
edamontologyDocs Documentation

Release latest

Apr 20, 2023

Contents

1	What is EDAM?	3
1.1	Browsing	3
1.2	Download	3
1.3	Scope	4
1.4	Architecture	5
1.5	Status	5
1.6	Priorities	6
1.7	Principles	6
1.8	Motivation	6
1.9	Applications	7
1.10	Citing EDAM	7
2	Getting involved	9
2.1	Mailing lists	9
2.2	Suggestions & requests	9
2.3	Joining the team	10
2.4	Hackathons	10
3	Technical details	11
3.1	Concepts	11
3.2	Relations	12
3.3	Concept types	14
3.4	Hierarchy depth	16
3.5	Terms and synonyms	16
3.6	Subsets	17
3.7	Identifiers & persistent URLs	17
3.8	Color scheme	17
4	Users Guide	19
4.1	General guidelines	19
4.2	Specific use-cases	20
5	Editors Guide	25
5.1	General considerations	25
5.2	Rules of thumb for EDAM development	26
6	Developers Guide	33

6.1	Technical recipes	33
6.2	EDAM release process	38
6.3	Continuous Integration	41
6.4	Modifications in a GitHub fork	42
7	Governance	43
8	Contributors	45
8.1	EDAM Developers	45
8.2	EDAM Editors	45
8.3	EDAM Advisory Group	45
8.4	Contributors	47
8.5	Recent workshops (2014 -)	47
9	Hangouts	49
10	Applications	51
10.1	ELIXIR Tools & Data Services Registry	51
10.2	EBI Train online	51
10.3	RAINBio registry of bioinformatics cloud appliances	51
10.4	SEQwiki	51
10.5	eSysbio	52
11	License	53
12	Frequently asked questions	55
12.1	Where do I ask questions about EDAM?	55
12.2	How I do I request new terms?	55
12.3	How do I cite EDAM?	55
12.4	I want to add loads of terms - what do I do?	55
12.5	Can I join the EDAM team?	55

This is the documentation for [EDAM](#).

Contents:

What is EDAM?

EDAM is a domain ontology of data analysis and data management in bio- and other sciences, and science-based applications. It comprises concepts related to analysis, modelling, optimisation, and data life-cycle. Targetting usability by diverse users, the structure of EDAM is relatively simple, divided into 4 main sections: Topic, Operation, Data (incl. Identifier), and Format.

EDAM is particularly suitable for semantic annotations and categorisation of diverse resources related to data analysis and management: *e.g.* tools, workflows, learning materials, or standards. EDAM is also useful in data management itself, for recording provenance metadata of processed data.

1.1 Browsing

You can browse EDAM visually here:

- [NCBO BioPortal](#) (all-newest *unstable* version)
- [OLS](#) (latest **stable** version)
- [EDAM Browser](#) (**new: all versions!**)
- and other ontology browsers

1.2 Download

Latest version

- <http://edamontology.org/EDAM.owl>

The very latest, *unstable* version:

