td>
</tr>
<tr>
<td>Operating system</td>
<td>Mandatory <em>(Desktop application)</em></td>
</tr>
<tr>
<td>Maturity</td>
<td>Recommended <em>(Command-line tool)</em></td>
</tr>
<tr>
<td>Accessibility</td>
<td>Recommended <em>(Bioinformatics portal, Database portal, Web application)</em></td>
</tr>
<tr>
<td>Language</td>
<td>Recommended <em>(Command-line tool)</em></td>
</tr>
</tbody>
</table>

- **Tool type** describes the type of the tool: a *bio.tools* entry can have more than one type. See *below* [{learn more}].
- **Topic** is the general scientific domain the tool serves, or other general category (an EDAM term). See *below* [{learn more}].
- **Cost** is the monetary cost of acquiring the software [{learn more}].
- **License** is a software or data usage license. See *below* [{learn more}].
- **Operating system** is the operating system supported by a downloadable software package - pick all that apply [{learn more}].
- **Maturity** is how mature the software product is; *Emerging*, *Mature* or *Legacy*. Don’t pick *Mature* for tools which aren’t really mature yet! [{learn more}].
- **Accessibility** is whether an online service is freely available for use; either *Open access* or *Restricted access*. Read the definitions before picking these terms! [{learn more}].
- **Language** is the name of a programming language the tool source code was written in [{learn more}].

**Tip:** You can use Collection to assign tools which are somehow related to one or more groups. These collections can have any names you like. Other ways to group tools are by creating a *bio.tools* subdomain (from *Menu...Manage subdomains*) and by defining relations between tools.

---

**Note:** ELIXIRNode and ELIXIRPlatform define the name of an ELIXIR node or ELIXIR platform, respectively, that is credited for the tool. All tools in the IFB catalogue will have the ELIXIRNode credit set to “France”. These are not normally be set by *bio.tools* users [{learn more}].
Tool type

The scope of bio.tools is very broad - ranging from simple scripts to comprehensive bioinformatics portals - as defined by 15 different tool types. The vast majority of entries are of the following types:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bioinformatics portal</td>
<td>A web site providing a platform/portal to multiple resources used for research in a focused area, including biological databases, web applications, training resources and so on.</td>
</tr>
<tr>
<td>Database portal</td>
<td>A Web site providing a portal to a biological database, typically allowing a user to browse, deposit, search, visualise, analyse or download data.</td>
</tr>
<tr>
<td>Web application</td>
<td>A tool with a graphical user interface that runs in your Web browser.</td>
</tr>
<tr>
<td>Desktop application</td>
<td>A tool with a graphical user interface that runs on your desktop environment, e.g. on a PC or mobile device.</td>
</tr>
<tr>
<td>Command-line tool</td>
<td>A tool with a text-based (command-line) interface.</td>
</tr>
</tbody>
</table>

Other common types incude:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web API</td>
<td>An application programming interface (API) consisting of endpoints to a request-response message system accessible via HTTP. Includes everything from simple data-access URLs to RESTful APIs.</td>
</tr>
<tr>
<td>Workflow</td>
<td>A set of tools which have been composed together into a pipeline of some sort. Such tools are (typically) standalone, but are composed for convenience, for instance for batch execution via some workflow engine or script.</td>
</tr>
<tr>
<td>Suite</td>
<td>A collection of tools which are bundled together into a convenient toolkit. Such tools typically share related functionality, a common user interface and can exchange data conveniently. This includes collections of stand-alone command-line tools, or Web applications within a common portal.</td>
</tr>
<tr>
<td>Workbench</td>
<td>An application or suite with a graphical user interface, providing an integrated environment for data analysis which includes or may be extended with any number of functions or tools. Includes workflow systems, platforms, frameworks etc.</td>
</tr>
<tr>
<td>Workflow</td>
<td>A set of tools which have been composed together into a pipeline of some sort. Such tools are (typically) standalone, but are composed for convenience, for instance for batch execution via some workflow engine or script.</td>
</tr>
<tr>
<td>Library</td>
<td>A collection of components that are used to construct other tools. bio.tools scope includes component libraries performing high-level bioinformatics functions but excludes lower-level programming libraries.</td>
</tr>
</tbody>
</table>

A single bio.tools entry is annotated with one or more types, reflecting different facets of the tool described by the entry. Be sure to understand the type(s) of tool you have, because it determines the information that’s expected. A few suggestions:

- **Bioinformatics portals** aggregate information about tools and databases, but don’t (typically) directly serve them. Only use Bioinformatics portal for sites that cover multiple other resources, each which are their own distinct entity (and should have their own bio.tools entries). Good examples include IMGT and wheatIS.

- **Suite** might be more applicable than Bioinformatics portal. Example include Web application suites such as CRISRP-Cas++ and EvryRNA, and suites of command-line tools such as BioConductor.

- **Workbench** might be also be more applicable than Bioinformatics portal. This includes online and desktop integrated environments. Example include the general-purpose Galaxy workbench and domain-specific ones such as MetaExplore and MicroScope.
• Typically use only one of *Bioinformatics portal*, *Database portal* or *Web application* in a single entry. If the resource is providing a database, then just go with *Database portal*, a good example being Norine.

• In general, often a single tool type will do. For example, LoRDEC (a *Command-line tool*), GINsim (a *Desktop application*), and Ocean Gene Atlas or Genomicus (both are *Web application*).

• But do pick all the types that apply. For example the BOOSTER *Command-line tool* is also available as a *Web application* (and if these implementations have essentially the same functionality, they’d be described in a single bio.tools entry).

• If a database has an API (most do!) then use both *Database portal* and *Web API*, for example aNISEED.

• Use the more specialised tool types where they are applicable, for example *Workflow* for Workflow4Metabolomics and *Library* for any R packages.

**Tip:** Software is complex and it can be tricky to assign a type. Make sure you understand the tool type definitions before you use them. For example, in *bio.tools* a *Web service* is specifically a SOAP+WSDL implementation. Most likely you need *Web API* (which covers most APIs nowadays) or just *Web application* (for a tool delivered via the Web but without an API).

---

**License**

All downloadable software should be licensed. If you can’t find your license in the list, use one of the terms below:

**Note:** There are many good reasons why you should license your software, ideally picking a FOSS (Free and Open Source Software) license. Read A Quick Guide to Software Licensing for the Scientist-Programmer. Some types of tools *e.g.* Web application” are not licensed, but instead, should have a Terms of use document. Proprietary licenses are definitely not open-source or, and should be avoided!

If your software is available under a license not supported by *bio.tools*, then please request the license is added. If you find yourself picking Not licensed - this is bad - license your software!

---

**Topic**

**Topic** is the place to tag your tool with EDAM terms describing the scientific domain the tool serves, or other general category.

- specify the most important and relevant scientific topic; up to 3 topics will usually suffice
- don’t exhaustively specify all the topics of lower or secondary relevance

**Tip:** Don’t rely on the term picker included in *bio.tools* to find topics and other EDAM terms - use the *EDAM browsers* or EDAM Tool Annotator instead. They are much more powerful!

---

**Function**

In the *bio.tools* software model, a tool has one or more basic functions, or modes of operation. Each *function* performs at least one *operation*, and has one or more primary *inputs* and *outputs*. Each input and output are of single defined *type of data* and list one or more supported *format(s)*.

This is shown in a diagram on the Tool Cards that look like this:
For example, the tool signalp has a single function performing two operations, with a single input and two outputs:

Whereas the tool HMMER3 has multiple functions (only 3 shown here):

Note: The HMMER3 entry has very nicely annotated functionality, but is a good example of where the entry would be easier to understand if the functionality was described in separate entries - retaining the existing entry for the suite, but creating a new entry for each of the HMMER programs (alimask, hmmalign, hmmbuild etc.).

In the Function tab you specify the functions of the tool, expressed in concepts from the EDAM ontology.

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Mandatory (Database portal, Web application, Desktop application, Command-line tool)</td>
</tr>
<tr>
<td>Input-&gt;data</td>
<td>Mandatory (Command-line tool)</td>
</tr>
<tr>
<td>Input-&gt;format</td>
<td>Recommended (Command-line tool)</td>
</tr>
<tr>
<td>Output-&gt;data</td>
<td>Mandatory (Database portal, Command-line tool)</td>
</tr>
<tr>
<td>Output-&gt;format</td>
<td>Recommended (Web application, Desktop application)</td>
</tr>
</tbody>
</table>

- **Operation** describes the basic operation(s) performed by this software function. See below {learn more}.
- **Data** is a type of primary input or output data. See below {learn more}.
- **Format** is the allowed format(s) of the input or output data. See below {learn more}.

Note: You can use Note to add a concise comment about this function, if this is not apparent from the software.
description and EDAM annotations.

**Tip:** When deciding how to describe your tools, in terms of *bio.tools* entries, their functions and operations, always keep the end-user in mind and try to describe your tools in a way that will be clear to them.

It can be difficult to find the right terms to describe a tools operation(s), input(s) or output(s). It’s highly recommended to use [EDAM Tool Annotator](http://edamontology.org) to describe the function, and carefully copy-paste the JSON output (in the bottom pane of the window) into the **JSON** tab of the *bio.tools* editing interface. Or use the OLS, BioPortal or EDAM Browser alongside *bio.tools* when describing your tools. If you’re not sure, mail registry-support for help.

**Operation**

Before describing your tools, you should carefully identify the distinct functions and the individual operations associated with each one. This is often straightforward, as different functions (modes) typically perform distinct operations:

- if a tool has an option between doing one thing or another, then you should annotate the operations as distinct functions
- if in contrast a tool always does one or more things, then you should annotate these as distinct operations within a single function
- only specify the primary functions and operations, from a typical end-user perspective - tools often do many other things than its central, advertised purpose - you don’t need to describe everything!

**Tip:** *Database portal* usually provide one ore more of a common set of operations:

- **Browse** - no term in EDAM yet
- **Deposit** - [Deposition](http://edamontology.org/operation_3431)
- **Search** - [Database search](http://edamontology.org/operation_2421)
- **Visualise** - [Visualisation](http://edamontology.org/operation_0337)
- **Analyse** - [Analysis](http://edamontology.org/operation_2945)
- **Download** - [Data retrieval](http://edamontology.org/operation_2422)

When annotating the operations, you should specify all of these that apply. Consider carefully whether the **Analyse** operation(s) would be better listed as functions of discrete tools described in their own own entries (see *bio.tools* entries).

**Data**

- data terms must be correctly associated with the operation(s)
- only specify the primary inputs and outputs, *e.g.* a sequence alignment tool would be annotated as reading sequences (input), and writing a sequence alignment (output), but not with gap insertion and extension penalties, or other parameters.

**Tip:** For *Database portal*:

- for **Deposition** and **Data retrieval** operations, you can associate the types of *data* available for upload (input) or download (output).
• for Search operation, you can specify Database search results (http://edamontology.org/data_2080) as an output, or some other more specific term in the EDAM Data subontology.

Format

• format terms must be correctly associated with an input or output data type
• specify the most widely used of the supported data formats - it can be impractical / onerous to be exhaustive!

Links

In the Links tab you specify miscellaneous links for the tool. The type of information obtained when resolving the link is specified by Link type:

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td><strong>Recommended</strong> (Desktop application, Command-line tool)</td>
</tr>
<tr>
<td>Mailing list</td>
<td><strong>Recommended</strong></td>
</tr>
<tr>
<td>Issue tracker</td>
<td><strong>Recommended</strong> (Database portal, Web application, Desktop application, Command-line tool)</td>
</tr>
<tr>
<td>Helpdesk</td>
<td><strong>Recommended</strong> (Database portal)</td>
</tr>
</tbody>
</table>

• **Repository** is where source code, data and other files may be downloaded, *e.g.* a GitHub repo, or an FTP site.
• **Mailing list** is for software announcements, discussions, support *etc.*
• **Issue tracker** is for software issues, bug reports, feature requests *etc.*
• **Helpdesk** is a phone line, web site or email-based system providing help to the end-user of the software.

**Tip:** A single link might resolve to a page containing information of more than one type; in these cases pick all of the types that apply!

**Note:** It’s strongly recommended to put your source code and other downloadable resources in a public repository such as GitHub. It takes little effort to do so. A repo can serve as a homepage for your tool, and provide an issue tracker and open forum for discussion. If you don’t have a repo, you should at least provide a downloads page.

**Caution:** Currently, to assign a link to more than one type you have to enter the URL more than once, picking a different type each type. In future, you’ll be able to enter the URL once and pick multiple types.

Download

In the Download tab you specify links to downloads for your software.
### TYPE | REQUIREMENT
--- | ---
Source code | **Recommended** (Database portal, Web application, Desktop application, Command-line tool)
Binaries | **Recommended** (Desktop application, Command-line tool)
Software package | **Recommended** (Desktop application, Command-line tool)
Downloads page | **Recommended** (Database portal, Desktop application)
API specification | **Recommended** (Database portal - with API)
Test data | **Recommended** (Desktop application, Command-line tool)
Test script | **Recommended** (Command-line tool)

- **Source code** should trigger a download of the *latest* source code available (typically the latest stable version)
- **Binaries** should trigger a download of the *latest* binaries available
- **Software package** should trigger a download of the *latest* software package.
- **Downloads page** is a Web page summarising general downloads available for the software.
- **API specification** is a file providing a machine-readable API specification for the software, *e.g.* Swagger/OpenAPI, WSDL or RAML file. It’s *not* for human-readable API documentation (see *documentation* for that).
- **Test data** is data for testing the software is working correctly.
- **Test script** is a script used for testing testing whether the software is working correctly.

**Tip:** With the exception of **Downloads page**, the expectation is that a link annotated in the **Download** section will trigger a download of a file. If you’re adding a link which doesn’t have this behaviour, you should see whether an attribute in the **Links** section is more appropriate.

**Note:** **Command-line specification** and **API specification** are files providing a machine-readable specification of the command line or API, for the software. These are *not* used for the typical human-readable documentation (see *Documentation* for that).

### Documentation

In the **Documentation** tab you link to documentation about the software:
<table>
<thead>
<tr>
<th>TYPE</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Mandatory (Database portal, Web application, Desktop application, Command-line tool)</td>
</tr>
<tr>
<td>API documentation</td>
<td>Mandatory (Database portal or Web application - with API)</td>
</tr>
<tr>
<td>Terms of use</td>
<td>Mandatory (Bioinformatics portal, Database portal, Web application)</td>
</tr>
<tr>
<td>Command-line options</td>
<td>Mandatory (Command-line tool)</td>
</tr>
<tr>
<td>Citation instructions</td>
<td>Recommended</td>
</tr>
<tr>
<td>Contributions policy</td>
<td>Recommended</td>
</tr>
<tr>
<td>Training material</td>
<td>Recommended</td>
</tr>
<tr>
<td>Installation instructions</td>
<td>Recommended (Desktop application, Command-line tool)</td>
</tr>
<tr>
<td>User manual</td>
<td>Recommended (Desktop application)</td>
</tr>
<tr>
<td>Release notes</td>
<td>Recommended (Desktop application, Command-line tool)</td>
</tr>
</tbody>
</table>

- **General** is for general documentation. If your tool doesn’t have a dedicated documentation page, but is documented elsewhere (e.g. on the homepage or a GitHub README.md) then specify that URL instead.

- **API documentation** is human-readable API documentation, and should be specified for any Database portal or Web application with an API.

- **Terms of use** are rules that one must agree to abide by in order to use a service. Note, this is different to License!

- **Command-line options** are human-readable documentation about the command-line interface of a tool.

- **Citation instructions** give information on how to correctly cite use of the software; typically which publication(s) to cite, or something more general, e.g. a form of words to use. This is especially important where there are multiple relevant publications.

- **Contributions policy** is information about policy for making contributions to the software project.

- **Training material** is an online training material such as a tutorial, a presentation, video etc.

- **Installation instructions** are instructions how to install the software.

- **User manual** is information on how to use the software, structured into a comprehensive user manual (don’t just link here to general documentation).

- **Release notes** are notes about a software release or changes to the software (a change log). For example a CHANGELOG.md file on GitHub.

**Note:** You should create contribution guidelines to communicate how people should contribute to your open source project. In GitHub this is done by creating a CONTRIBUTING.md file. Lots of good advice, templates and examples are available (e.g. Atom editor, Ruby on Rails and Open Government).

A well maintained change log will make it easier for users and contributors to see precisely what notable changes have been made between each release (or version) of the project. For some great advice, see keepachangelog.com.

**Note:** Command-line tools should always have a human-readable description of their command-line options. Similarly, an API on a Database portal or Web application should have a human-readable description of their API. If machine-readable command-line or API specifications (files) are also available, then you should link to those in the Download section.
Important: You must not specify a link to a general page where a more specific one is available. For example, don’t link to the homepage in the General field if, in fact, there’s a dedicated page for documentation. If you want to link to some documentation not of a type supported by biotoolsSchema, then use the Other value.

Caution: Currently, to assign a documentation to more than one type you have to enter the URL more than once, picking a different type each type. In future, you’ll be able to enter the URL once and pick multiple types.

Publications

In the Publications tab you specify publications about the software:

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary publication</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

Publications are defined as one of the following types:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>The principal publication about the tool itself; the article to cite when acknowledging use of the tool.</td>
</tr>
<tr>
<td>Method</td>
<td>A publication describing a scientific method or algorithm implemented by the tool.</td>
</tr>
<tr>
<td>Usage</td>
<td>A publication describing the application of the tool to scientific research, a particular task or dataset.</td>
</tr>
<tr>
<td>Benchmarking study</td>
<td>A publication which assessed the performance of the tool.</td>
</tr>
<tr>
<td>Review</td>
<td>A publication where the tool was reviewed.</td>
</tr>
<tr>
<td>Other</td>
<td>A publication of relevance to the tool but not fitting the other categories.</td>
</tr>
</tbody>
</table>

and can have the following attributes defined:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmcid</td>
<td>PubMed Central Identifier of a publication about the software.</td>
</tr>
<tr>
<td>pmid</td>
<td>PubMed Identifier.</td>
</tr>
<tr>
<td>doi</td>
<td>Digital Object Identifier.</td>
</tr>
<tr>
<td>note</td>
<td>Comment about the publication.</td>
</tr>
<tr>
<td>version</td>
<td>Version information (typically a version number) of the software applicable to this publication.</td>
</tr>
</tbody>
</table>

- Specify at least the primary publication for your tool, and ideally any others that are relevant.
- Pick one or more types for each publication, as applicable.

Tip: You should specify DOI for publications (if available) and do not have to also specify pmid and pmcid. But if you do, then be sure to specify multiple IDs for a single publication within a single publication group.

You can ignore note and version.
Note: It’s very important that your tool has some form of publication, if for no other reason than to make it citable. If you don’t have a publication in the scientific press, then you can use Zenodo to create a DOI for this purpose. Such a DOI should resolve to a page describing the tool. For example http://doi.org/10.5281/zenodo.3519603.

Caution: Currently, to assign a publication to more than one type you have to enter the DOI more than once, picking a different type each time. In future, you’ll be able to enter the DOI once and pick multiple types.

Credits & Support

In the Credits & Support tab you specify individuals or organisations that should be credited, or may be contacted about the software. Credits include all type of entities that contributed to the development, maintenance or provision of the resource:

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>REQUIREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary contact</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Credited institute</td>
<td>Recommended</td>
</tr>
<tr>
<td>Credited funding agency</td>
<td>Recommended</td>
</tr>
<tr>
<td>Credited developer</td>
<td>Recommended</td>
</tr>
<tr>
<td>Credited maintainer</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

Creditable entities have one of the following types:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Credit of an individual.</td>
</tr>
<tr>
<td>Project</td>
<td>Credit of a community software project not formally associated with any single institute.</td>
</tr>
<tr>
<td>Division</td>
<td>Credit of or a formal part of an institutional organisation, e.g. a department, research group, team, etc</td>
</tr>
<tr>
<td>Institute</td>
<td>Credit of an organisation such as a university, hospital, research institute, service center, unit etc.</td>
</tr>
<tr>
<td>Consortium</td>
<td>Credit of an association of two or more institutes or other legal entities which have joined forces for some common purpose. Includes Research Infrastructures (RIs) such as ELIXIR, parts of an RI such as an ELIXIR node etc.</td>
</tr>
<tr>
<td>Funding agency</td>
<td>Credit of a legal entity providing funding for development of the software or provision of an online service.</td>
</tr>
</tbody>
</table>

and also have a role:

<table>
<thead>
<tr>
<th>ROLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>Author of the original software source code.</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Maintainer of a mature software providing packaging, patching, distribution etc.</td>
</tr>
<tr>
<td>Provider</td>
<td>Institutional provider of an online service.</td>
</tr>
<tr>
<td>Documentor</td>
<td>Author of software documentation including making edits to a bio.tools entry.</td>
</tr>
<tr>
<td>Contributor</td>
<td>Some other role in software production or service delivery including design, deployment, system administration, evaluation, testing, documentation, training, user support etc.</td>
</tr>
<tr>
<td>Support</td>
<td>Provider of support in using the software.</td>
</tr>
<tr>
<td>Primary contact</td>
<td>The primary point of contact for the software.</td>
</tr>
</tbody>
</table>
You must specify at least:

- A credit of role **Primary contact** with an applicable type. You can opt to give more than one primary contact, for example specifying one for a project and another for a person.

It’s recommended to specify:

- A credit of type **Institute** with one or more applicable roles
- A credit of type **Funding agency**
- A credit of role **Developer** with one or more applicable types
- A credit of role **Maintainer** with one or more applicable types

For any credit, you can specify any of the following:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity that is credited.</td>
</tr>
<tr>
<td>orcidid</td>
<td>Unique identifier (ORCID iD) of a person that is credited.</td>
</tr>
<tr>
<td>gridid</td>
<td>Unique identifier (GRID ID) of an organisation that is credited.</td>
</tr>
<tr>
<td>email</td>
<td>Email address.</td>
</tr>
<tr>
<td>url</td>
<td>URL, e.g. homepage of an institute.</td>
</tr>
<tr>
<td>tel</td>
<td>Telephone number.</td>
</tr>
<tr>
<td>typeEntity</td>
<td>Type of entity that is credited (see above)</td>
</tr>
<tr>
<td>typeRole</td>
<td>Role performed by entity that is credited (see above)</td>
</tr>
<tr>
<td>note</td>
<td>A comment about the credit.</td>
</tr>
</tbody>
</table>

**Important:** A credit can have multiple role. When creating a credit, pick all of the roles that apply; don’t create duplicate credit groupings!

**Note:** It’s strongly recommended that if you (or other people to be credited) don’t have an ORCID iD, that you get one now. ORCID provides a persistent digital identifier that distinguishes you from every other researcher and, through integration in key research workflows such as manuscript and grant submission, supports automated linkages between you and your professional activities ensuring that your work is recognized.

**Note:** Nearly all organisations credited in *bio.tools* will have a GRID ID. The Global Research Identifier Database (GRID) provides unambiguous institutional information at persistent IDs, to ensure data consistency.

**Relations**

In the **Relations** tab you can specify details of a relationship this software shares with other software registered in *bio.tools*.

The relationships currently available:
<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isNewVersionOf</td>
<td>The software is a new version of an existing software, typically providing new or improved functionality.</td>
</tr>
<tr>
<td>hasNewVersion</td>
<td>(inverse of above)</td>
</tr>
<tr>
<td>uses</td>
<td>The software provides an interface to or in some other way uses the functions of other software under the hood, e.g. invoking a command-line tool or calling a Web API, Web service or SPARQL endpoint to perform its function.</td>
</tr>
<tr>
<td>usedBy</td>
<td>(inverse of above)</td>
</tr>
<tr>
<td>includes</td>
<td>A workbench, toolkit or workflow includes some other, independently available, software.</td>
</tr>
<tr>
<td>includedIn</td>
<td>(inverse of above)</td>
</tr>
</tbody>
</table>

You can ignore this for now, except:

- when annotating a *Suite* (or other collection) it’s recommended to specify other tools that the suite **includes**
- when annotating a *Workflow* it’s recommended to specify other tools that the workflow **uses**

**JSON**

In the **JSON** tab you see all the information that you’ve specified for a tool so far. You can work directly in this pane if you wish. This can be very useful when using the EDAM Tool Annotator to define the tool’s function (see the section on *EDAM*.)

**Permissions**

In the **Permissions** tab you can decide to make the entry either editable only by yourself, a list of users or anyone. See the section on *bio.tools accounts*.

### 5.2 Workflow composition (Lorentz workshop)

These instructions provide a quick guide to the steps to register your tools in *bio.tools* and describe them using the EDAM ontology to a standard suitable for applications in workflow composition. They were written to support the workshop Automated Workflow Composition in the Life Sciences hosted by the Lorentz Center.

Help is available:

- To get help using *bio.tools*, or for general curation advice, please mail registry-support.
- To get help using the EDAM ontology, please head over to [GitHub 'https://github.com/edamontology/edamontology'] and post an issue.
- If you have questions specifically about curation of tools for the Lorentz workshop, you can mail Jon Ison directly.

**Note:** For more information and help, see:

- Guidelines for IFB tools cover similar ground to this document, but in more detail.
- The *bio.tools* Curators Guide provides in-depth curation guidelines.
- The biotoolsSchema documentation summarises the attributes, information model and controlled vocabularies - including the EDAM ontology - used by *bio.tools*. 

---

66 Chapter 5. Community-specific guidelines
The EDAM ontology documentation everything you need to know about contributing to, editing and developing EDAM.

If you find a bug in this document, or have any questions or suggestions, please post them on GitHub.

5.2.1 1. Get a bio.tools account

You’ll need an account to create bio.tools entries or edit existing ones. Creating an account is simple: just go to bio.tools and click on Sign-up at the top-right corner of the page.

You’ll be asked for a username, email address and password. Your account will be setup immediately.

Note: bio.tools entries are owned by the individuals who created them. Owners may grant edit rights, or transfer ownership of their entries to other registered users - these rights are set in the Permissions tab of the bio.tools editing interface.

The rightful owner of a bio.tools entry is usually the person who developed the tool, or provides an online service, but it can be some other responsible person, e.g. a dedicated curator.

5.2.2 2. Claim your bio.tools entries

As a software developer or service provider, you should own the bio.tools entries describing your tools, by claiming ownership of existing entries or creating new ones. Alternatively, if you just want to edit a tool description, than you can request edit rights for it.

You’ll need to login first, by clicking on Login at the top-right corner of the page.

To see whether a tool is already registered, search for it by its name. Simply type the name in the search box. You may need to click on the Name facet to narrow-down the search:

Once you find your entry you can go ahead and update it. If you can’t find the entry, you’ll need to create it.

5.2. Workflow composition (Lorentz workshop)
Important: Many of the tools you need already have been registered, but possibly with only very basic details. You will need to take ownership or request edit rights on these entries to improve them. Before starting work, please ensure you understand the information requirement and follow the guidelines below.

Updating entries

To edit an existing entry, you need to click through to the Tool Card for the tool in question, e.g. https://bio.tools/signalp. You’ll see a one or two buttons at the bottom right of the Tool Card, depending on whether you’re logged in, and own the entry or not.

- Click on Request ownership if you want to claim ownership of the entry
- Click on Request editing rights if you want to edit rights on the entry, but not own it
- Click on Update this record to edit the entry (visible only if you own the entry or have editing rights)

Note: It can take a little while for other users to respond to requests for edit rights or ownership. If these are not granted within a day or two, please mail registry-support.

Creating entries

To create new entries you’ll need to be logged onto bio.tools. Click on Menu … Add content:

bio.tools editing interface

The bio.tools editing interface helps you to create valid tool descriptions. It’s organised into different tabs (Summary, Function, Labels etc.):
The editing interface provides some hints, and ensures that the information you set is in the right format. At any moment, you can click on \[\text{Save}\] to save your edits, and immediately publish the changes online. The information you specified is checked to ensure it’s in the right syntax. To (optionally) force a manual syntax check, click on \[\text{Validate}\].

**Important:** The attributes required by \textit{bio.tools} (tool name, description and homepage URL) are marked with a red asterix \* in the editing interface, and must be given before an entry can be saved. You will need to provide more information to support workflow composition - especially about the tool function - but this is not enforced by \textit{bio.tools}!

**Note:** It’s possible to create tool descriptions in JSON format directly in a text editor and paste these into the editing interface (“JSON” tab), where you see all the information that you’ve specified for a tool so far. You can work directly in this pane if you wish. This can be very useful when using the EDAM Tool Annotator to define the tool’s function (see the section on \textit{EDAM}.)

**Removing entries**

To remove an entry, click on \textit{Update this record} button (bottom right of the Tool Card). Then you can remove the entry by clicking on \[\text{Remove}\].
5.2.3 3. Plan your curation work

Information requirement

`bio.tools` requires only the name, description and homepage URL for a tool registration, but supports a comprehensive set of attributes for rich tool descriptions.

To support workflow composition, very careful curation of tool functionality - it’s input(s), operation(s) and output(s) - is required. The guidelines below cover exactly what’s needed, and go through the curation process in a step-by-step way.

**Note:** The attributes supported by `bio.tools`, their structure and their syntax are defined in formalised XML schema called `biotoolsSchema`. You don’t need to look at the schema, because everything is handled through `bio.tools`. If you’d like to learn more or contribute to this project, please head over to GitHub.

**Before you start**

**Important:** Before you use `bio.tools` to create and edit tool descriptions, be sure to understand:

1. The type of tool being described - covered below in the section below on Basic information.
2. The tool functionality and how it should be described - covered in the section on tool functions.
3. Whether one or more entries are needed (see below).

Plan what new entries (if any) are required to describe your tools:

- A discrete tool - one which is clearly an individual, distinct entity - should have it’s own entry. This is the case for most command-line tools and desktop applications.

- `bio.tools` aims to catalogue unique tool functionality. Different implementations but with essentially the same functionality can be described by a single entry, e.g. a command-line tool that is later adapted into an R package for the Bioconductor suite, or which is served online via a Galaxy server.

- In some cases, e.g. complex software packages, it’s not obvious whether to have one or multiple entries. Pick the option which mostly clearly illustrates the tool’s functionality to end-users.

- Tool collections should be described by multiple entries. For example, one entry to describe a suite, and multiple other entries to describe the individual tools within that suite.

- Software with multiple interfaces should be described by a single entry, assuming these interfaces have essentially the same functionality. For example, a command-line tool whose functionality is also available via a web application, or a database portal with a web API.

For more information, see the IFB tools tutorial.
Familiarise yourself with EDAM

The EDAM ontology provides bio.tools with a controlled vocabulary to describe the scientific function of a tool, including the general scientific domain, specific operations it performs, types of input and output data, and supported data formats.
The EDAM ontology includes four main types of concept (or subontologies), shown in boxes above. The concepts are Topic, Operation, Data and Format, with Identifier being a specialisation of Data. Relationships between EDAM concepts are defined internally within the ontology. You don’t need to worry about these details, as it’s all handled by bio/tools.

**Picking EDAM terms**

Three EDAM browsers, each with different functionality, can be used to find EDAM terms:

- OLS
- BioPortal
- EDAM Browser

**Tip:** The EDAM term picker currently implemented in bio.tools is not very powerful. It’s strongly recommended to use the browsers above. If you can’t find exactly the terms you need, multiple searches using synonyms, alternative spellings etc. can help.

A much better term picker is on the way, and while not yet fully integrated into bio.tools is already very useful:

- EDAM Tool Annotator
You can use this to pick relevant topics and define the function of your tools. The output (in the bottom pane of the window) is a JSON object that can be copy-pasted into the JSON tab of the bio.tools editing interface, when editing a tool description.

If you can’t find the right term, please request that it’s added to EDAM via GitHub but first read the guidelines on how to request a term. It takes some time for new terms to be supported in bio.tools, so if you need many new terms, please plan ahead and contact the EDAM developers if you need help.

### 5.2.4 4. Describe your tools

**Note:** Only those tool attributes that are *Mandatory* in bio.tools, or which are required to support workflow composition are described below, but you can of course specify much more. The {learn more} links take you to more detailed guidelines in IFB tools or the bio.tools Curators Guide. Follow these links whenever you’re not sure about what information is needed.

#### Basic information

In the Summary tab you specify basic information about the software:

- **Name** is the short-form name by which the tool is commonly known, e.g. “BLAST” not “Basic Local Alignment Search Tool”. Database names should follow a pattern where the name and abbreviation are given e.g. “The Protein Databank (PDB)” {learn more}.

- **Description** is a concise textual summary of the tool function or purpose. It can usually be copy-pasted from the tool homepage. Do not include statements about performance, provenance, governance etc. {learn more}.

- **Homepage URL** is the tool’s homepage, or some URL that best serves this purpose {learn more}.

**Important:** A unique identifier - the bio.tools **toolID** - is created for a tool when a new entry is created. The ID value is a URL-safe version of the supplied tool name. The ID provides a persistent reference to the tool, used by bio.tools and other systems. toolIDs are used in the Tool Card URLs, which can be represented in a short form as a “compact URI” or “CURIE”:

- **toolID**: signalp

- **CURIE**: biotools:signalp

- **Tool Card URL**: https://biotools/signalp

The ID should be sensible and intuitive. For databases, or tools with long names, the abbreviation should be used. For example, the GnplS tool has the ID “gnpis” and not “Genetic and Genomic Information System”.

**Tip:** The toolID is not currently editable, so if you want the ID to differ from the name (e.g. an ID of “PDB” for the tool name “Protein databank (PDB)”), you have to apply a workaround:

1) create the entry giving a value for “Name” which is the desired ID value, e.g. “PDB”

2) Save the entry

3) Edit the entry, resetting the name, e.g. to “Protein Databank (PDB)”

To request an ID change post-registration (to be avoided!) you have to mail Registry Support.
In the *Labels* tab you specify miscellaneous scientific, technical and administrative details, expressed in terms from controlled vocabularies. For now, annotate the following:

- **Tool type** is where you specify the type(s) of software that is described by the entry. A single *bio.tools* entry may be annotated with one or more types, reflecting different facets of the tool described by the entry. [Learn more].

- **Topic** is the place to tag your tool with EDAM terms describing the scientific domain the tool serves, or other general category.
  - specify the most important and relevant scientific topic; up to 3 topics will usually suffice
  - don’t exhaustively specify all the topics of lower or secondary relevance

**Note:** The scope of *bio.tools* is very broad - ranging from simple scripts to comprehensive bioinformatics portals - as defined by 15 different tool types. The vast majority of tools used to compose workflows are of the following types:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-line tool</td>
<td>A tool with a text-based (command-line) interface.</td>
</tr>
<tr>
<td>Web application</td>
<td>A tool with a graphical user interface that runs in your Web browser.</td>
</tr>
<tr>
<td>Desktop application</td>
<td>A tool with a graphical user interface that runs on your desktop environment, e.g. on a PC or mobile device.</td>
</tr>
<tr>
<td>Web API</td>
<td>An application programming interface (API) consisting of endpoints to a request-response message system accessible via HTTP. Includes everything from simple data-access URLs to RESTful APIs.</td>
</tr>
</tbody>
</table>

**Tip:** Don’t rely on the term picker included in *bio.tools* to find topics and other EDAM terms - use the *EDAM browsers* or *EDAM Tool Annotator* instead. They are much more powerful!

**Function**

In the *bio.tools* software model (*biotoolsSchema*), a tool has one or more basic functions, or modes of operation. Each function performs at least one operation, and has one or more primary inputs and outputs. Each input and output are of single defined type of data and list one or more supported format(s).

This is shown in a diagram on the Tool Cards that look like this:

For example, the tool *signalp* has a single function performing two operations, with a single input and two outputs:
Whereas the tool HMMER3 has multiple functions (only 3 shown here):

Note: The HMMER3 entry has very nicely annotated functionality, but is a good example of where the entry would be easier to understand if the functionality was described in separate entries - retaining the existing entry for the suite, but creating a new entry for each of the HMMER programs (alimask, hmmalign, hmmbuild etc.).

In the Function tab you specify the functions of the tool, expressed in concepts from the EDAM ontology.

- **Operation** describes the basic operation(s) performed by this software function. See below {learn more}.
- **Data** is a type of primary input or output data. See below {learn more}.
- **Format** is the allowed format(s) of the input or output data. See below {learn more}.

Note: You can use Note to add a concise comment about this function, if this is not apparent from the software description and EDAM annotations.

Tip: When deciding how to describe your tools, in terms of bio.tools entries, their functions and operations, always keep the end-user in mind and try to describe your tools in a way that will be clear to them.

It can be difficult to find the right terms to describe a tools operation(s), input(s) or output(s). It’s highly recommended to use EDAM Tool Annotator to describe the function, and carefully copy-paste the JSON output (in the bottom pane of the window) into the JSON tab of the bio.tools editing interface. Or use the OLS, BioPortal or EDAM Browser alongside bio.tools when describing your tools. If you’re not sure, mail registry-support for help.

### Operation

Before describing your tools, you should carefully identify the distinct functions and the individual operations associated with each one. This is often straightforward, as different functions (modes) typically perform distinct operations:

- if a tool has an option between doing one thing or another, then you should annotate the operations as distinct functions
• if in contrast a tool always does one or more things, then you should annotate these as distinct operations within a single function

• only specify the primary functions and operations, from a typical end-user perspective - tools often do many other things than its central, advertised purpose - you don’t need to describe everything!

Data

• data terms must be correctly associated with the operation(s)

• only specify the primary inputs and outputs, e.g. a sequence alignment tool would be annotated as reading sequences (input), and writing a sequence alignment (output), but not with gap insertion and extension penalties, or other parameters.

Format

• format terms must be correctly associated with an input or output data type

• specify the most widely used of the supported data formats - it can be impractical / onerous to be exhaustive!

Other fields

Other sections in the bio.tools editing interface include:

• Labels - specify miscellaneous scientific, technical and administrative details, expressed in terms from controlled vocabularies. [learn more]

• Links - specify miscellaneous links for the tool. The type of information obtained when resolving the link is specified by Link type. [learn more]

• Download - specify links to downloads for your software. [learn more]

• Documentation - link to documentation about the software. [learn more]

• Publications - specify publications about the software. [learn more]

• Credits & Support - specify individuals or organisations that should be credited, or may be contacted about the software. Credits include all type of entities that contributed to the development, maintenance or provision of the resource. [learn more]

• Relations - specify details of a relationship this software shares with other software registered in bio.tools. [learn more]
Thematic editors are registry-dev members responsible for overseeing coverage and quality in specific thematic areas. The editorships will enable expanding accurate high-standard software annotations in bio.tools to most scientific topics in the life sciences.

..note:: The concept of “thematic editors” is a work in progress which we will develop as funding and capacity permits.

### 6.1 Background

The sheer number of software and continuous advancements in tool development demand extensive and sustained efforts to provide comprehensive annotations. Such challenges can be overcome by distributing the curation effort across a network of engaged and experienced partners, which contribute to improve bio.tools content and to adjust the EDAM vocabulary (used in bio.tools) to the needs of the respective communities. Careful coordination of the distributed tasks is required to maintain quality standards and increase tool interoperability across the board.

Thematic editors are well connected with their respective scientific community, experts in their field, have a broad knowledge about commonly used software and are motivated to promote bio.tools. Editors normally oversee the work of one or more student curators, on a range of tasks to improve EDAM and bio.tools content:

- Review of bio.tools and EDAM to survey coverage of concepts, terms and tools.
- Help develop strategy to achieve and sustain a minimum information level in bio.tools, as per the emerging information standard.
- Engage with their community, supervise hackathons and promote bio.tools in general.
- Mentor a student for practical curation work
- Implement sustainable procedures for systematic tool reviews and curation standards.

**Benefits:** A thematic editorships provides the opportunity to become a curation expert and contribute to a project for the benefit of the whole bioinformatics community. Editors will acquire extensive knowledge and experience about available software in their fields e.g. with opportunity to publish high-quality reviews. The editors will train additional experts during training and curation events (e.g. hackathons) as well as via student supervision. Moreover, the ELIXIR infrastructure will support them to amplify their activities within a broader context.
6.2 Candidate thematic editors

We have expressions of interest from the following people:

<table>
<thead>
<tr>
<th>Editor</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anne Wenzel</td>
<td>RNA structure</td>
</tr>
<tr>
<td>Jon Ison</td>
<td>General purpose sequence analysis</td>
</tr>
<tr>
<td>Jose M. Carazo</td>
<td>Electron microscopy, structural biology</td>
</tr>
<tr>
<td>Josep Gelpi</td>
<td>Structural bioinformatics</td>
</tr>
<tr>
<td>Jurgen Haas</td>
<td>Protein structural biology, structural bioinformatics</td>
</tr>
<tr>
<td>Marta Villegas</td>
<td>Natural language processing</td>
</tr>
<tr>
<td>Martin Krallinger</td>
<td>Text Mining, natural language processing, named entity recognition</td>
</tr>
<tr>
<td>Reza Salek</td>
<td>Metabolomics</td>
</tr>
<tr>
<td>Veit Schwämmle</td>
<td>Proteomics, statistics</td>
</tr>
<tr>
<td>Vivi R. Gregersen</td>
<td>Agricultural science (association analysis, genomics)</td>
</tr>
</tbody>
</table>

People listed below, in the past served as EDAM Editors; we are aiming for a single, consolidated group of Thematic Editors serving both EDAM and bio.tools.

<table>
<thead>
<tr>
<th>Editor</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>David Sehnal</td>
<td>General bioinformatics</td>
</tr>
<tr>
<td>Dmitri Repchevsky</td>
<td>ES tools &amp; services</td>
</tr>
<tr>
<td>Ivan Mičetić</td>
<td>Protein structure</td>
</tr>
<tr>
<td>Laura Emery</td>
<td>EBI tools and training</td>
</tr>
<tr>
<td>Lukáš Pravda</td>
<td>Structural bioinformatics</td>
</tr>
<tr>
<td>Stanislav Geidl</td>
<td>Chemoinformatics</td>
</tr>
</tbody>
</table>

bio.tools studentships can support thematic editors. A typical studentship comprises (at least) one full month of full working hours which are mostly distributed over a longer time period. Applications are written on basis of the template (see e.g. proteomics studentship). The students are recruited after proposal dissemination via GitHub, project mailing lists etc.
The bio.tools Web API provides an easy way to access the bio.tools database.

### 7.1 List tools

`List and search through all the available tools. Can sort, filter, and search the results.`

**HTTP GET**

```
https://bio.tools/api/tool/
https://bio.tools/api/t/
```
7.1.1 Endpoint parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>No</td>
<td>Integer</td>
<td>1</td>
<td>Result page number</td>
</tr>
<tr>
<td>format</td>
<td>No</td>
<td>String(json, api)</td>
<td>json</td>
<td>Response media type</td>
</tr>
<tr>
<td>q</td>
<td>No</td>
<td>String</td>
<td></td>
<td>Query term, used for searching, matches all attributes</td>
</tr>
<tr>
<td>sort</td>
<td>No</td>
<td>String(lastUpdate, additionDate, name, affiliation, score)</td>
<td>lastUpdate</td>
<td>Sorts the results by chosen value (score only available when there is a query)</td>
</tr>
<tr>
<td>ord</td>
<td>No</td>
<td>String(desc, asc)</td>
<td>desc</td>
<td>Orders the results by either Ascending or Descending order</td>
</tr>
<tr>
<td>&lt;attribute&gt;</td>
<td>No</td>
<td>String</td>
<td></td>
<td>Filter by &lt;attribute&gt;. List of supported attributes below.</td>
</tr>
</tbody>
</table>

7.1.2 Filtering

To filter the results by attribute name, the attribute name has to be added as a parameter to the URL, with the value being the desired search term, e.g. ?name=signalp

Attributes

These are the attributes supported by bio.tools:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Search behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>biotoolsID</td>
<td>Search for bio.tools tool ID (usually quoted - to get exact match) biotoolsID=&quot;signalp&quot;</td>
</tr>
<tr>
<td>name</td>
<td>Search for tool name (quoted as needed) name=signalp</td>
</tr>
<tr>
<td>homepage</td>
<td>Exact search for tool homepage URL (must be quoted) homepage=&quot;<a href="http://cbs.dtu.dk/services/SignalP/">http://cbs.dtu.dk/services/SignalP/</a>&quot;</td>
</tr>
<tr>
<td>description</td>
<td>Search over tool description (quoted as needed) description=&quot;peptide cleavage&quot;</td>
</tr>
<tr>
<td>version</td>
<td>Exact search for tool version (must be quoted) version=&quot;4.1&quot;</td>
</tr>
<tr>
<td>topic</td>
<td>Search for EDAM Topic (term) (quoted as needed) topic=&quot;Proteomics&quot;</td>
</tr>
<tr>
<td>topicID</td>
<td>Exact search for EDAM Topic (URI): must be quoted topicID=&quot;topic_3510&quot;</td>
</tr>
<tr>
<td>function</td>
<td>Fuzzy search over function (input, operation, output, note and command) function=&quot;Sequence analysis&quot;</td>
</tr>
<tr>
<td>operation</td>
<td>Fuzzy search for EDAM Operation (term) (quoted as needed) operation=&quot;Sequence analysis&quot;</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Search behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>operationID</td>
<td>Exact search for EDAM Operation (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>operationID=&quot;operation_2403&quot;</td>
</tr>
<tr>
<td>dataType</td>
<td>Fuzzy search over input and output for EDAM Data (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>dataType=&quot;Protein sequence&quot;</td>
</tr>
<tr>
<td>dataTypeID</td>
<td>Exact search over input and output for EDAM Data (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>dataTypeID=&quot;data_2976&quot;</td>
</tr>
<tr>
<td>dataFormat</td>
<td>Fuzzy search over input and output for EDAM Format (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>dataFormat=&quot;FASTA&quot;</td>
</tr>
<tr>
<td>dataFormatID</td>
<td>Exact search over input and output for EDAM Format (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>dataFormatID=&quot;format_1929&quot;</td>
</tr>
<tr>
<td>input</td>
<td>Fuzzy search over input for EDAM Data and Format (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>input=&quot;Protein sequence&quot;</td>
</tr>
<tr>
<td>inputID</td>
<td>Exact search over input for EDAM Data and Format (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>inputID=&quot;data_2976&quot;</td>
</tr>
<tr>
<td>inputDataType</td>
<td>Fuzzy search over input for EDAM Data (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>inputDataType=&quot;Protein sequence&quot;</td>
</tr>
<tr>
<td>inputDataTypeID</td>
<td>Exact search over input for EDAM Data (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>inputDataTypeID=&quot;data_2976&quot;</td>
</tr>
<tr>
<td>inputDataFormat</td>
<td>Fuzzy search over input for EDAM Format (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>inputDataFormat=&quot;FASTA&quot;</td>
</tr>
<tr>
<td>inputDataFormatID</td>
<td>Exact search over input for EDAM Format (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>inputDataFormatID=&quot;format_1929&quot;</td>
</tr>
<tr>
<td>output</td>
<td>Fuzzy search over output for EDAM Data and Format (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>output=&quot;Sequence alignment&quot;</td>
</tr>
<tr>
<td>outputID</td>
<td>Exact search over output for EDAM Data and Format (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>outputID=&quot;data_0863&quot;</td>
</tr>
<tr>
<td>outputDataType</td>
<td>Fuzzy search over output for EDAM Data (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>outputDataType=&quot;Sequence alignment&quot;</td>
</tr>
<tr>
<td>outputDataTypeID</td>
<td>Exact search over output for EDAM Data (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>outputDataTypeID=&quot;data_0863&quot;</td>
</tr>
<tr>
<td>outputDataFormat</td>
<td>Fuzzy search over output for EDAM Format (term) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>outputDataFormat=&quot;ClustalW format&quot;</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Search behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputDataFormatID</td>
<td>Exact search over output for EDAM Format (ID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>outputDataFormatID=&quot;format_1982&quot;</td>
</tr>
<tr>
<td>toolType</td>
<td>Exact search for tool type</td>
</tr>
<tr>
<td></td>
<td>toolType=&quot;Command-line tool&quot;</td>
</tr>
<tr>
<td>collectionID</td>
<td>Exact search for tool collection (normally quoted)</td>
</tr>
<tr>
<td></td>
<td>collectionID=&quot;Rare Disease&quot;</td>
</tr>
<tr>
<td>maturity</td>
<td>Exact search for tool maturity</td>
</tr>
<tr>
<td></td>
<td>maturity=&quot;Mature&quot;</td>
</tr>
<tr>
<td>operatingSystem</td>
<td>Exact search for tool operating system</td>
</tr>
<tr>
<td></td>
<td>operatingSystem=&quot;Linux&quot;</td>
</tr>
<tr>
<td>language</td>
<td>Exact search for programming language</td>
</tr>
<tr>
<td></td>
<td>language=&quot;Java&quot;</td>
</tr>
<tr>
<td>cost</td>
<td>Exact search for cost</td>
</tr>
<tr>
<td></td>
<td>cost=&quot;Free of charge&quot;</td>
</tr>
<tr>
<td>license</td>
<td>Exact search for software or data usage license (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>license=&quot;GPL-3.0&quot;</td>
</tr>
<tr>
<td>accessibility</td>
<td>Exact search for tool accessibility</td>
</tr>
<tr>
<td></td>
<td>accessibility=&quot;Open access&quot;</td>
</tr>
<tr>
<td>credit</td>
<td>Fuzzy search over credit (name, email, URL, ORCID iD, type of entity, type of role and note)</td>
</tr>
<tr>
<td></td>
<td>credit=&quot;Henrik Nielsen&quot;</td>
</tr>
<tr>
<td>creditName</td>
<td>Exact search for name of credited entity</td>
</tr>
<tr>
<td></td>
<td>creditName=&quot;Henrik Nielsen&quot;</td>
</tr>
<tr>
<td>creditTypeRole</td>
<td>Exact search for role of credited entity</td>
</tr>
<tr>
<td></td>
<td>creditTypeRole=&quot;Developer&quot;</td>
</tr>
<tr>
<td>creditTypeEntity</td>
<td>Exact search for type of credited entity</td>
</tr>
<tr>
<td></td>
<td>creditTypeEntity=&quot;Funding agency&quot;</td>
</tr>
<tr>
<td>creditOrcidID</td>
<td>Exact search for ORCID iD of credited entity <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>creditOrcidID=&quot;0000-0001-5121-2036&quot;</td>
</tr>
<tr>
<td>publication</td>
<td>Fuzzy search over publication (DOI, PMID, PMCID, publication type and tool version) (quoted as needed)</td>
</tr>
<tr>
<td></td>
<td>publication=10.12688/f1000research.12974.1</td>
</tr>
<tr>
<td>publicationID</td>
<td>Exact search for publication ID (DOI, PMID or PMCID) <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>publicationID=&quot;10.12688/f1000research.12974.1&quot;</td>
</tr>
<tr>
<td>publicationType</td>
<td>Exact search for publication type</td>
</tr>
<tr>
<td></td>
<td>publicationType=&quot;Primary&quot;</td>
</tr>
<tr>
<td>publicationVersion</td>
<td>Exact search for tool version associated with a publication <em>(must be quoted)</em></td>
</tr>
<tr>
<td></td>
<td>publicationVersion=&quot;1.0&quot;</td>
</tr>
<tr>
<td>link</td>
<td>Fuzzy search over general link (URL, type and note) (quote as needed)</td>
</tr>
<tr>
<td></td>
<td>link=&quot;Issue tracker&quot;</td>
</tr>
<tr>
<td>linkType</td>
<td>Exact search for type of information found at a link</td>
</tr>
<tr>
<td></td>
<td>linkType=&quot;Issue tracker&quot;</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Search behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentation</td>
<td>Fuzzy search over documentation link (URL, type and note) (quote as needed)</td>
</tr>
<tr>
<td>documentationType</td>
<td>Exact search for type of documentation</td>
</tr>
<tr>
<td>download</td>
<td>Fuzzy search over download link (URL, type, version and note) (quote as needed)</td>
</tr>
<tr>
<td>downloadType</td>
<td>Exact search for type of download</td>
</tr>
<tr>
<td>downloadVersion</td>
<td>Exact search for tool version associated with a download (must be quoted)</td>
</tr>
<tr>
<td>otherID</td>
<td>Fuzzy search over alternate tool IDs (ID value, type of ID and version)</td>
</tr>
<tr>
<td>otherIDValue</td>
<td>Exact search for value of alternate tool ID (must be quoted)</td>
</tr>
<tr>
<td>otherIDType</td>
<td>Exact search for type of alternate tool ID</td>
</tr>
<tr>
<td>otherIDVersion</td>
<td>Exact search for tool version associated with an alternate ID (must be quoted)</td>
</tr>
</tbody>
</table>

**Important:**

Values of the following parameters must be given in quotes to get sensible (or any) results:

- homepage
- version
- topicID
- operationID
- dataTypeID
- dataFormatID
- inputID
- inputDataTypeID
- inputDataFormatID
- outputID
- outputDataTypeID
- outputDataFormatID
- creditOrcidID
- publicationID
- publicationVersion
• downloadVersion
• otherIDValue
• otherIDVersion

e.g.
• https://bio.tools/api/tool?topicID="topic_3510"

**Values of other parameters can be quoted or unquoted:**

• Unquoted values invoke a fuzzy word search: it will search for fuzzy matches of words in the search phrase, to the target field
• Quoted values invoke an exact phrase search; it will search for an exact match of the full-length of the search phrase, to the target field (matches to target substrings are allowed)

e.g.
• https://bio.tools/api/tool?biotoolsID="blast" returns the tool with biotoolsID of “blast” (the “canonical” blast)
• https://bio.tools/api/tool?biotoolsID=blast returns all tools with “blast” in their biotoolsID (all blast flavours)

**Caution:** The parameters are (currently) case-sensitive, e.g. you must use &biotoolsID= and not &biotoolsid

### 7.1.3 Example

```bash
curl -X GET "https://bio.tools/api/tool/?page=1&format=json&name=signalp&sort=name&ord=asc&q=protein-signal-peptide-detection"
```

**Note:**

An EDAM concept ID can be specified as a concept URI or ID:

• Concept URI e.g. http://edamontology.org/operation_2403
• Concept ID e.g. operation_2403

**In future we may add support for:**

• Concept CURIE e.g. EDAM:operation_2403
• Numerical ID e.g. 2403

**Note:** URIs and IDs must be quoted, e.g. &topicID="operation_2403"
### 7.1.4 Response data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>The total tool count results for your query</td>
<td>2313</td>
</tr>
<tr>
<td>previous</td>
<td>Link to the previous page</td>
<td>?page=4</td>
</tr>
<tr>
<td>next</td>
<td>Link to the next page</td>
<td>?page=6</td>
</tr>
<tr>
<td>list</td>
<td>An array with multiple tools and their relative information</td>
<td>ARRAY</td>
</tr>
</tbody>
</table>

### 7.2 Tool detail

*Obtain information about a single tool.*

**HTTP GET**

```
https://bio.tools/api/tool/:id/
https://bio.tools/api/t/:id/
https://bio.tools/api/:id/
```

#### 7.2.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Yes</td>
<td>String</td>
<td>biotoolsID</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>No</td>
<td>String(json, xml, api)</td>
<td>json</td>
<td>Response media type</td>
</tr>
</tbody>
</table>

#### 7.2.2 Example

```
curl -X GET "https://bio.tools/api/tool/signalp/?format=json"
```

**Caution:** bio.tools supports upload/download of data in XML format compliant to biotoolsScheme v3.0.0. If you want to download in XML format you should use these endpoints (see Tool detail below):

```
https://bio.tools/api/tool/id/
https://bio.tools/api/t/id/
https://bio.tools/api/id/
```

e.g. https://bio.tools/api/tool/signalp

Were you to try to get XML format returned from a search over bio.tools
e.g. https://bio.tools/api/tool?toolid=signalp&format=xml

currently you’d get garbled / invalid XML (don’t use it!) - we’re looking at a fix.

### 7.3 Register a tool

*Register a new tool.*
Important: This method requires the user to be authenticated. Learn how to Log in / obtain token.

**HTTP POST**

https://bio.tools/api/tool/
https://bio.tools/api/t/

### 7.3.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Yes</td>
<td>Tool</td>
<td>Tool you wish to register. See an example tool.</td>
</tr>
</tbody>
</table>

**Note:** It is possible to specify editing permissions for tools. Learn how to manage Editing permissions.

### 7.3.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>Media type</td>
</tr>
</tbody>
</table>

### 7.3.3 Example

```
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Token 028595d6b2541e7e3df32306ecb720dadafd7" \
-d '<resource>' "https://bio.tools/api/tool/"
```

### 7.4 Validate registering a tool

Test registering a tool without it actually being saved into the database.

Important: This method requires the user to be authenticated. Learn how to Log in / obtain token.

**HTTP POST**

https://bio.tools/api/tool/validate/
https://bio.tools/api/t/validate/
7.4.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Yes</td>
<td>Tool</td>
<td>Tool you wish to validate. See an example tool.</td>
</tr>
</tbody>
</table>

7.4.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>Media type</td>
</tr>
<tr>
<td>Authorization</td>
<td>Yes</td>
<td>String('Token &lt;authorization token&gt;')</td>
<td>Authorization header. Learn how to Log in / obtain token.</td>
</tr>
</tbody>
</table>

7.4.3 Example

```
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Token 028595d682541e7e1a5dcf2306eccb720dadaf7d" \ 
-d '"resource"' "https://bio.tools/api/tool/validate/"
```

7.5 Update a tool

_Update a tool description._

---

**Important:** This method requires the user to be authenticated. Learn how to _Log in / obtain token._

**HTTP PUT**

https://bio.tools/api/tool/:id/
https://bio.tools/api/t/:id/
https://bio.tools/api/:id/

7.5.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Yes</td>
<td>String</td>
<td>biotoolsID</td>
</tr>
<tr>
<td>data</td>
<td>Yes</td>
<td>Tool</td>
<td>Description with which you wish to update the tool See an example tool.</td>
</tr>
</tbody>
</table>

---

**Note:** It is possible to specify editing permissions for tools. Learn how to manage _Editing permissions._
## 7.5.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String (application/json, application/xml)</td>
<td>Media type</td>
</tr>
<tr>
<td>Authorization</td>
<td>Yes</td>
<td>String (&quot;Token &lt;authorization token&gt;&quot;)</td>
<td>Authorization header. Learn how to Log in / obtain token</td>
</tr>
</tbody>
</table>

## 7.5.3 Example

```bash
curl -X PUT -H "Content-Type: application/json" \
-H "Authorization: Token 028595d682541e7e1a5df2306eccb720dada7d7" \
-d '"<resource>" "https://bio.tools/api/tool/SignalP"
```

## 7.6 Validate updating a tool

*Test updating a tool without it actually being saved into the database.*

**Important:** This method requires the user to be authenticated. Learn how to Log in / obtain token.

**HTTP PUT**

https://bio.tools/api/tool/:id/validate/
https://bio.tools/api/t/:id/validate/
https://bio.tools/api/:id/validate/

### 7.6.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Yes</td>
<td>String</td>
<td>biotoolsID</td>
</tr>
<tr>
<td>data</td>
<td>Yes</td>
<td>String</td>
<td>Tool Description with which you wish to update the tool for validation See an example tool.</td>
</tr>
</tbody>
</table>

### 7.6.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String (application/json, application/xml)</td>
<td>Media type</td>
</tr>
<tr>
<td>Authorization</td>
<td>Yes</td>
<td>String (&quot;Token &lt;authorization token&gt;&quot;)</td>
<td>Authorization header. Learn how to Log in / obtain token</td>
</tr>
</tbody>
</table>

88 Chapter 7. API Reference
7.6.3 Example

```bash
curl -X PUT -H "Content-Type: application/json" \ 
-H "Authorization: Token 028595d682541e7e1a5dcf2306eccb720dadaf7" \ 
-d '<resource>' "https://bio.tools/api/tool/SignalP/validate/
```

7.7 Editing permissions

*Manage editing permissions for the registered tools.*

There are currently three types of editing permissions supported by the system:

7.7.1 Private

A private tool can only be edited by the creator of the tool. This is the default option. In order to set this kind of permission, add the following info into the tool data:

```json
"editPermission": {
    "type": "private"
}
```

7.7.2 Public

Public tool can be modified by any user registered in the system. In order to set this kind of permission, add the following info into the tool data:

```json
"editPermission": {
    "type": "public"
}
```

7.7.3 Group

Specify a list of users in the system that can edit the tool. In order to set this kind of permission, add the following info into the tool data:

```json
"editPermission": {
    "type": "private",
    "authors": [
        "registered_user_1", "registered_user_2"
    ]
}
```

7.8 Delete a tool

*Removes a tool from the registry.*

Important: This method requires the user to be authenticated. Learn how to *Log in / obtain token.*
**HTTP DELETE**

https://bio.tools/api/tool/:id/
https://bio.tools/api/t/:id/
https://bio.tools/api/:id/

### 7.8.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Yes</td>
<td>String</td>
<td>biotoolsID</td>
</tr>
</tbody>
</table>

### 7.8.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Yes</td>
<td>String(&quot;Token &lt;authorization token&gt;&quot;)</td>
<td>Authorization header. Learn how to <a href="/#">Log in / obtain token</a>.</td>
</tr>
</tbody>
</table>

### 7.8.3 Example

```bash
curl -X DELETE \
-H "Authorization: Token 028595d682541e7e1a5df2306eccb720dadafd7" \
"https://bio.tools/api/tool/SignalP"
```

### 7.9 List used terms

*Obtain a list of terms registered with tools for some attributes, e.g. a list of names of all tools.*

**HTTP GET**

https://bio.tools/api/used-terms/:attribute

#### 7.9.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>Yes</td>
<td>String(name, topic, functionName, input, output, credits, all)</td>
<td></td>
<td>Attribute for which a list of used terms will be returned</td>
</tr>
<tr>
<td>format</td>
<td>No</td>
<td>String(json, xml, api)</td>
<td>json</td>
<td>Response media type</td>
</tr>
</tbody>
</table>

#### 7.9.2 Example

```bash
curl -X GET "https://bio.tools/api/used-terms/name/?format=json"
```
7.9.3 Response data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A list of used terms</td>
</tr>
</tbody>
</table>

7.10 Create a user account

Creates a user account and emails a verification email.

HTTP POST

https://bio.tools/api/rest-auth/registration/

7.10.1 POST data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Yes</td>
<td>String</td>
<td>Account username</td>
<td>username</td>
</tr>
<tr>
<td>password1</td>
<td>Yes</td>
<td>String</td>
<td>Password</td>
<td>password</td>
</tr>
<tr>
<td>password2</td>
<td>Yes</td>
<td>String</td>
<td>Repeated password</td>
<td>password</td>
</tr>
<tr>
<td>email</td>
<td>Yes</td>
<td>String</td>
<td>Account email. The verification email will be sent to this address</td>
<td><a href="mailto:example@example.org">example@example.org</a></td>
</tr>
</tbody>
</table>

7.10.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>POST data media type</td>
</tr>
</tbody>
</table>

7.10.3 Example

```
curl -X POST -H "Content-Type: application/json" \  
-d '{"username":"username", "password1":"password", \  "password2":"password", "email":"example@example.org"}' \  "https://bio.tools/api/rest-auth/registration/"
```

7.11 Verify a user account

Verifies a user account based on the emailed verification key.

HTTP POST

https://bio.tools/api/rest-auth/registration/verify-email/
7.11.1 POST data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Yes</td>
<td>String</td>
<td>Verification key from account creation email</td>
<td>ndwotbpmk5zxdxfwgu2822xynjidhizhwosycve7hrco1f156hjwdsf1f6gbn</td>
</tr>
</tbody>
</table>

7.11.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>POST data media type</td>
</tr>
</tbody>
</table>

7.11.3 Example

```bash
curl -X POST -H "Content-Type: application/json" \ 
   -d '{"key":"ndwotbpmk5zxdxfwgu2822xynjidhizhwosycve7hrco1f156hjwdsf1f6gbn"}' \ 
   "https://bio.tools/api/rest-auth/registration/verify-email/
```

7.12 Log in / obtain token

Logs the user in and returns an authentication token.

HTTP POST

https://bio.tools/api/rest-auth/login/

7.12.1 POST data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Yes</td>
<td>String</td>
<td>Account username</td>
<td>username</td>
</tr>
<tr>
<td>password</td>
<td>Yes</td>
<td>String</td>
<td>Password</td>
<td>password</td>
</tr>
</tbody>
</table>

7.12.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>POST data media type</td>
</tr>
</tbody>
</table>

7.12.3 Example

```bash
curl -X POST -H "Content-Type: application/json" \ 
   -d '{"username":"username","password":"password"}' \ 
   "https://bio.tools/api/rest-auth/login/
```
7.12.4 Response data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Authentication token</td>
</tr>
</tbody>
</table>

7.13 Get user information

Return information about the logged in user account, including a list of registered tool (name, id, version, additionDate, lastUpdate)

**Important:** This method requires the user to be authenticated. Learn how to Log in / obtain token.

*HTTP GET*

https://bio.tools/api/rest-auth/user/

7.13.1 Endpoint Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>No</td>
<td>String(json, xml, api)</td>
<td>json</td>
<td>Response media type</td>
</tr>
</tbody>
</table>

7.13.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
</table>

7.13.3 Example

```bash
curl -X GET \
-H "Authorization: Token 028595d682541e7e6a5d0f2306ecb720dadafd7" \ 
"https://bio.tools/api/rest-auth/user/?format=json"
```

7.13.4 Response data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Account username</td>
</tr>
<tr>
<td>email</td>
<td>Account email</td>
</tr>
<tr>
<td>resources</td>
<td>List of registered tools (limited to name, id, version, additionDate, lastUpdate)</td>
</tr>
</tbody>
</table>
7.14 Log out

Log out of the system.

**Important:** This method requires the user to be authenticated. Learn how to Log in / obtain token.

**HTTP POST**

https://bio.tools/api/rest-auth/logout/

7.14.1 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Yes</td>
<td>String('Token &lt;authorization token&gt;')</td>
<td>Authorization header. Learn how to Log in / obtain token.</td>
</tr>
</tbody>
</table>

7.14.2 Example

```bash
curl -X POST
-H "Authorization: Token 028595d682541e7e1a5dcf2306eccb720dadaaf7"
"https://bio.tools/api/rest-auth/logout/
```

7.15 Reset user password

Send a password reset email.

**HTTP POST**

https://bio.tools/api/rest-auth/password/reset/

7.15.1 POST data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>Yes</td>
<td>String</td>
<td>Account email</td>
<td><a href="mailto:example@example.org">example@example.org</a></td>
</tr>
</tbody>
</table>

7.15.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>POST data media type</td>
</tr>
</tbody>
</table>
7.15.3 Example

curl -X POST -H "Content-Type: application/json" \
-d '{"email":"example@example.org"}'} \
"https://bio.tools/api/rest-auth/password/reset/"

7.16 Confirm password reset

Confirm a password reset using uid and token from a password reset email.

HTTP POST

https://bio.tools/api/rest-auth/password/reset/confirm/

7.16.1 POST data

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>Yes</td>
<td>String</td>
<td>UID from password reset email</td>
<td>MQ</td>
</tr>
<tr>
<td>token</td>
<td>Yes</td>
<td>String</td>
<td>Token from password reset email</td>
<td>4ct-67e90a1ab4f22fbb9b9f</td>
</tr>
<tr>
<td>password1</td>
<td>Yes</td>
<td>String</td>
<td>New password</td>
<td>new_password</td>
</tr>
<tr>
<td>password2</td>
<td>Yes</td>
<td>String</td>
<td>New password repeated</td>
<td>new_password</td>
</tr>
</tbody>
</table>

7.16.2 Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>Yes</td>
<td>String(application/json, application/xml)</td>
<td>POST data media type</td>
</tr>
</tbody>
</table>

7.16.3 Example

curl -X POST -H "Content-Type: application/json" \
-d '{"uid":"MQ", "token":"4ct-67e90a1ab4f22fbb9b9f", \
"password1":"new_password", "password2":"new_password"}'} \
"https://bio.tools/api/rest-auth/password/reset/confirm/"

7.17 Stats

Compile stats about the registry.

HTTP GET

https://bio.tools/api/stats
7.17.1 Example

curl -X GET "https://bio.tools/api/stats"
bio.tools implements the model of software information defined in biotoolsScheme v3.0.0. This page summarises the structure and syntax of an XML / JSON file that describes a tool for submission to bio.tools via the API.

### 8.1 Payload formats

To submit a tool via the bio.tools API you’ll need to POST a tool description to the tool endpoint. The API supports XML and JSON format uploads and downloads comatible with biotoolsSchema.

#### 8.1.1 XML

See the sample XML document.

**Important:** When working in XML, please first read the biotoolsSchema docs. It is essential to stick to the element order (including nested elements) in the sample XML documents and as shown below.
8.1.2 JSON

A sample JSON document may look like this:

```json
{
    "name": "SignalP",
    "description": "Prediction of the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms.",
    "homepage": "http://cbs.dtu.dk/services/SignalP/",
    "biotoolsID": "signalp",
    "biotoolsCURIE": "biotools:signalp",
    "version": [
        "6.4.0.0",
        "1.1 - 1.4, 2.0-alpha, 2.0-beta-01 - 2.0-beta-04, 2.0.0"
    ],
    "otherID": [
        {
            "value": "RRID:SCR_015644",
            "type": "rrid",
            "version": "4.1"
        },
        {
            "value": "doi:10.1007/978-1-4939-7015-5_6",
            "type": "doi",
            "version": "4.1"
        }
    ],
    "function": [
        {
            "operation": [
                {
                    "uri": "http://edamontology.org/operation_0418",
                    "term": "Protein signal peptide detection"
                },
                {
                    "uri": "http://edamontology.org/operation_0422",
                    "term": "Protein cleavage site prediction"
                }
            ],
            "input": [
                {
                    "data": {
                        "uri": "http://edamontology.org/data_2044",
                        "term": "Sequence"
                    }
                },
                {
                    "format": [
                        {
                            "uri": "http://edamontology.org/format_1929",
                            "term": "FASTA"
                        }
                    ]
                }
            ]
        }
    ]
}
```

(continues on next page)
{
    "uri": "http://edamontology.org/format_3008",
    "term": "MAF"
}
]
"output": [
{
    "data": {
        "uri": "http://edamontology.org/data_1277",
        "term": "Protein features"
    },
    "format": [
    {
        "uri": "http://edamontology.org/format_2305",
        "term": "GFF"
    },
    { 
        "uri": "http://edamontology.org/format_3164",
        "term": "GTrack"
    }
    ]
 ],
{
    "data": {
        "uri": "http://edamontology.org/data_2955",
        "term": "Sequence report"
    },
    "format": [
    { 
        "uri": "http://edamontology.org/format_2331",
        "term": "HTML"
    }
    ]
},
"note": "Predicts the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms."
"cmd": "--someOption",
"toolType": ["Command-line tool", "Web application"
],
"topic": [
    {
        "uri": "http://edamontology.org/topic_0080",
        "term": "Sequence analysis"
    }
]
bio.tools Documentation, Release latest

),
{
  "uri": "http://edamontology.org/topic_0078",
  "term": "Proteins"
}
],
"operatingSystem": [
  "Linux",
  "Mac"
],
"language": [
  "ActionScript",
  "C"
],
"license": "Proprietary",
"collectionID": [
  "CBS",
  "mytools"
],
"maturity": "Mature",
"cost": "Free of charge (with restrictions)",
"accessibility": [
  "Open access",
],
"link": [
  {
    "url": "http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp",
    "type": "Repository",
    "note": "A comment goes here"
  },
  {
    "url": "http://www.cbs.dtu.dk/helpdesk",
    "type": "Helpdesk",
    "type": "Issue tracker",
    "note": "A comment goes here"
  }
],
"download": [
  {
    "url": "http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp",
    "type": "Source code",
    "note": "A comment goes here"
    "version": "1.4"
  },
  {
    "url": "http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp",
    "type": "Binaries",
    "note": "A comment goes here"
    "version": "1.4"
  }
],
(continues on next page)
8.1. Payload formats
8.2 Tool attributes

8.2.1 Name

Canonical software name assigned by the software developer or service provider, e.g. “needle”

Attribute name: name

Required: Yes

Cardinality: 1 only

Type: String

Restrictions:
- Min length: 1
- Max length: 100
- Pattern: \p{Zs}A-Za-z0-9+\.,\-_\:;()*

Example

```
# XML
<name>needle</name>

# JSON
"name": "needle"
```

Note:
- name may only contain space, uppercase and lowercase letters, decimal digits, plus symbol, period, comma, dash, underscore, colon, semicolon and parentheses.
- line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed.
- see the curation guidelines.

8.2.2 Description

Textual description of the software, e.g. “needle reads two input sequences and writes their optimal global sequence alignment to file. It uses the Needleman-Wunsch alignment algorithm to find the optimum alignment (including gaps) of two sequences along their entire length. The algorithm uses a dynamic programming method to ensure the alignment is optimum, by exploring all possible alignments and choosing the best.”

Attribute name: description
**Required**  Yes 

**Cardinality**  1 only 

**Type**  String 

**Restrictions**  Min length: 10  
Max length: 1000 

**Example**

```xml
<description>needle reads two input sequences and writes their optimal global sequence alignment to file. It uses the Needleman-Wunsch alignment algorithm to find the optimum alignment (including gaps) of two sequences along their entire length. The algorithm uses a dynamic programming method to ensure the alignment is optimum, by exploring all possible alignments and choosing the best.</description>
```

```json
"description": "needle reads two input sequences and writes their optimal global sequence alignment to file. It uses the Needleman-Wunsch alignment algorithm to find the optimum alignment (including gaps) of two sequences along their entire length. The algorithm uses a dynamic programming method to ensure the alignment is optimum, by exploring all possible alignments and choosing the best."
```

**Note:**
- minimum 10 and maximum 1000 characters. 
- line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed. 
- see the curation guidelines.

### 8.2.3 Homepage

*Homepage of the software, or some URL that best serves this purpose, e.g. “http://emboss.open-bio.org/rel/rel6/apps/needle.html”*

**Attribute name**  homepage 

**Required**  Yes 

**Cardinality**  1 

**Type**  URL 

**Restrictions**  Pattern: http(s)?://[^\s/$.?#].[^\s]* 

**Example**

```xml
```

```json
"homepage": "http://emboss.open-bio.org/rel/rel6/apps/needle.html"
```

**Note:**

---

8.2. Tool attributes 103
• a single valid URL is specified.
• see the curation guidelines.

8.2.4 biotoolsID

Unique ID (case insensitive) of the tool that is assigned upon registration of the software in bio.tools, normally identical to tool name, e.g. “needle”.

Attribute name: biotoolsID
Required: No
Cardinality: 0 or 1
Type: String
Restrictions: Pattern: \[_\-.0-9a-zA-Z]*
Example:

# XML
<biotoolsID>needle</biotoolsID>

# JSON
"biotoolsID": "needle"

Attention:
• a biotoolsID is set (and can only be changed) by bio.tools admin. It can be retrieved by API, but if specified in the payload to a PUT or POST request will be disregarded.

Note:
• the biotoolssID is a URL-safe and Linked-Data-safe derivative of (often identical to) the tool name. Allowed characters are uppercase and lowercase English letters (case insensitive!), decimal digits, hyphen, period, and underscore. Spaces can be preserved as underscore (“_”).
• see the curation guidelines.

8.2.5 biotoolsCURIE

bio.tools CURIE (compact URI) based on the unique bio.tools ID of the tool, e.g. “biotools:needle”

Attribute name: biotoolsCURIE
Required: No
Cardinality: 0 or 1
Type: String
Restrictions: Pattern: biotools:\[_\-.0-9a-zA-Z]*
Example: 
# XML
<biotoolsCURIE>needle</biotoolsCURIE>

# JSON
"biotoolsCURIE": "needle"

Attention:
- a biotoolsCURIE is set (and can only be changed) by bio.tools admin. It can be retrieved by API, but if specified in the payload to a PUT or POST request will be disregarded.

Note:
- the bio.tools CURIE is simply the bio.tools tool ID with the prefix “biotools:”.
- see the curation guidelines.

8.2.6 Version

Version information (typically a version number) of the software applicable to this bio.tools entry, e.g. “6.4.0.0”

Attribute name version

Required No

Cardinality 0 to many

Type String array

Restrictions Min length: 1

Max length: 100

Pattern: \[\p{Zs}A-Za-z0-9+,\-,\:,;()]*

Example

# XML
<version>6.4.0.0</version>
<version>1.1 - 1.4, 2.0-alpha, 2.0-beta-01 - 2.0-beta-04, 2.0.0</version>

# JSON
"version": [
  "6.4.0.0",
  "1.1 - 1.4, 2.0-alpha, 2.0-beta-01 - 2.0-beta-04, 2.0.0"
]

Note:
- name may only contain space, uppercase and lowercase English letters, decimal digits, plus symbol, period, comma, dash, colon, semicolon and parentheses.
- line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed.
• see the curation guidelines.

8.2.7 Other IDs

A unique identifier of the software, typically assigned by an ID-assignment authority other than bio.tools, e.g. “RRID:SCR_015644”

Attribute name: otherID

Required: No
Cardinality: 0 to many
Type: List of otherID objects

otherID object definition

• value
  – Required: Yes
  – Cardinality: 1 only
  – Type: String
  – Pattern: (doi|DOI):?10.[0-9]{4,9}[A-Za-z0-9;\)\(_/.-]+
  – Pattern: (rrid|RRID):.+ 
  – Pattern: (cpe|CPE):.+ 
  – Pattern: (biotools|BIOTOOLS):[\._-]0-9a-zA-Z]*

• type
  – Required: No
  – Cardinality: 0 or 1
  – Type: ENUM (list)
  – Allowed values (see Curators Guide)
    * doi
    * rrid
    * cpe
    * biotoolsCURIE

• version
  – Required: No
  – Cardinality: 0 or 1
  – Type: String
  – Restrictions: Min length: 1, Max length: 100
  – Pattern: [\p{Zs}A-Za-z0-9+\._-;()]*

Example
Note:

- type can normally be inferred from the value but should be specified otherwise. In the example it was not actually necessary to specify “type”.
- see the curation guidelines.

### 8.2.8 Function

*Details of a function (i.e. mode of operation) the software provides, expressed in terms from the EDAM ontology.*

**Attribute name** function

**Required** No

**Cardinality** 0 to many

**Type** List of function objects

**Function object definition**

**Content**

- **Operation**
  - Required: Yes
  - Cardinality: 1 to many
  - Type: List of EDAM objects
- **Input**
– Required: No
– Cardinality: 0 to many
– Type: List of input objects

• **Output**
  – Required: No
  – Cardinality: 0 to many
  – Type: List of output objects

• **note**
  – Required: No
  – Cardinality: 0 or 1
  – Type: String
  – Restrictions: min length: 10, max length: 1000

• **cmd**
  – Required: No
  – Cardinality: 0 or 1
  – Type: String
  – Restrictions: min length: 1, max length: 1000

Note:

• **note** and **cmd**: line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed.

• see the curation guidelines for the function group, note and command.

Example

```xml
# XML
<function>
  <operation>
    <uri>http://edamontology.org/operation_0418</uri>
    <term>Protein signal peptide detection</term>
  </operation>
  <operation>
    <uri>http://edamontology.org/operation_0422</uri>
    <term>Protein cleavage site prediction</term>
  </operation>
  <input>
    <data>
      <uri>http://edamontology.org/data_2044</uri>
      <term>Sequence</term>
    </data>
    <format>
      <uri>http://edamontology.org/format_1929</uri>
      <term>FASTA</term>
    </format>
  </input>
  <output>
```

(continues on next page)
<data>
  <uri>http://edamontology.org/data_1277</uri>
  <term>Protein features</term>
</data>
<data>
  <uri>http://edamontology.org/format_2305</uri>
  <term>GFF</term>
</data>
<data>
  <uri>http://edamontology.org/data_2955</uri>
  <term>Sequence report</term>
</data>
<data>
  <uri>http://edamontology.org/format_1929</uri>
  <term>FASTA</term>
</data>

<note>Predicts the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms.</note>
<cmd>-s best</cmd>
</function>

# JSON
"function": [
  {
    "operation": [
      {
        "uri": "http://edamontology.org/operation_0418",
        "term": "Protein signal peptide detection"
      },
      {
        "uri": "http://edamontology.org/operation_0422",
        "term": "Protein cleavage site prediction"
      }
    ],
    "input": [
      {
        "data": {
          "uri": "http://edamontology.org/data_2044",
          "term": "Sequence"
        },
        "format": [
          {
            "uri": "http://edamontology.org/format_1929",
            "term": "FASTA"
          }
        ]
      }
    ],
    "output": [
      (continues on next page)
{  
  "data":  
  {  
    "uri": "http://edamontology.org/data_1277",  
    "term": "Protein features"  
  },  
  "format":  
  [  
    {  
      "uri": "http://edamontology.org/format_2305",  
      "term": "GFF"  
    }  
  ]  
},  
{  
  "data":  
  {  
    "uri": "http://edamontology.org/data_2955",  
    "term": "Sequence report"  
  },  
  "format":  
  [  
    {  
      "uri": "http://edamontology.org/format_1929",  
      "term": "FASTA"  
    }  
  ]  
}  
]  
"note": "Predicts the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms.",  
"cmd": "-s best",  
}  
}

**Operation**

*The basic operation(s) performed by this software function (EDAM Operation), e.g. "Protein signal peptide detection" (http://edamontology.org/operation_0418)*

**Attribute name**  operation

**Required**  Yes

**Cardinality**  1 to many

**Child of**  Function

**Type**  List of EDAM objects

**EDAM object definition**

**Content**

- **uri**
  - Required: No (if term present), Yes (otherwise)
  - Cardinality: 0 or 1
- Type: URL
  - **term**
    - Required: No (if URI present), Yes (otherwise)
    - Cardinality: 0 or 1
    - Type: String

**Note:**
- an EDAM ontology Operation concept URL and / or term are specified, e.g. “Multiple sequence alignment”, http://edamontology.org/operation_0492.
- URI and term are validated against EDAM ontology; if term and URI do not match, an error will be returned.
- synonyms of terms (as defined in EDAM) are accepted
- see the curation guidelines.

**Example**

```xml
<operation>
  <uri>http://edamontology.org/operation_0418</uri>
  <term>Protein signal peptide detection</term>
</operation>
<operation>
  <uri>http://edamontology.org/operation_0422</uri>
  <term>Protein cleavage site prediction</term>
</operation>
```

```json
"operation": [
  {
    "uri": "http://edamontology.org/operation_0418",
    "term": "Protein signal peptide detection"
  },
  {
    "uri": "http://edamontology.org/operation_0422",
    "term": "Protein cleavage site prediction"
  }
]
```

**Input**

*Primary input data (if any)*

**Attribute name** input

**Required** No

**Cardinality** 0 to many

**Child of** Function

**Type** List of input objects

**Input object definition**

### 8.2. Tool attributes
Content

- **data**
  - Required: Yes
  - Cardinality: 1 only
  - Type: EDAM object

- **format**
  - Required: No
  - Cardinality: 0 to many
  - Type: List of EDAM objects

Example

```xml
# XML
<data>
  <uri>http://edamontology.org/data_2044</uri>
  <term>Sequence</term>
</data>

<format>
  <uri>http://edamontology.org/format_1929</uri>
  <term>FASTA</term>
</format>
```

```
# JSON
"input": [
  {
    "data": {
      "uri": "http://edamontology.org/data_2044",
      "term": "Sequence"
    },
    "format": [
      {
        "uri": "http://edamontology.org/format_1929",
        "term": "FASTA"
      }
    ]
  }
]
```

Output

*Primary output data (if any)*

**Attribute name**  output

**Required**  No

**Cardinality**  0 to many

**Child of**  Function

**Type**  List of output objects
Output object definition

Content

• data
  – Required: Yes
  – Cardinality: 1 only
  – Type: EDAM object

• format
  – Required: No
  – Cardinality: 0 to many
  – Type: List of EDAM objects

Example

```xml
<output>
  <data>
    <uri>http://edamontology.org/data_2044</uri>
    <term>Sequence</term>
  </data>
  <format>
    <uri>http://edamontology.org/format_1929</uri>
    <term>FASTA</term>
  </format>
</output>
```

```json
"output": [
  {
    "data": {
      "uri": "http://edamontology.org/data_2044",
      "term": "Sequence"
    },
    "format": [
      {
        "uri": "http://edamontology.org/format_1929",
        "term": "FASTA"
      }
    ]
  }
]
```

Data

EDAM Data concept, e.g. “Sequence” (http://edamontology.org/data_2044)” Attribute name
data

Required  Yes
Cardinality  1 only
Child of *Input* or *Output*

Type  EDAM object

**EDAM object definition**

**Content**

- **uri**
  - Required: No (if term present), Yes (otherwise)
  - Cardinality: 0 or 1
  - Type: URL
- **term**
  - Required: No (if URI present), Yes (otherwise)
  - Cardinality: 0 or 1
  - Type: String

**Note:**

- an EDAM ontology Data concept URL and / or term are specified, *e.g.* “Protein sequences”, http://edamontology.org/data_2976.
- URI and term are validated against EDAM ontology; if term and URI do not match, an error will be returned.
- synonyms of terms (as defined in EDAM) are accepted, however, the synonym will be replaced with main term.
- see the curation guidelines.

**Example**

```xml
<data>
  <uri>http://edamontology.org/data_2044</uri>
  <term>Sequence</term>
</data>
```

```json
"data": {
  "uri": "http://edamontology.org/data_2044",
  "term": "Sequence"
}
```

**Format**

*EDAM Format concept, e.g. ‘FASTA’ (http://edamontology.org/format_1929)”*

**Attribute name**  format

**Required**  No

**Cardinality**  0 to many

Child of  *Input* or *Output*
**Type**  List of EDAM objects

**EDAM object definition**

**Content**

- **uri**
  - Required: No (if term present), Yes (otherwise)
  - Cardinality: 0 or 1
  - Type: URL
- **term**
  - Required: No (if URI present), Yes (otherwise)
  - Cardinality: 0 or 1
  - Type: String

**Note:**

- an EDAM ontology Format concept URL and / or term are specified, e.g. “FASTA”, http://edamontology.org/format_1929.
- URI and term are validated against EDAM ontology; if term and URI do not match, an error will be returned.
- synonyms of terms (as defined in EDAM) are accepted, however, the synonym will be replaced with main term.
- see the curation guidelines.

**Example**

```xml
# XML
<format>
  <uri>http://edamontology.org/format_1929</uri>
  <term>FASTA</term>
</format>
```

```json
# JSON
"format":
[
  {
    "uri": "http://edamontology.org/format_1929",
    "term": "FASTA"
  }
]
```

**8.2.9 Tool type**

The type of application software: a discrete software entity can have more than one type, e.g. “Command-line tool, Web application”

**Attribute name** toolType

**Required** No

**Cardinality** 0 to many
Type  ENUM (list)

Allowed values (see Curators Guide)

- Bioinformatics portal
- Command-line tool
- Database portal
- Desktop application
- Library
- Ontology
- Plug-in
- Script
- SPARQL endpoint
- Suite
- Web application
- Web API
- Web service
- Workbench
- Workflow

Example

```
# XML
<toolType>Command-line tool</toolType>
<toolType>Web application</toolType>

# JSON
"toolType": [
  "Command-line tool",
  "Web application"
]
```

Note:

- see the curation guidelines.

8.2.10 Topic

General scientific domain the software serves or other general category (EDAM Topic), e.g. “Protein sites, features and motifs” (http://edamontology.org/topic_3510)

Attribute name  topic

Required  No

Cardinality  0 to many

Type  List of EDAM objects
EDAM object definition

Content

• uri
  – Required: No (if term present), Yes (otherwise)
  – Cardinality: 0 or 1
  – Type: URL

• term
  – Required: No (if URI present), Yes (otherwise)
  – Cardinality: 0 or 1
  – Type: String

Example

```xml
<topic>
  <uri>http://edamontology.org/topic_0605</uri>
  <term>Informatics</term>
</topic>
<topic>
  <uri>http://edamontology.org/topic_3303</uri>
  <term>Medicine</term>
</topic>
```

```json
"topic": [
  {
    "uri": "http://edamontology.org/topic_0605",
    "term": "Informatics"
  },
  {
    "uri": "http://edamontology.org/topic_3303",
    "term": "Medicine"
  }
]
```

Note:

• an EDAM ontology Topic concept URL and / or term are specified, e.g. “Proteomics”, http://edamontology.org/topic_0121.

• URI and term are validated against EDAM ontology; if term and URI do not match, an error will be returned.

• synonyms of terms (as defined in EDAM) are accepted, however, the synonym will be replaced with main term.

• see the curation guidelines.

8.2.11 Operating system

The operating system supported by a downloadable software package, e.g. “Linux”
Attribute name  operatingSystem

Required  No
Cardinality  0 to many
Type  ENUM (list)

Allowed values (see Curators Guide)

- Linux
- Windows
- Mac

Example

```xml
<operatingSystem>Linux</operatingSystem>
<operatingSystem>Mac</operatingSystem>
```

```json
"operatingSystem":
[  
  "Linux",
  "Mac"
]
```

Note:

- see the curation guidelines.

### 8.2.12 Programming language

Name of programming language the software source code was written in, e.g. “C”

Attribute name  language

Required  No
Cardinality  0 to many
Type  ENUM (list)


Example

```xml
<language>Python</language>
<language>C</language>
```

```json
"language":
[  
  "Python",
  "C"
]
```
"language": {
  "Python",
  "C"
}

Note:

- see the curation guidelines.

8.2.13 License

Software or data usage license, e.g. “GPL-3.0”

Attribute name  license

Required  No

Cardinality  0 or 1

Type  ENUM

Allowed values (see Curators Guide)  OBSD, AAL, ADSL, AFL-1.1, AFL-1.2, AFL-2.0, AFL-2.1, AFL-3.0, AGPL-1.0, AGPL-3.0, AMPLPA, AML, AMFAS, ANTLR-PD, APAFML, APL-1.0, APSL-1.0, APSL-1.1, APSL-1.2, APSL-2.0, Abstyles, Adobe-2006, Adobe-Glyph, Afmparse, Aladdin, Apache-1.0, Apache-1.1, Apache-2.0, Artistic-1.0, Artistic-1.0-Perl, Artistic-1.0-cl8, Artistic-2.0, BSD-2-Clause, BSD-2-Clause-FreeBSD, BSD-2-Clause-NetBSD, BSD-3-Clause, BSD-3-Clause-Attribution, BSD-3-Clause-Clear, BSD-3-Clause-LBNL, BSD-3-Clause-No-Nuclear-License, BSD-3-Clause-No-Nuclear-License-2014, BSD-3-Clause-No-Nuclear-Warranty, BSD-4-Clause, BSD-4-Clause-UC, BSD-Protection, BSD-Source-Code, BSL-1.0, Bahyph, Barr, Beerware, BitTorrent-1.0, BitTorrent-1.1, Borceux, CATOSL-1.1, CC-BY-1.0, CC-BY-2.0, CC-BY-2.5, CC-BY-3.0, CC-BY-4.0, CC-BY-NC-1.0, CC-BY-NC-2.0, CC-BY-NC-2.5, CC-BY-NC-3.0, CC-BY-NC-4.0, CC-BY-NC-ND-1.0, CC-BY-NC-ND-2.0, CC-BY-NC-ND-2.5, CC-BY-NC-ND-3.0, CC-BY-NC-ND-4.0, CC-BY-NC-SA-1.0, CC-BY-NC-SA-2.0, CC-BY-NC-SA-2.5, CC-BY-NC-SA-3.0, CC-BY-NC-SA-4.0, CC-BY-ND-1.0, CC-BY-ND-2.0, CC-BY-ND-2.5, CC-BY-ND-3.0, CC-BY-ND-4.0, CC-BY-SA-1.0, CC-BY-SA-2.0, CC-BY-SA-2.5, CC-BY-SA-3.0, CC-BY-SA-4.0, CC0-1.0, CDDL-1.0, CDDL-1.1, CECILL-1.0, CECILL-1.1, CECILL-2.0, CECILL-B, CECILL-C, CNRI-Jython, CNRI-Python, CNRI-Python-GPL-Compatible, CPAL-1.0, CPL-1.0, CPOL-1.02, CU-0PL-1.0, Caldera, CLArtistic, Condor-1.1, Crossword, CrystalStacker, Cube, D-FSL-1.0, DOC, DDS, Dotseqn, ECL-1.0, ECL-2.0, EFL-1.0, EFL-2.0, EPL-1.0, EUdataGrid, EUPL-1.0, EUPPL-1.0, Entessa, ErlPL-1.1, Eurosymb, FSFAP, FSFUL, FSFURL, FTL, Fair, Framework-1.0, FreeImage, GFDSL-1.1, GFDSL-1.2, GFDSL-1.3, GL2PS, GPL-1.0, GPL-2.0, GPL-3.0, Giftware, Glide, Gluixe, HPND, HaskellReport, IBM-pibs, ICU, IJG, IPA, IPL-1.0, ISC, ImageMagick, Imlib3, Info-ZIP, Intel, Intel-ACPI, Interbase-1.0, JSON, JasPer-2.0, LAL-1.2, LAL-1.3, LGPL-2.0, LGPL-2.1, LGPL-3.0, LGPLLRR, LPL-1.0, LPL-1.02, LPPL-1.0, LPPL-1.1, LPPL-1.2, LPPL-1.3a, LPPL-1.3c, Latex2e, Leptonica, LiLiQ-P-1.1, LiLiQ-R-1.1, LiLiQ-Rplus-1.1, Libpng, MIT, MIT-CMU, MIT-advertising, MIT-enna, MIT-feh, MITNFA, MPL-1.0, MPL-1.1, MPL-2.0, MPL-2.0-no-copyright-exception, MS-PL, MS-RL, MTLL, MakeIndex, MirOS, Motosoto, Multics, Mup, NASA-1.3, NBPL-1.0, NCSA, NGPL, NLOD-1.0, NLPL, NSWL, NPL-1.0, NPL-1.1, NPOSL-3.0, NRL, NTP, Naumen, NetCDF, Newsltr, Noki, Noweb, Nunit, OCCT-PL, OCLC-2.0, ODbL-1.0, OFL-1.0, OFL-1.1, OGTL, OLDB-1.1,
Example

```xml
<license>Proprietary</license>
```

```json
"license": "Proprietary"
```

Note:
- see the curation guidelines.

### 8.2.14 Collection

*Unique ID of a collection that the software has been assigned to within bio.tools, e.g. “CBS*

**Attribute name** collectionID

**Required** No

**Cardinality** 0 to many

**Type** List of strings

**Restrictions** Min length: 1

- Max length: 100
- Pattern: `[\p{Zs}A-Za-z0-9\-\_\.;\(\)]*`

Example

```xml
<collectionID>CBS</collectionID>
<collectionID>NorduGrid</collectionID>
```

```json
"collectionID": [
  "CBS",
  "NorduGrid"
]
```
8.2.15 Maturity

*How mature the software product is, e.g. “Mature”*

**Attribute name** maturity

**Required** No

**Cardinality** 0 or 1

**Type** ENUM

**Allowed values** (see Curators Guide)

- Emerging
- Mature
- Legacy

**Example**

```xml
<maturity>Mature</maturity>
```

```json
"maturity": "Mature"
```

**Note:**
- see the curation guidelines.

8.2.16 Cost

*Monetary cost of acquiring the software, e.g. “Free of charge (with retritions)”*

**Attribute name** cost

**Required** No

**Cardinality** 0 or 1

**Type** ENUM

**Allowed values** (see Curators Guide)

- Free of charge
- Free of charge (with restrictions)
• Commercial

Example

```xml
<cost>Free of charge (with restrictions)</cost>
```

```json
"cost": "Free of charge (with restrictions)"
```

Note:
• see the curation guidelines.

### 8.2.17 Accessibility

*Whether the software is freely available for use, e.g. “Open access”*

**Attribute name** accessibility

**Required** No

**Cardinality** 0 or 1

**Type** ENUM (list)

**Allowed values (see Curators Guide)**
  • Open access
  • Restricted access

Example

```xml
<accessibility>Open access</accessibility>
```

```json
"accessibility": [
  "Open access",
]
```

Note:
• see the curation guidelines.

### 8.2.18 ELIXIR platform

*ELIXIR platform credited for developing or providing the software.*

**Attribute name** elixirPlatform

**Required** No

**Cardinality** 0 to many
**Type**  ENUM (list)

Allowed values (see Curators Guide)
- Data
- Tools
- Compute
- Interoperability
- Training

**Example**

```xml
<elixirPlatform>Open access</elixirPlatform>
<elixirPlatform>Freeware</elixirPlatform>
```

```json
"elixirPlatform": ["Tools", "Compute"
```

**Note:**
- see the curation guidelines.

## 8.2.19 ELIXIR node

**ELIXIR node credited for developing or providing the software - the software is in Node Service Delivery Plan.**

**Attribute name**  elixirNode

**Required**  No

**Cardinality**  0 to many

**Type**  ENUM (list)

Allowed values (see Curators Guide)
- Belgium
- Czech Republic
- Denmark
- EMBL
- Estonia
- Finland
- France
- Germany
- Greece
- Hungary
• Ireland
• Israel
• Italy
• Luxembourg
• Netherlands
• Norway
• Portugal
• Slovenia
• Spain
• Sweden
• Switzerland
• UK

Example

```
# XML
<elixirNode>Denmark</elixirNode>
<elixirNode>France</elixirNode>

# JSON
"elixirNode":
[
  "Denmark",
  "France"
]
```

Note:

• see the curation guidelines.

8.2.20 Link

Miscellaneous links for the software e.g. repository, issue tracker or mailing list.

Attribute name  link

Required  No

Cardinality  0 to many

Type  List of link objects

Link object definition

  Content
   • url
      – Required: Yes
      – Cardinality: 1 only
      – Type: URL
- **Pattern:** `http(s)?://[\s/\$/?#]\.[^\s]*

* type
  - **Required:** Yes
  - **Cardinality:** 1 or more
  - **Type:** ENUM
  - **Allowed values:** (see Curators Guide)
    - Discussion forum
    - Galaxy service
    - Helpdesk
    - Issue tracker
    - Mailing list
    - Mirror
    - Software catalogue
    - Repository
    - Service
    - Social media
    - Technical monitoring
    - Other

* note
  - **Required:** No
  - **Cardinality:** 0 or 1
  - **Type:** String
  - **Restrictions:** min length: 10, max length: 1000

**Example**

```xml
<link>
  <url>http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp</url>
  <type>Helpdesk</type>
  <type>Issue tracker</type>
  <note>A comment goes here.</note>
</link>
```

```json
"link": [
  {
    "url": "http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp",
    "type": "Helpdesk",
    "type": "Issue tracker",
    "note": "A comment goes here."
  }
]
```

8.2. Tool attributes  125
8.2.21 Download

Links to downloads for the software, e.g. source code, virtual machine image or container.

Attribute name  download
Required  No
Cardinality  0 to many
Type  List of download objects

Download object definition

Content

- url
  - Required: Yes
  - Cardinality: 1 only
  - Type: URL
  - Pattern: http(s)?://[^\s/$.?#].[^\s\]]*

- type
  - Required: Yes
  - Cardinality: 1 only
  - Type: ENUM
  - Allowed values: (see Curators Guide)
    * API specification
    * Biological data
    * Binaries
    * Command-line specification
    * Container file
    * Icon
    * Screenshot
    * Source code
    * Software package
    * Test data
    * Test script
    * Tool wrapper (CWL)
* Tool wrapper (galaxy)
* Tool wrapper (taverna)
* Tool wrapper (other)
* VM image
* Downloads page
* Other

* note
  - Required: No
  - Cardinality: 0 or 1
  - Type: String
  - Restrictions: min length: 10, max length: 1000

* version
  - Required: No
  - Cardinality: 0 or 1
  - Type: String
  - Restrictions: Min length: 1, Max length: 100
  - Pattern: [\p{Zs}A-Za-z0-9+\-,\_,\_;\{\}]*

Example

```xml
<download>
  <url>http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp</url>
  <type>Source code</type>
  <note>Complete distribution</note>
  <version>1.4</version>
</download>
```

```json
{  
  "download": [
    {  
      "url": "http://www.cbs.dtu.dk/cgi-bin/sw_request?signalp",
      "type": "Source code",
      "note": "Complete distribution",
      "version": "1.4"
    }
  ]
}
```

Note:

- the note is minimum 10 and maximum 1000 characters. Line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed.
- see the curation guidelines.

---

8.2. Tool attributes
8.2.22 Documentation

Links to documentation about the software e.g. manual, API specification or training material.

Attribute name documentation

Required No

Cardinality 0 to many

Type List of documentation objects

Documentation object definition

Content

• url
  
  – Required: Yes
  
  – Cardinality: 1 only
  
  – Type: URL
  
  – Pattern: http(s)://[^\s/$.?#].[^\s]*

• type
  
  – Required: Yes
  
  – Cardinality: 1 or more
  
  – Type: ENUM
  
  – Allowed values: (see Curators Guide)

  * API documentation
  * Citation instructions
  * Code of conduct
  * Command-line options
  * Contributions policy
  * FAQ
  * General
  * Governance
  * Installation instructions
  * User manual
  * Terms of use
  * Release notes
  * Training material
  * Other

• note
  
  – Required: No
  
  – Cardinality: 0 or 1
  
  – Type: String
Restrictions: min length: 10, max length: 1000

Example

```xml
<documentation>
  <url>http://www.cbs.dtu.dk/services/SignalP</url>
  <type>General</type>
  <type>Code of conduct</type>
  <note>Comprehensive usage instructions.</note>
</documentation>
```

```json
"documentation": [
  {
    "url": "http://www.cbs.dtu.dk/services/SignalP",
    "type": "General",
    "type": "Code of conduct",
    "note": "Comprehensive usage instructions"
  }
]
```

Note:
- the note is minimum 10 and maximum 1000 characters. Line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed.
- see the curation guidelines.

### 8.2.23 Relation

Details of a relationship this software shares with other software registered in bio.tools.

**Attribute name** relation

**Required** No

**Cardinality** 0 to many

**Type** List of relation objects

**Relation object definition**

**Content**

- **biotoolsID**
  - Required: Yes
  - Cardinality: 1 only
  - Type: String
  - Pattern: \[\_.0-9a-zA-Z\]*

- **type**
  - Required: Yes
  - Cardinality: 1 only
- Type: ENUM
- Allowed values: (see Curators Guide)
  * isNewVersionOf
  * hasNewVersion
  * uses
  * usedBy
  * includes
  * includedIn

Example

```xml
# XML
<relation>
  <biotoolsID>needle</biotoolsID>
  <type>isNewVersionOf</type>
</relation>
```

```json
# JSON
"relation": [
  {
    "biotoolsID": "needle",
    "type": "isNewVersionOf"
  },
  {
    "biotoolsID": "emboss",
    "type": "includedIn"
  }
]
```

Note:
- see the curation guidelines.

8.2.24 Publication

Publications about the software

Attribute name  publication

Required  Yes

Cardinality  0 to many

Type  List of publication objects

Publication object definition

Content
- pmcid
  - Required: One of doi, pmid or pmcid must be specified.
  - Cardinality: 0 or 1
- Type: PMCID
  - Pattern: (PMC)[1-9][0-9]{0,8}

- pmid
  - Required: One of doi, pmid or pmcid must be specified.
  - Cardinality: 0 or 1
  - Type: PMID
  - Pattern: [1-9][0-9]{0,8}

- doi
  - Required: One of doi, pmid or pmcid must be specified.
  - Cardinality: 0 or 1
  - Type: DOI
  - Pattern: 10.[0-9]{4,9}[A-Za-z0-9:;\(_/.-]+

- type
  - Required: No
  - Cardinality: 0 to many
  - Type: ENUM
  - Allowed values: (see Curators Guide) - Primary - Method - Usage - Benchmarking study - Review - Other

- note
  - Required: No
  - Cardinality: 0 or 1
  - Type: String
  - Restrictions: min length: 10, max length: 1000

- version
  - Required: No
  - Cardinality: 0 or 1
  - Type: String
  - Restrictions: Min length: 1, Max length: 100
  - Pattern: [\p{Zs}A-Za-z0-9+\.\-_\[:;\]*/]*

Example

```xml
# XML
<publication>
  <pmcid>21959131</pmcid>
  <pmid>21959131</pmid>
  <doi>10.1038/nmeth.1701</doi>
  <type>Primary</type>
  <type>Method</type>
  <note>A comment goes here</note>
  <version>4.0</version>
</publication>
```

(continues on next page)
8.2.25 Credit

*Individuals or organisations that should be credited, or may be contacted about the software.*

**Attribute name** credit  
**Required** No  
**Cardinality** 0 to many  
**Type** List of credit objects

**Credit object definition**

**Content**

- **name**
  - Required: Yes  
  - Cardinality: 0 or 1  
  - Type: String  
  - Restrictions: min length: 1, max length: 100

- **orcidid**
  - Required: No  
  - Cardinality: 0 or 1  
  - Type: String  
  - Restrictions: pattern: http://orcid.org/[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}  
  - Restrictions: pattern: https://orcid.org/[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}

- **gridid**
  - Required: No
- Cardinality: 0 or 1
- Type: String
- Restrictions: pattern: grid.[0-9]{4,}[a-f0-9]{1,2}

• email
  - Required: No
  - Cardinality: 0 or 1
  - Type: Email
  - Restrictions: pattern: [A-Za-z0-9_.]+([-+.']?[A-Za-z0-9_.]+)*@[A-Za-z0-9_.]+([-+.']?[A-Za-z0-9_.]+)*

• url
  - Required: No
  - Cardinality: 0 or 1
  - Type: URL
  - Restrictions: pattern: http(s?)://[^s/$.?#]*[^s]*

• typeEntity
  - Required: No
  - Cardinality: 0 or 1
  - Type: ENUM
  - Allowed values: (see Curators Guide)
    * Person
    * Project
    * Division
    * Institute
    * Consortium
    * Funding agency

• typeRole
  - Required: No
  - Cardinality: 0 to many
  - Type: ENUM (list)
  - Allowed values: (see Curators Guide)
    * Developer
    * Maintainer
    * Provider
    * Documentor
    * Contributor
    * Support
* Primary contact

* note
  - Required: No
  - Cardinality: 0 or 1
  - Type: String
  - Restrictions: min length: 10, max length: 1000

Example

```xml
<credit>
  <name>TN Petersen</name>
  <orcidid>http://orcid.org/0000-0002-1825-0097</orcidid>
  <gridid>grid.5170.3</gridid>
  <email>test@cbs.dtu.dk</email>
  <url>http://cbs.dtu.dk</url>
  <typeEntity>Person</typeEntity>
  <typeRole>Developer</typeRole>
  <typeRole>Documentor</typeRole>
  <note>Lead developer</note>
</credit>
```

```json
"credit": [
  {
    "name": "TN Petersen",
    "orcidid": "http://orcid.org/0000-0002-1825-0097",
    "gridid": "grid.5170.3",
    "url": "http://cbs.dtu.dk",
    "email": "test@cbs.dtu.dk",
    "typeEntity": "Person",
    "typeRole": [
      "Developer",
      "Documentor"
    ],
    "note": "Lead developer"
  }
]
```

Example

```xml
<credit>
  <elixirPlatform>Tools</elixirPlatform>
</credit>
```

```json
"credit": [
  {
    "elixirPlatform": "Norway"
  }
]
```
Note:
- one of <name>, <email> or <url> must be specified.
- the credit name may only contain space, uppercase and lowercase letters, decimal digits, plus symbol, period, comma, dash, underscore, colon, semicolon and parentheses.
- line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces are not allowed / will be removed.
- see the curation guidelines.

8.3 Entry management attributes

8.3.1 Permissions

Attribute name editPermission

Required No
Cardinality todo
Type Permission object

Permission object definition

Content

- type
  - Required: Yes
  - Cardinality: todo
  - Type: ENUM
  - Allowed values: -private-public-group
- authors
  - Required: No
  - Cardinality: todo
  - Type: List of usernames

Notes ‘authors’ only need to be provided when type is set to group.

Example

# XML

# JSON

"editPermission": {
  "type": "group",
  "authors": [
    "ekry",
    "lukbe"
  ]
}
CHAPTER 9

Hangouts

Regular informal meetings to discuss all matters around bio.tools including ELIXIR EXCELERATE WP1 ("Tools Interoperability and Service Registry") tasks, activities of ELIXIR Denmark technical staff and partners.

You are welcome to suggest or attend a call; please mail Jon Ison (jison@bioinformatics.dtu.dk), including your gmail and skype addresses. To understand how we organise tasks and projects, read the Contributors Guide.
All developments of *bio.tools* software and content are informed by:

1. Community requests including partners and end-users.
2. Delivering priorities of the ELIXIR EXCELERATE grant (granted in April 2015) including revisions in light of 2017 midterm review.
3. Priorities of the ELIXIR Danish node.
4. Personal priorities of the *bio.tools* team, having insight of the core requirements.
5. Events on the ground.

For a summary of active issues see GitHub. We aim for quarterly major new releases of the production deployment (https://bio.tools) with new changes more regularly pushed to the development deployment (https://dev.bio.tools).

Tasks are assigned to quarterly milestones in GitHub. See in particular Critical and High priority issues and issues which are in progress.

Please join the discussion in GitHub. The *bio.tools* core team is very small, so bug fixes, new features etc. take a while - you’re patience is appreciated!

For higher-level project management tasks, see https://biotools.sifterapp.com/ (for a sifter account mail Jon Ison).
ELIXIR Denmark - the coordinating node of the bio.tools project - earmars funds (as available) to support studentships to work on curation-focussed mini-projects for bio.tools. Projects must have clear and quantifiable impact on bio.tools content, in terms of number of entries and/or content quality. Projects can include developments of some tooling, so long as this contributes directly to the project goals.

If you would like to propose a project, then please discuss your ideas first by mailing Jon Ison cc Peter Longreen. If following this discussion, we all agree there is basis for a project, then we’d require a 1-page project proposal, the text of which we can work on together and in collaboration with other members of the registry-core group. Funding will be prioritised (by the Danish Node management) by proposals having the biggest potential impact on bio.tools content and quality.

We anticipate most projects to be short duration (normally the equivalent of a month full time work) however there is flexibility, especially where we find talented students who can clearly demonstrate that there work has made an impact. In case of project continuation, progress would be reviewed, and funding for projects that did not perform would be terminated.

11.1 Requirements

• each proposal requires (at least) two named mentors:
  – someone to vouch for the student, provide local on-site supervision, and handle payment of the student
  – someone (normally from the the registry-dev group) who will assist with supervision and oversee the delivery of the work

• students must be enrolled with an accedited University, or have accepted a place at such

• any tooling developed during the studentship would have to made freely available under open license

11.2 Answers to FAQ

• you are welcome to apply at any time
• there is no limit to the number of proposals, although a student can only be employed on one project at one time
• you cannot participate both as a mentor and a student
• only an individual may work on a project; groups cannot submit proposals
• when writing a proposal, please refer to the existing proposals below and follow the general structure and style
• projects must have clear and quantifiable impact on *bio.tools* content, but you are free to propose anything to these ends: you will need to inspect [https://bio.tools](https://bio.tools) and [https://dev.bio.tools](https://dev.bio.tools) (latest dev server) to assess current status
• for further information, mail Jon Ison cc Peter Longreen.
• we particularly welcome proposals from thematic editors.

### 11.3 Proposals

Finalised proposals are uploaded to [https://github.com/bio-tools/Studentships/](https://github.com/bio-tools/Studentships/).

---

**Mining the Scientific Literature for and Annotating Proteomics Software using the EDAM ontology and biotools-XSD**

**STATUS:** Complete. See Proposal. Open for comments.

---

**Harvesting service descriptions for bio.tools using OpenAPI standards**

**STATUS:** Complete. See Proposal.

See update on progress and pre-publication.

---

**Annotating software tools in a scientific context**

**STATUS:** Complete (5 students). See Proposal.

---

**Annotating software tools in domains of the Life Sciences**

**STATUS:** Funded and ongoing (metabolomics). Open for comments.
GitHub projects

There are now many subprojects concerning *bio.tools* and *EDAM*, including documentation, information scheman, adapters for file format conversion, shims for content import and export, utilities for text mining, and more.

The projects are nearly all hosted under the bio.tools and EDAM GitHub organisations:

- https://github.com/bio-tools
- https://github.com/edamontology/
We have organised many events and regularly attend events organised by others. If you want to attend an event or have an idea for an event, please mail registry@elixir-dk.org. As a rule we try to avoid events in July & August. All attendees should please read our code of conduct.

- **Curation Hackathons** (“curatathons”) gather providers from across the board to curate their resources, critique the Registry interfaces, and provide a forum for knowledge exchange and collaboration.

- **Thematic Hackathons** engage experts in a specific scientific area to help improve the relevant branches of EDAM, consolidate the existing registry annotations, as well as register new resources within the theme.

- **Resource Hackathons** collaborate with experts from a specific collection of tools and services, typically some other registry, community project or Web portal, to bring the collection up to the ELIXIR annotation standard and expose it in the Registry.

- **Technical Hackathons** focus on ontology, software or other technical developments in support of curation of the Registry, its technical development, applications and integration with other systems.

### 13.1 Forthcoming events

**Technical Hackathon** : Towards a comprehensive catalogue of data formats (Autumn 2017 tbd, Amsterdam, NL) **tentative**

A hackathon aimed at providing comprehensive coverage of data formats in EDAM. More details will be added soon.

### 13.2 Past events

**Meeting: ELIXIR Tools Platform** (Jan 28 - 30, Ghent, BE)

A more sustainable content management architecture, based on GitHub, was outlined in more detail. You can track developments on GitHub.

**Hackathon: Paris BioHackathon** (Nov 12-16, Campus des berges de Seine, FR)
The *bio.tools* and EDAM developers ran drop-in sessions each day of this 5-day hackathon, ranging from *bio.tools* testing and feature prioritisation, curation methods and tooling, through to prototyping a more sustainable content management architecture based on GitHub.

https://2018.biohackathon-europe.org/

**Meeting: ELIXIR Tools and ELIXIR Compute Platforms : coordination (Sep 19, Schipol, UK)**

*bio.tools* was presented in context of a coordinated effort to establish ELIXIR-wide standards, protocols and processes for the orchestration of containerised applications and workloads provided by ELIXIR Communities.

**Conference: ECCB 2018 - 17TH European Conference on Computational Biology (Sep 8-12 2018, Athens, GR)**

http://eccb18.org/

Work on exploring the application of *bio.tools* data to automated workflow composition in mass spectrometry-based proteomics was presented. The work was recently published in Bioinformatics (10.1093/bioinformatics/bty646).

**Conference: ELIXIR-DK @ 4th Annual Danish Bioinformatics Conference (Aug 29-30 2018, Odense, DK)**

http://elixir-node.cbs.dtu.dk/?page_id=2369

This will showcase the work of the Danish ELIXIR node in context of the Danish and European bioinformatics community.

**Conference: BOSC 2018 (Jun 25-30 2018, Portland, USA)**

https://gccbosc2018.sched.com/list/descriptions/

The 2018 Galaxy Community Conference (GCC2018) and Bioinformatics Open Source Conference 2018 (BOSC2018) are meeting together in Portland, Oregon, United States, June 25-30, 2018. There will be two days of training, a two+ day meeting, and four days of intense collaboration. The meeting features joint & parallel sessions, shared keynotes, poster & demo sessions, birds-of-a-feather, and social events. EDAM, and the tooling around *bio.tools* for integration with Galaxy, will be presented.

**Conference: ELIXIR All-hands (Jun 4-7 2018, Berlin, DE)**

https://www.elixir-europe.org/events/elixir-all-hands-2018

The fourth ELIXIR All Hands meeting, bringing together members of the ELIXIR community from across the ELIXIR Nodes, and collaborators from partner organisations, in order to review ELIXIR achievements and activities so far and discuss plans for the future.

**Meeting: ELIXIR EXCELERATE WP1 Meeting (Feb 20-22 2018, Copenhagen, DK)**

https://tinyurl.com/wp1f2f-2018

A face-to-face meeting to discuss matters around ELIXIR EXCELERATE WP1 (tools) developments. The meeting is primarily for WP1 partners, however anyone who is involved in *bio.tools* development is welcome to attend.

**Workshop: Debian Med 2018 Sprint (Feb 10-12 2018, Barcelona, ES)**


This is an informal co-working and co-learning event, participants are welcome to attend on the days that work for their schedule. EDAM and the integration of *bio.tools* with Debian Med and CWL was worked on.
Meeting: ELIXIR Tools Platform Meeting (Feb 8-9 2018, Barcelona, ES)
https://tinyurl.com/etp-feb2018
ELIXIR Tools Platform all-hands meeting to discuss activities of the platform and its projects.

Meeting: ELIXIR EXCELERATE WP2 Meeting (Feb 7 2018, Barcelona, ES)
https://docs.google.com/document/d/1-Ydv-SxTH_al4XaGh4g0I1mINgfMKG_yz6wNma1s9hY/edit
Meeting of ELIXIR EXCELERATE WP2 to discuss progress of OpenEBench, strategies for reaching out scientific communities running benchmark activities and practical examples on both technical monitoring and scientific benchmarking activities.

Workshop: bio.tools & EDAM @ 3rd NEUBIAS taggathon (Sep 11-14 2017, Gothenburg, SE)
http://eubias.org/NEUBIAS/what-is-taggathon/new-3-gothenburg-sweden/
The purpose of the taggathons is to implement and feed the content of NEUBIAS webtool; an organized repository of bio image analysis software and workflows for biologists, bioimage analysts and algorithm developers, complementary to ELIXIR bio.tools. The tagathon focuses on curation (identifying and tagging tools), semantics development including synonymous terms between Biology and Image Analysis, with development of EDAM-Bioimaging, and Semantic Web queries.

Conference: ELIXIR Denmark - 3rd Annual Danish Bioinformatics Conference (Aug 24-25 2017, Odense, DK)
http://elixir-node.cbs.dtu.dk/?page_id=2120
The third Danish Bioinformatics Conference organised by ELIXIR Denmark, bringing together members of the bioinformatics community from Denmark and across Europe.

http://tinyurl.com/registryhackathon14
An informal day of presentations, discussion and hacking around activities of the Danish ELIXIR node, including presentations about the ELIXIR Tools and Data Services Registry (https://bio.tools), bio.tools content and feature development, the EDAM ontology, applications of the registry, future plans and more.

Conference: BOSC 2017 (Jul 22-23 2017, Prague, CZ)
The Bioinformatics Open Source Conference (BOSC) is organized by the Open Bioinformatics Foundation (OBF), a non-profit group dedicated to promoting the practice and philosophy of open source software development and open science within the biological research community. BOSC has provided a forum for developers and users to interact and share research results and ideas in open source bioinformatics. EDAM was presented.

Technical hackathon: CodeFest 2017 (Jul 20-21 2017, Prague, CZ)
https://www.open-bio.org/wiki/Codefest_2017
This is an opportunity for anyone interested in open science, biology and programming to meet, discuss and work collaboratively. Everyone is welcome to attend. We will have a mix of experienced developers, newcomers to bioinformatics and everything in between. EDAM and bio.tools integration with the Common Workflow Language (CWL) were worked on.
Conference: ELIXIR All-hands (Mar 20-22 2017, Rome, IT)
https://www.elixir-europe.org/events/elixir-all-hands-2017
The third ELIXIR All Hands meeting, bringing together members of the ELIXIR community from across the ELIXIR Nodes, and collaborators from partner organisations, in order to review ELIXIR achievements and activities so far and discuss plans for the future.

Technical Hackathon: Visual Workflows in bio.tools (Mar 1-3 2017, Tallin, EE)
http://tinyurl.com/registryhackathon13
A three day workshop organised by ELIXIR-EE and partners aiming to implement a proof-of-principle for “visual workflows” in bio.tools: navigation of bio.tools content with cross-links to TeSS via diagrams for common analytical workflows.

Workshop: The future of proteomics in ELIXIR (Mar 1-2 2017, Tübingen, DE)
https://www.elixir-europe.org/events/strategic-workshop-future-proteomics-elixir
Focussed on creating a white paper to discuss the common infrastructures and services needed by the European proteomics community. bio.tools and EDAM were discussed.

Workshop: ELIXIR discovery portals (ELIXIR Innovation and SME Forum: Genomics and Health - Global resources for local Innovation, Feb 27-28 2017, Helsinki, FI)
The forum was aimed at the companies that use public bioinformatics resources in their business and would like to further streamline this process. The event was jointly organized by ELIXIR Finland, ELIXIR Estonia and the ELIXIR Hub. bio.tools was presented.
https://www.elixir-europe.org/events/elixir-innovation-and-sme-forum%3A-genomics-and-health-global-resources-local-innovation

Meeting: ELIXIR Tools Platform Meeting (Feb 22-23 2017, Barcelona, ES)
https://www.elixir-europe.org/events/elixir-tools-platform-all-hands-meeting
The 2nd meeting to discuss progress and plans for the [ELIXIR Tools Platform](https://www.elixir-europe.org/platforms/tools).

Workshop: bio.tools & EDAM @ 2nd NEUBIAS taggathon (Feb 13-15 2017, Oeiras near Lisbon, PT)
http://eubias.org/NEUBIAS/what-is-taggathon/taggathon-2-gulbenkian-oeliras/
The 2nd NEUBIAS Taggathon hosted and supported by the Gulbenkian Institute of Science, organized by the working group “Webtool” (WG4) of NEUBIAS, and in conjunction with the NEUBIAS training school and the following NEUBIAS conference. We extended the bioimaging sub-domain of EDAM in team work with bioimaging experts, and coordinated the development of bii.info/BISE with bio.tools.

Curatathon: Genomics tools in crop & animal breeding (Feb 2-3 2017, Aarhus, DK)
http://tinyurl.com/registryhackathon12
A curation hackathon aimed at curating software tools used for crop and animal breeding research.

**Workshop : bio.tools @ Debian Med Sprint (Jan 12-16 2017, Bucharest, RO)**
bio.tools folk join the Debian Med folk for co-hacking and co-learning. We improved EDAM annotations in Debian Med, and progressed towards importing high-quality software information from Debian (Med) to bio.tools.

**Thematic Hackathon : Computational Proteomics Resources (Jan 10-13, 2017, Semmering, AT)**
http://tinyurl.com/registryhackathon11
A thematic hackathon aimed at curating tools for computational proteomics, co-located with the Computational Proteomics Conference.

**Technical Hackathon : bio.tools @ NETTAB : (Oct 24 2016, Rome, IT)**
http://www.igst.it/nettab/2016/programme/hackathon/
http://tinyurl.com/registryhackathon10
A one day bioinformatics hackathon organized by ELIXIR held in occasion of the NETTAB 2016 Workshop. The hackathon will include the following two main strands: 1) Biosoftware description using bio.tools and schema.org. 2) Deployment of bioinformatics tools and services through Docker.

**Workshop: bio.tools & EDAM @ 1st NEUBIAS taggathon (Sep 14-16 2016, Barcelona, ES)**
The 1st NEUBIAS Taggathon hosted and supported by Universitat Pompeu Fabra, organized by the working group “Wehtool” (WG4) of NEUBIAS, and in conjunction with the NEUBIAS training school. The aim was to bring-in pre-incubated ideas and elements of the next biii.info/BISE webtool and to progress with its implementation. The presence of bio.tools and EDAM projects ensured coordination of NEUBIAS and EuroBioimaging registry and ontology developments with ELIXIR.
http://eubias.org/NEUBIAS/?page_id=228

**Conference: ELIXIR-DK @ ECCB (Sep 3-7 2016, The Hague, NL)**
http://www.eccb2016.org/
ELIXIR-DK will have a booth at ECCB and will showcase the work of the Danish ELIXIR node including the ELIXIR Tools & Data Services Registry (dev.bio.tools) and the EDAM ontology.

**Conference: ELIXIR-DK @ 2nd Annual Danish Bioinformatics Conference (Aug 25-26 2016, Odense, DK)**
http://www.conferencemanager.dk/DKBiC-2016/home.html
ELIXIR-DK will have a booth at DKBC and will showcase the work of the Danish ELIXIR node including the ELIXIR Tools & Data Services Registry (dev.bio.tools) and the EDAM ontology.

http://tinyurl.com/registryhackathon9

13.2. Past events
bio.tools Documentation, Release latest

An informal day of presentations, discussion and hacking, combining two events in one: 1) ELIXIR-DK staff technical get-together and 2) bio.tools workshop.

Conference: ELIXIR-DK @ IMSB 2016 (Jul 8-12 2016, Orlando, USA)
ELIXIR-DK will have a booth at IMSB 2016 and will showcase the work of the Danish ELIXIR node including the ELIXIR Tools & Data Services Registry (dev.bio.tools) and the EDAM ontology.

Technical Hackathon : Tools, Workflows and Workbenches (May 18-20, 2016, Institut Pasteur, Paris, FR)
http://tinyurl.com/registryhackathon8
A hackathon bringing together developers from key technical projects from ELIXIR and beyond including: the ELIXIR Tools & Data Services Registry (bio.tools), workbench/workflow projects (CWL, Galaxy, Taverna, Arvados), bioinformatics container solutions and registries, and the EDAM ontology.

Resource Hackathon : ELIXIR-SI Tools & Data Services (Apr 8, 2016, University of Ljubljana, SI)
ELIXIR-SI Registry Hackathon will take place on Apr 8, 2016 12-18h at the Faculty of Computer and Information Science (room PR05). The aim of the hackathon is to register Slovenian Bioinformatics Resources and create a national catalogue of Bioinformatics Tools and Data Services.

Thematic Hackathon : Metagenomics Training Resources (Apr 7-8, 2016, EMBL-EBI, UK)
Organised in collaboration with the GOBLET and the ELIXIR Training Platform.

Resource Hackathon : French Tools & Data Services (Mar 24-25, 2016, Gif-sur-Yvette, FR)
http://tinyurl.com/registryhackathon6
A hackathon bringing together representatives of French bioinformatics communities with the ELIXIR Tools & Data Services Registry, dedicated to the description and cataloguing of French tools and services, to boost their discovery and utility.

Resource Hackathon : Norwegian Tools & Data Services (Mar 16-18, 2016, NTNU Trondheim, NO)
A hackathon bringing together representatives of Norwegian bioinformatics communities with the ELIXIR Tools & Data Services Registry, dedicated to the description and cataloguing of Norway tools and services, to boost their discovery and utility.

Resource Hackathon : bio.tools @ Debian Med Sprint (Feb 4-7 2016, Lyngby, DK)
A resource hackathon focussed on curation and software development towards annotation and registration of tool packages from Debian Med. Annotation of Debian Med packages with EDAM.

Resource Hackathon : EMBL EBI tools (Jan 27-28 2016, EMBL EBI, UK)
A mini-hackathon aimed at curation of EMBL EBI software tools.

**Resource Hackathon : de.NBI EDAM Codefest (Jan 19-20 2016, Freiburg Uni., DE)**

http://tinyurl.com/registryhackathon7

This hackathon, organised by University of Freiburg, will focus on 1) annotation of de.NBI tools and services, 2) ELIXIR Registry and registration process and 3) Publishing tools in the ELIXIR Registry.

**Technical Hackathon : EDAM development heuristics (Dec 1-4 2015, Amsterdam, NL)**

http://tinyurl.com/registryhackathon5

This hackathon aimed at preparing EDAM for scaling with registry growth. The focus was to enumerate EDAM development heuristics to ensure usability, identify desirable clean-ups, and to devise quality assurance methods, including usability benchmarking in different scenarios. It also included a thematic session focussing on protein structural biology and the WHAT-IF package.

**Curatathon : bio.tools curation (Nov 4-6 2015, Brno, CZ)**

http://tinyurl.com/registryhackathon3

The second in the series, will aim for representation in the registry of all ELIXIR nodes, including new partners from Spain, Netherlands, Sweden and Finland, and other key resources beyond ELIXIR.

**Thematic Hackathon : RNA analysis (Sep 23-25 2015, Copenhagen, DK).**

A thematic hackathon focussed on RNA analysis and seeking to establish an ELIXIR RNA Tools Consortium that the Registry can draw upon in the future.

**Thematic Hackathon : defining good practice for resource annotation and registry curation (Aug 23-25 2015, Tallin, EE).**

http://tinyurl.com/registryhackathon4

A three day workshop organised and financed by ELIXIR-EE aiming to identify relevant processes and good practice for the annotation and curation of resources for their integration into the emerging ELIXIR infrastructure, focussed on next generation sequencing (NGS) analysis and the SeqWIKI Resource Hub.

**Technical Hackathon - EDAM Development & Governance (Mar 11-13 2015, Lyngby, DK)**

http://tinyurl.com/registryhackathon2

Focused on EDAM technical maintenance and usability, and produced a mock-up of tooling to assure optimal usage of EDAM for registry curation.

**Curatathon - Registration of Tool & Data Services (Nov 19-21 2014, Lyngby, DK)**

http://tinyurl.com/registryhackathon

Gathered representatives of institutes and key projects within ELIXIR and beyond. The participants performed a valuable pre-release critique of the Registry mechanism and interfaces, and added more than 300 resources to the content.
Mobyle, EDAM and Service Registry hackathon (Jun 17-18 2014, Paris, FR)

Workshop - ELIXIR, BioMedBridges & RDA Workshop: A common vocabulary to classify resources in the life sciences (Oct 7-8 2014, Brussels, NL)

ALLBIO Workshop - Metagenomics & interoperability (Apr 10-12 2014, Amsterdam, NL)

BioMedBridges AGM Tools Workshop (Mar 9-12 2014, Florence, IT)

bio.tools @ Debian Med Sprint (Jan 31-Feb 3 2014, Aberdeen, UK)

ELIXIR/BioMedBridges Workshop on Tool Registries (Oct 16-18 2013, CBS-DTU, DK)

BioMedBridges Registry Workshop (May 8 2013, Imperial College, UK)

AllBio / EMBRACE Continuity Workshop (Mar 18-20 2013, Amsterdam, NL)

BioMedBridges AGM Registry Workshop (Mar 11-12 2013, Dusseldorf, DE)

EDAM hackathon (Oct 9-13 2012, EMBL-EBI, UK)

AllBio workshop - ¡”Web services for improved interoperability in bioinformatics¢ (Oct 2-5 2012, Munich, DE)

### 13.3 Code of Conduct

We respectfully ask all attendees at meetings to conduct themselves in a way that maintains focus, respect, order - and enjoyment! Suggestions include:

- Bear in mind that you are as responsible for the success of the meeting as anyone else.
- Stick to the meeting agenda if stipulated (most of our meetings do not have rigid agendas).
- Remain focused on the task at hand.
- Come prepared.
- Use an analytic, facts-based approach to problem solving whenever possible.
- Manage meeting time wisely.
• Brainstorm when fresh ideas are in short supply or complex problems present challenges.
• Allow for the expression of every person’s ideas, and give all ideas a serious hearing.
• Listen carefully to each other, and be courteous.
• Accommodate disagreements and criticisms without hostility.
• Refrain from all personal attacks.
• Demonstrate flexibility.
• Make meetings enjoyable; employ humour and respect.
• Resolve conflict through compromise and consensus whenever possible.
bio.tools follows a simple governance model of three tiers under the leadership of the Danish ELIXIR node (Professor Søren Brunak, Head of Node) which is providing long-term funding and support for bio.tools. Development on the ground is led by Danish Node staff, currently Jon Ison, Piotr Chmura and Hans Ienasescu, in close collaboration with the registry-dev group (see below) and EDAM developers.

If you’d like to get involved with the project please mail registry@elixir-dk.org.

14.1 registry-dev

registry-dev includes the technical and scientific experts at the heart of the development and curation of bio.tools. Priorities are set in a quasi-democratic way; bio.tools is a “doocracy” with the Danish ELIXIR Node staff having the final say, where necessary (in so far as this is meaningful). registry-dev members are either funded, or have the intent and some bandwidth, to support bio.tools in the long-term. Danish Node staff ensure the registry-dev group and all Contributors are listened to and informed.

Members of registry-dev are responsible for agreeing aims and general good practice. They are expected to advocate bio.tools and (as bandwidth allows) collaborate with one another to help develop the registry software, related technical projects and registry content, e.g.:

- add new and improve existing content through collaboration with EDAM Developers
- routine content maintenance including quality control
- work collaboratively within the Curation Task Force (see below) and attend Hackathons
- suggest or implement new features
- develop software for the registry and related technical projects
- evaluate the registry and provide feedback, to ensure the registry software is fit for purpose

Danish Node staff are responsible for reporting software development and curation priorities, and progress, to the ELIXIR-DK Management.
registry-dev will assemble virtually or in person as circumstances dictate, in meetings with open agenda and followed up with actions and notes on key recommendations. registry-dev members are signed up to the registry-core mailing list.

14.1.1 Thematic editors

Named thematic editors are registry-dev members responsible for overseeing coverage and quality in specific thematic areas, e.g.

- evaluating existing coverage (EDAM, tools)
- driving coverage (EDAM, tools)
- liaising with community & leading workshops in their specialist area

See the Editors Guide.

14.2 Registry contributors

Registry contributors include anyone who makes significant contributions to the registry content or registry-related software, by whatever means, but have none of the responsibilities or expectations of registry-dev.

An important (but voluntary) role of contributors is to function in an advisory capacity, i.e. review the progress and priorities of registry-dev and advise them on their priorities and how best to achieve the current aims. To these ends, the following actions are welcome:

- read the bio.tools milestones and changelog and provide feedback on the reported progress and priorities.
- oversee the curation and development of bio.tools and actively offer constructive advice based on their practical experience, requirements and expertise
- advocate bio.tools to colleagues

The registry-dev group will respect this feedback and advice and reflect it in subsequent rounds of development and curation. We very much welcome new contributors: for further information please mail registry@elixir-dk.org.

14.3 Registry end-users

We particularly welcome input from end-users from the life science community including scientists, technicians and managers from academia and industry:

- to test, evaluate and critique the registry software and content
- to provide feedback and constructive advice based on their practical experience, requirements and expertise

registry-dev will respect this feedback and advice and reflect it in subsequent rounds of registry development and curation. Anyone who is considering using the registry - but especially typical scientist / bioinformatician end-users - are welcome to mail registry@elixir-dk.org.
Note: You can contact the registry-dev group by mailing registry-core@elixir-dk.org

15.1 registry-dev

- Jon Ison (DTU, DK) - technical coordination, lead engineer for biotoolsSchema & EDAM ontology development
- Hans-Ioan Ienasescu (DTU, DK) - curation (lead), Web development
- Piotr Chmura (KU, DK) - technical coordination, software development (lead), bio.tools development (backend)
- Hervé Ménager (Institut Pasteur, FR) - workbench integration, user engagement, ontology & schema development
- Kenzo Hillion (Institut Pasteur, FR) - workbench integration,
- Matúš Kalaš (University of Bergen, NO) - schema & ontology developer, user engagement, ontology & schema development
- Ahto Salumets (UT, EE) - curation
- Tomáš Raček (Masaryk University, CZ) - curation
- Alban Gaignard (CNRS, France) Semantic Web applications
- Anne Wenzel (RTH, DK) - curation (RNA tools)
- Erik Jaaniso (UT, EE) - software development, lead engineer for edammap
- Bjoern Gruening (University of Freiburg, DE) - de.NBI & Galaxy integration
- Dmitry Repchevsky (BSC, ES) - Web services & monitoring
- Jacques van Helden (Aix-Marseille University, FR) - advisor
• Dan Bolser (EMBL-EBI, EU) - WIKI integration
• Magnus Palmblad (LUMC, NL) - msutil.org integration
• José María Fernández (CNIO, ES) - benchmarking
• Karel Berka (Palacky University, CZ) - advisor
• Michael Crusoe (Common Workflow Language project) - advisor, CWL integration
• Peter Juvan (University of Ljubljana, SI) - curation
• Rabie SAIDI (EMBL-EBI, EU) - text mining
• Salvador Capella (INB, ES) - benchmarking
• Sebastien Moretti (SIB, CH) - curation
• Severine Duvaud (SIB, CH) - SIB / ExPASy integration
• Tunca Dogan (EMBL-EBI, EU) - text mining
• Wojtek Dabrowski (RKI, DE) - benchmarking

15.2 registry-dev (Thematic Editors)

• José Maria Carazo (CNB/CSIC, ES) - electron microscopy
• Josep Gelpí (INB / BSC-CSN, ES) - structural bioinformatics, benchmarking & tools interoperability
• Juergen Haas (University of Basel, CH) - protein structural biology, benchmarking
• Marta Villegas (BSC, ES) - NLP and text mining
• Veit Schwämmle (SDU-BMB, DK) - proteomics, ontology development, bio.tools applications
• Vivi Raundahl Gregersen (Aarhus University, DK) - agricultural science
• Carlos Oscar Sorzano (CNB/CSIC, ES) - electron microscopy

15.3 registry-dev (tentative)

• Anthony Bretaudeau (INRA - GenOuest/BIPAA)
• Christian Anthon (University of Copenhagen)
• Laura Emery (EMBL-EBI)
• Olivier Collin (CNRS - GenOuest)
• Peter Rice (Imperial College London)
• Priit Adler (University of Tartu)
• Steffen Möller (University of Rostock, DE)

15.4 Registry Contributors

Thanks to the many people who have contributed - if you're not listed below, please let us know!

• Aleksandra Nenadic (University of Manchester)
• Anders Dannesboe (BIRC, DK) - virtualization / container services
• Anthony Bretadeau (INRA - GenOuest/BIPAA)
• Bjoern Gruening (Uni-Freiburg)
• Bren Vaughan (EMBL-EBI, EU) - EBI integration
• Carole Goble (ELIXIR-UK)
• Chris Morris (STFC)
• Christian Anthon (University of Copenhagen)
• Christophe Blanchet (ELIXIR FR)
• Dan Bolser (EMBL-EBI, UK)
• Daniel Faria (FCG)
• Daniel Kahn (INRA, Lyon 1 University & PRABI)
• Emil Rydza (KU, DK)
• Federico Zambelli (CNR-IBBE)
• Frederik Coppens (VIB, BE)
• Gert Vriend (CMBI, NL)
• Gianluca Della Vedova (Univ. Milano-Bicocca, IT)
• Gianni Cesereni (University of Rome “Tor Vergata”)
• Giuseppe Profiti (ELIXIR-IT & University of Bologna, IT)
• Gonçalo Antunes (INESC-ID)
• Guy Yachdav (TUM, DE)
• Hedi Peterson (University of Tartu)
• Heinz Stockinger (SIB Swiss Institute of Bioinformatics)
• Helen Parkinson (EMBL-EBI, UK)
• Henriette Husum Bak-Jensen (UCPH, DK)
• Hervé Ménager (Institut Pasteur)
• Inge Jonassen (ELIXIR NO)
• Ivan Mičetić (University of Padova)
• Jan Brezovský (International Clinical Research Center and Masaryk university)
• Jiří Vondrášek (ELIXIR-CZ)
• José María Fernández (CNIO)
• Karel Berka (UPOL, CZ)
• Kaur Alasoo (University of Tartu)
• Kristian Davidsen (DTU, DK)
• Kristoffer Rapacki (DTU, DK) - advisor
• Laura Emery (EMBL-EBI)
• Luana Licata (University of Rome “Tor Vergata”)

15.4. Registry Contributors
• Ludek Matyska (Masaryk University)
• Lukasz Berger (DTU, DK)
• Manuela Helmer-Citterich (University Tor Vergata, Rome)
• Maria Maddalena Sperotto (DTU, ELIXIR-DK)
• Marie Grosjean (IFB, FR)
• Marie-Paule Lefranc (IMGT, IGH, CNRS, Université de Montpellier)
• Niall Beard (University of Manchester)
• Niclas Jareborg (ELIXIR SE)
• Olivier Collin (CNRS - GenOuest)
• Paola Roncaglia (EMBL-EBI)
• Paolo Romano (IRCCS AOU San Martino IST)
• Peter Juvan (University of Ljubljana)
• Peter Rice (Imperial College London)
• Priit Adler (University of Tartu)
• Rabie Saidi (EMBL-EBI, UK)
• Radka Svobodova (MU, CZ)
• Rafael Jimenez (ELIXIR HUB)
• Rodrigo Lopez (EMBL-EBI)
• Rune Friborg (Birc, au)
• Rune Møllegaard Friborg (BIRC, DK) - virtualization / container services
• Sebastien Moretti (SIB Swiss Institute of Bioinformatics)
• Severine Duvaud (SIB Swiss Institute of Bioinformatics)
• Silvio Tosatto (University of Padua)
• Sofia Kossida (IMGT, IGH CNRS, University of Montpellier)
• Steven Newhouse (ELIXIR EMBL-EBI)
• Tatyana Goldberg (TUM, DE)
• Timothy Karl (TUM, DE) (2remove: another important contact @rostlab)
• Tunca Dogan (EMBL-EBI, UK)
• Vegard Nygaard (ELIXIR NO)
• Victor de la Torre (INB)
• Wiktor Jurkowski (Earlham, UK)
Note: We are in the process of packaging the system in a docker container to achieve a more reproducible, environment-independent local deployment solution. In the meantime please see our guidelines below which summarize necessary components and steps to run bio.tools.

16.1 Source code

The source code of the registry is under standard GPL 3.0 license and is available from https://github.com/bio-tools/biotoolsRegistry/.

16.2 Architecture

The architecture of the registry is designed around the principle of decoupling frontend from the backend in order to eliminate performance bottlenecks at scale. The backend serves the data through the RESTful APIs which are in turn consumed by the frontend SPA (Single-Page Application). The main advantage of this paradigm is that the frontend uses the exact same API calls that any user can invoke in order to create their own customized interface. The overall goal of scalability can be achieved by hosting the components independently (master database, replicated cache layer and static frontend code in a CDN - Content Delivery Network - such as Cloudflare or Akamai).

16.3 Components

You only need Docker to run bio.tools locally.
CHAPTER 17

Installing bio.tools on your system

The local (development) installation is done via Docker. Other than Git (and a text editor), nothing else is required to run and write code for bio.tools.

17.1 1. Download and Install Docker

17.1.1 Docker main installation page

https://docs.docker.com/install/

Note: You will need to create a Docker Hub account.

Windows

https://docs.docker.com/docker-for-windows/install/

Note: Read the “What to know before you install” information to see if Docker Desktop for Windows can be installed on your system. If your system does not meet the requirements to run Docker Desktop for Windows, you can install the legacy Docker Toolbox.

MacOS

https://docs.docker.com/docker-for-mac/install/

CentOS

https://docs.docker.com/install/linux/docker-ce/centos/

Debian

https://docs.docker.com/install/linux/docker-ce/debian/

Fedora
17.2 2. Clone the repo

17.2.1 Using HTTPS

```bash
git clone https://github.com/bio-tools/biotoolsRegistry.git
```

17.2.2 Using SSH

```bash
git clone git@github.com:bio-tools/biotoolsRegistry.git
```

Go into the folder in which you cloned the bio.tools repo. By default it will be called `biotoolsRegistry`: (e.g. `cd biotoolsRegistry` or `cd /home/user/coding/biotoolsRegistry`)

17.2.3 3.0 Inside the bio.tools repo

**Note:** The Docker setup will require up to 5 GB of disk space. The bio.tools data will also add to this.

17.2.4 3.0.1 Build the necessary Docker images

```bash
docker-compose build
```

The above command will download / build all the Docker images required for bio.tools to run on your local machine. The images built can be seen by running: `docker image ls` and are:

- `biotools/frontend (~ 827MB)`
- `biotools/backend (~ 1.12GB)`
- `mysql (~ 205MB)` (will show up after running 3.0.2)
- `elasticsearch (~ 486MB)` (will show up after running 3.0.2)
- `python (~ 925MB)`
- `node (~ 650MB)`

17.2.5 3.0.2 Create and run the Docker containers

```bash
docker-compose up
```

The above command will create and run the required containers:

- `biotools-mysql`
• biotools-elasticsearch
• biotools-backend (depends on biotools-mysql and biotools-elasticsearch)
• biotools-frontend (depends on biotools-backend)

Note: After running the `docker-compose up` command, the containers will start and will output log messages which you can see in your terminal window. In order for the containers to keep running this window needs to stay open. You will need to open new terminal windows/tabs for other operations.

docker-compose up will also build the images if they do not exist, but in order to be sure your latest source code and image changes are running make sure you run docker-compose build beforehand.

Too see the running containers run: `docker container ls`

17.3 3.1 The short(er) setup

Run the steps below in the root folder of the Git project (e.g. biotoolsRegistry)

17.3.1 3.1.1 Make migrations

```
docker exec biotools-backend python manage.py makemigrations
```

Make Django migrations from the exiting models. Executed on the biotools-backend container. If you get the No changes detected message it means that you are up to date.

17.3.2 3.1.2 Migrate to the DB

```
docker exec biotools-backend python manage.py migrate
```

Create necessary tables and other DB objects from the migrations. Executed on the biotools-backend container. If you get the No migrations to apply. message it means that you are up to date.

17.3.3 3.1.3 Copy initial (seed) DB

```
docker cp initial_db.sql biotools-mysql:/root
```

Copies the initial_db.sql SQL file into the biotools-mysql container (where the MySQL database server runs) into the /root folder.

17.3.4 3.1.4 Copy initial DB load script file

```
docker cp load_initial_db.sh biotools-mysql:/root
```

Copies the load_initial_db.sh into the biotools-mysql container. This file will run the MySQL commands used to load the database described in initial_db.sql
17.3.5 3.1.5 Execute initial DB load script file

docker exec biotools-mysql bash /root/load_initial_db.sh

Executes the load_initial_db.sh file in the biotools-mysql container which loads the initial (seed) DB data.

Note: The initial DB contains 11 tool annotations, a superuser (username: biotools, password: biotools), an initial test subdomain and the necessary EDAM files. See 3.1.8 for more.

17.3.6 3.1.6 Purge Elasticsearch

docker exec biotools-backend python manage.py es_purge

Purges (clears) any data in the Elasticsearch index. Executed in the biotools-backend container which communicates with the biotools-elasticsearch container.

17.3.7 3.1.7 Regenerate Elasticsearch

docker exec biotools-backend python manage.py es_regenerate

Takes all the tools, subdomains annotations etc. in the DB and creates the equivalent entries in the Elasticsearch index. Executed in the biotools-backend container.

17.3.8 3.1.8 Done

At this point you can go to http://localhost:8000 to see the local bio.tools homepage.

The test subdomain can be viewed at http://test.localhost:8000

You can login with the existing superuser (user: biotools, password: biotools).

All running Docker containers can be stopped by running: docker-compose down from the root Git folder. This will preserve the data in the MySQL database and Elasticsearch. To reinstantiate everything again run: docker-compose up.

Only need to run docker-compose build once at the beginning or if changes are made to the bio.tools Docker settings files.

If you wish to remove the data along with the containers run: docker-compose down -v which will also remove the Docker volumes which preserve the MySQL and Elasticsearch data.

17.4 3.2 The longer setup

This is an alternative to 3.1 in which some of the steps were contained in the initial DB files. This will start with no data.

Run the steps below in the root folder of the Git project (e.g. biotoolsRegistry)
17.4.1 3.2.1 Make migrations

docker exec biotools-backend python manage.py makemigrations

Make Django migrations from the exiting models. Executed on the `biotools-backend` container.

17.4.2 3.2.2 Migrate to the DB

docker exec biotools-backend python manage.py migrate

Create necessary tables and other DB objects from the migrations. Executed on the `biotools-backend` container.

17.4.3 3.2.3 Create a superuser

docker exec -it biotools-backend python manage.py createsuperuser

Prompts the creation of a superuser, need to input superuser name, email (optional) and password. Executed on the `biotools-backend` container.

17.4.4 3.2.4 Setup EDAM ontology

docker exec biotools-backend bash /elixir/application/backend/data/edam/update_edam.sh

Download EDAM ontology and push it to the DB. Can also be used to update to new EDAM version. The file which indicates the EDAM version is `<git_project_root>/backend/data/edam/current_version.txt`, e.g. `biotoolsRegistry/backend/data/edam/current_version.txt`

17.4.5 3.2.5 Copy helper tables SQL

docker cp update_site_settings.sql biotools-mysql:/root

Copies the `update_site_settings.sql` SQL file into the `biotools-mysql` container (where the MySQL database server runs) into the `/root` folder. This file contains SQL instructions used to create helper tables and settings for the project.

17.4.6 3.2.6 Copy script file to run helper tables

docker cp update_site_settings.sh biotools-mysql:/root

Copies the `update_site_settings.sh` into the `biotools-mysql` container. This file will run the MySQL commands described in `update_site_settings.sql`

17.4.7 3.2.7 Execute script file

docker exec biotools-mysql bash /root/update_site_settings.sh

Executes the `update_site_settings.sh` file in the `biotools-mysql` container which loads the helper tables and settings in the DB.

17.4. 3.2 The longer setup
**17.4.8 3.2.8 Purge Elasticsearch**

```bash
docker exec biotools-backend python manage.py es_purge
```

Purges (clears) any data in the Elasticsearch index. Executed in the `biotools-backend` container which communicates with the `biotools-elasticsearch` container.

**17.4.9 3.2.9 Regenerate Elasticsearch**

```bash
docker exec biotools-backend python manage.py es_regenerate
```

Takes all the tools, subdomains annotations etc. in the DB and creates the equivalent entries in the Elasticsearch index. Executed in the `biotools-backend` container.

**17.4.10 3.1.10 Done**

At this point you can go to [http://localhost:8000](http://localhost:8000) to see the local bio.tools homepage. Login with the user created in **3.2.3**

No tools or subdomains are available, add tools at [http://localhost:8000/register](http://localhost:8000/register) and subdomains at [http://localhost:8000/subdomain](http://localhost:8000/subdomain)

All running Docker containers can be stopped by running: `docker-compose down` from the root Git folder. This will preserve the data in the MySQL database and Elasticsearch. To reinstantiate everything again run: `docker-compose up`.

Only need to run `docker-compose build` once at the beginning or if changes are made to the bio.tools Docker settings files.

If you wish to remove the data along with the containers run: `docker-compose down -v` which will also remove the Docker volumes which preserve the MySQL and Elasticsearch data.

**17.5 4. Useful information**

**17.6 4.0 Basic usage**

After completing steps 1-3 above, the only required commands for basic use are

```bash
docker-compose up
```

and

```bash
docker-compose down
```

and perhaps

```bash
docker-compose down -v
```
17.7 4.1 Local dev

After running `docker-compose up` you will see a number of log messages. These messages come from the running containers:

- `biotools-mysql` (MySQL logs)
- `biotools-elasticsearch` (Elasticsearch logs)
- `biotools-backend` (Mostly Apache logs, sometimes Python logs)
- `biotools-frontend` (Gulp logs)

17.7.1 4.1.1 Backend dev

The `biotools-backend` container is based on an image which uses an Apache server. The logs from `biotools-backend` come from Apache or sometimes from Python.

**Note:** Changes in Python/Django/backend files will be reflected in the `biotools-backend` container, **BUT** because of how Apache works, the changes won’t be reflected in your browser `http://localhost:8000` until Apache is reloaded. In order to see the changes in the reflected in the browser you need to run:

docker exec biotools-backend /etc/init.d/apache2 reload

**Remember** to run the above command whenever you want to see your code changes reflected in your local bio.tools.

Bringing the containers down and up again will also work, but this takes significantly longer. The above command is almost instant.

Most issues with the backend code will be reflected in the browser at `http://localhost:8000/api/{some_path}`, e.g. `http://localhost:8000/api/tool` or `http://localhost:8000/api/jaspar` etc.


17.7.2 4.1.2 Frontend dev

The `biotools-frontend` container outputs logs from `gulp` ([https://gulpjs.com/](https://gulpjs.com/)) which bundles all frontend JavaScript and CSS code.

Every time you change and save a `.js` or `.css` file in the frontend, gulp will re-bundle everything automatically. This implies that all changes in the frontend are reflected automatically in the browser, unlike for the backend.

**Note:** If you have a syntax error in your JavaScript or CSS files, gulp will fail and you won’t see any changes reflected in the browser. So, if your changes are not reflected, look at the `biotools-frontend` logs of gulp which will indicate if you made a syntax error in your code.

17.8 4.2 Update EDAM

Similarly to section 3.2.4, in order to update to the latest EDAM version (or just use a different EDAM version) the `update_edam.sh` needs to be executed on the `biotools-backend` container.
The version number used for updating EDAM is specified in the file:

```bash
<git_project_root>/backend/data/edam/current_version.txt
```

In order to update to the latest EDAM version (e.g. 1.23) edit the `current_version.txt` file to store the value 1.23, save the file and run:

```bash
docker exec biotools-backend bash /elixir/application/backend/data/edam/update_edam.sh
```

The script file will download the specific EDAM version .owl file from https://github.com/edamontology/edamontology and execute the:

```bash
python /elixi/application/manage.py parse_edam
```

command in the `biotools-backend container`.

**Note:** The `current_version.txt` file is tracked by Git and any changes involving EDAM versions other than latest should not be pushed to the main branches of the repo.

## 17.9 4.3 Local email setup

Important to note that the email system used to send emails regarding account creation and password reset will not work as intended out of the box.

In order for the emails to work you need to provide credentials (email, password, smtp settings) in the `backend/elixirapp/settings.py` file. bio.tools production uses Zoho mail (http://zoho.com) which currently works well with our setup.

The easy way would be to make a Zoho email account and use that email information to make the email functionality run. Gmail and Yahoo were tried and the connections are blocked by Gmail and Yahoo because of security reasons. This is because Gmail and Yahoo don’t accept a simple username-password login and require more strict settings. Feel free to implement this in your bio.tools instance.

## 17.10 4.4 Docker notes

### 17.10.1 Build bio.tools Docker images

```bash
docker-compose build
```

### 17.10.2 Run bio.tools containers

```bash
docker-compose up
```

### 17.10.3 Stop bio.tools containers

```bash
docker-compose down
```
17.10.4 Stop bio.tools containers and remove data

docker-compose down -v

17.10.5 View running containers

docker container ls

17.10.6 View all containers

docker container ls -a

17.10.7 Remove stopped containers

docker container rm <CONTAINER_ID>

or

docker container rm <CONTAINER_ID1> <CONTAINER_ID2> <CONTAINER_ID3>

17.10.8 Force remove containers

docker container rm -f <CONTAINER_ID>

or

docker container rm -f <CONTAINER_ID1> <CONTAINER_ID2> <CONTAINER_ID3>

17.10.9 Prune containers (Remove all stopped containers)

docker container prune

17.10.10 View images

docker image ls

17.10.11 Remove image

docker image rm <IMAGE_ID>

or
docker image rm <IMAGE_ID1> <IMAGE_ID2> <IMAGE_ID2>

(will not work if containers are running this image)

17.10.12 Enter a container and run commands

Any of the bio.tools running containers can provide a bash terminal to run commands inside the containers (similar to `docker exec`). Examples of the commands are:

- docker exec -it biotools-mysql bash
- docker exec -it biotools-elasticsearch bash
- docker exec -it biotools-backend bash
- docker exec -it biotools-frontend bash

As an example, to view the info in a MySQL database table run:

1. docker exec -it biotools-mysql bash
2. In container: `mysql -u elixir -p` (password is by default 123)
3. In MySQL:

```
use elixir;
SELECT * FROM elixir_resource WHERE visibility = 1;
```

17.10.13 bio.tools Docker settings files:

Backend build config file

```
<git_project_root>/backend/Dockerfile
```

Backend dockerignore file

```
<git_project_root>/backend/.dockerignore
```

Frontend build config file

```
<git_project_root>/frontend/Dockerfile
```

docker-compose YAML config file

```
<git_project_root>/docker-compose.yml
```

17.10.14 Docker documentation:

- https://docs.docker.com/
- https://docs.docker.com/reference/
- https://docs.docker.com/engine/reference/commandline/container/
- https://docs.docker.com/engine/reference/commandline/image/
- https://docs.docker.com/config/pruning/
• https://docs.docker.com/compose/
• https://hub.docker.com/

17.11 API Guidelines

You can also check out our API instructions at the links below:

• API reference
• API Usage Guide
The registry content is freely available to all under the Creative Commons Attribution licence (CC BY 4.0).
The source code of the registry is under standard GPL 3.0 license.


19.1 Citation

If you use bio.tools, please cite:

If you use EDAM or its part, please cite:

For help, support and feedback, please mail registry-support.
Note: Help with these docs is greatly appreciated. You can work on them directly via GitHub or make comments, requests, report bugs etc by using the issue tracker.

Documentation for bio.tools and related projects are maintained in GitHub:

https://github.com/bio-tools/biotoolsDocs

Documentation files are written in reStructuredText and have the file extension .rst. Uploading a file to GitHub will trigger a rebuild of the docs. GitHub include the file index.rst which defines the menu structure. Each menu item corresponds to a GitHub file, which (by convention) should have the same name as the menu item: use concise names!

21.1 reStructuredText links

- rST and Sphinx CheatSheet
- Quick Reference
- Primer
- Full documentation
- Online editor
- readthedocs FAQ
- readthedocs : getting started
- readthedocs build process
- thread on wide table handling
- links in rst docs