- http://edamontology.org/EDAM_unstable.owl

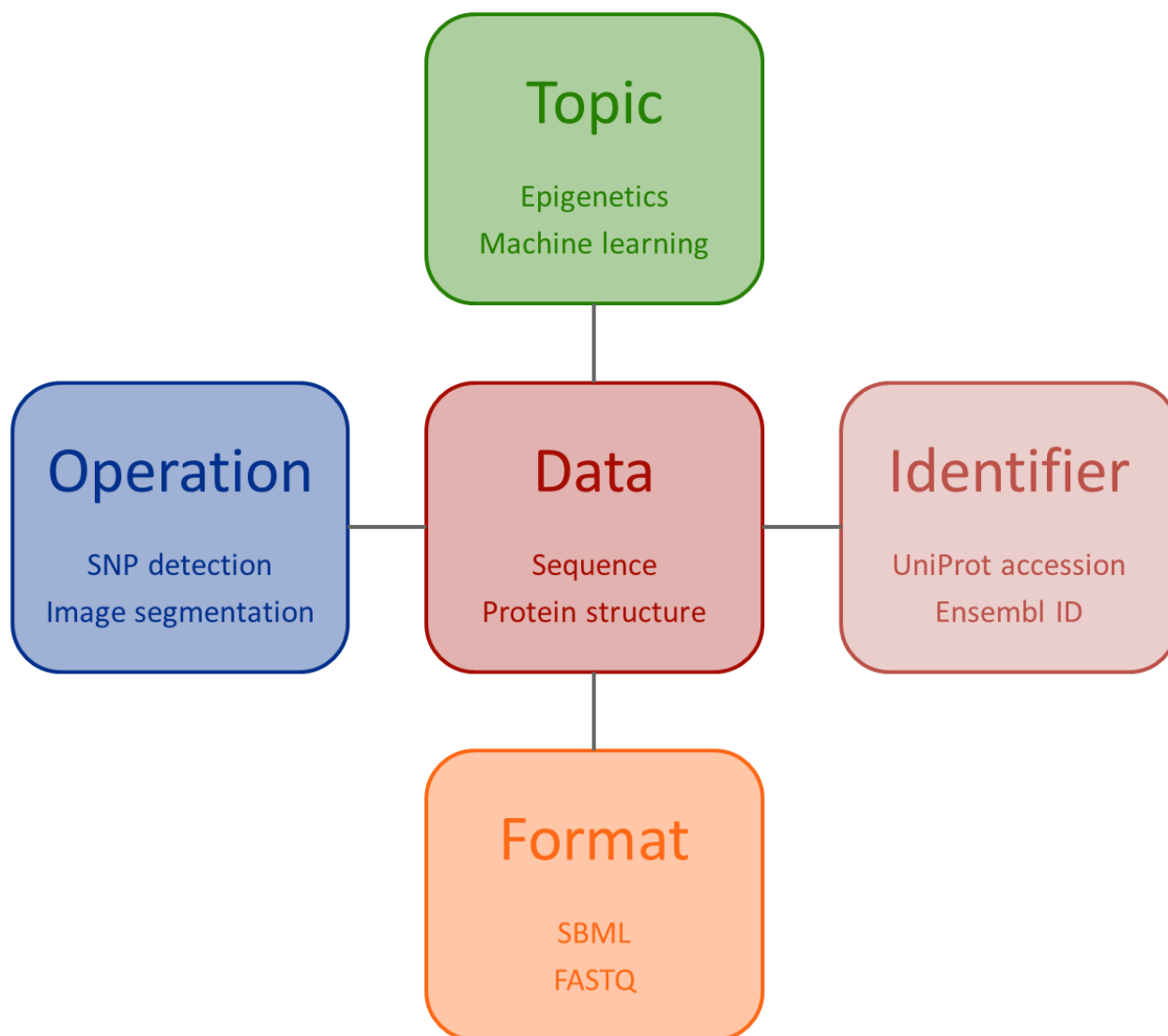
Versioned releases:

- [..edamontology.org/EDAM_<x.y>.owl](http://edamontology.org/EDAM_<x.y>.owl)

1.3 Scope

EDAM comprises 5 main sections:

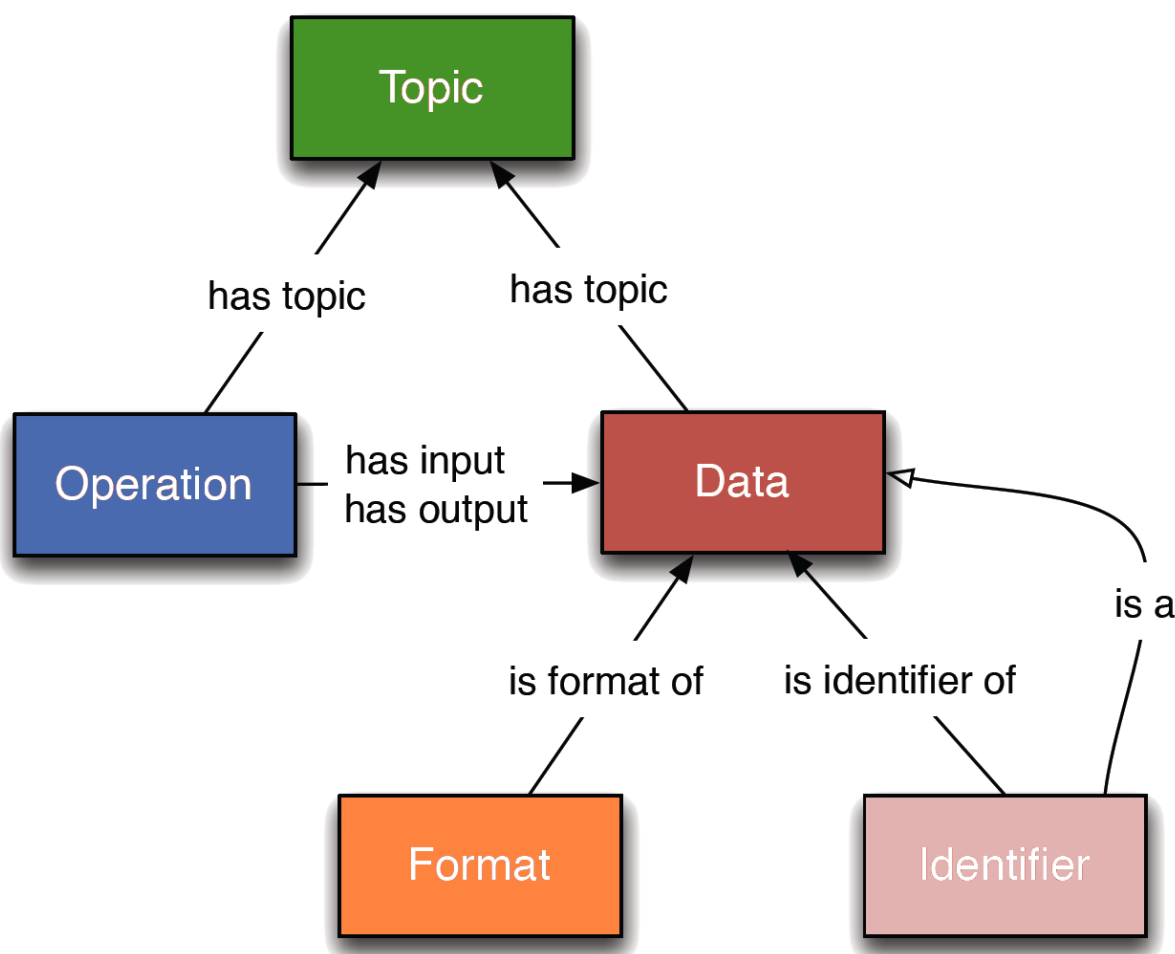
- **Topic** - A category denoting a rather broad domain or field of interest, of study, application, work, data, or technology.
- **Operation** - A function that processes a set of inputs and results in a set of outputs, or associates arguments (inputs) with values (outputs).
- **Data** - Information, represented in an information artefact (data record) that is ‘understandable’ by dedicated computational tools that can use the data as input or produce it as output.
- **Data->Identifier** - A text token, number or something else which identifies an entity, but which may not be persistent (stable) or unique (the same identifier may identify multiple things).
- **Format** - A defined way or layout of representing and structuring data in a computer file, blob, string, message, or elsewhere.



1.4 Architecture

EDAM has 3 components:

- **Concepts** - All concepts have a preferred label (or ‘term’) and definition. Further, a concept may have simple relations (see below) to other EDAM concepts, as well other intrinsic properties, *e.g.* an identifier may have a regular expression defining its syntax, and formats have links to their documentation/specification.
- **Hierarchy** - Every concept (excluding top-level concepts) is related to one or more other concepts within the same branch by an **is a** (specialisation) relation. Hence EDAM has 4 primary hierarchies (for *Data*, *Format*, *Operation*, and *Topic*).
- **Relations** - Concepts are defined as sharing specific relationships (see figure below), reflecting well established or self-evident principles; these relationships are used primarily to define internal consistency of EDAM.



1.5 Status

EDAM has been maturing steadily since its first creation ~2008. It is under active maintenance, developments are user-driven, for example by the [Bio.tools registry](#), [Galaxy](#), and other key infrastructure services. Future versions of EDAM will not depart fundamentally from the current structure (sections and relations). The development of EDAM can be followed on [GitHub](#).

For ways to contribute, please see the [documentation](#).

1.6 Priorities

Our core priority is to be responsive to users of EDAM. Further, to ensure essential EDAM maintenance and development remain on a sustainable footing, including:

- Content review and refactoring to ensure structural and semantic simplicity ensuring high usability
- Community build-up and development including more formal, but agile, governance and maintenance models and mechanisms
- Agile and responsive development of content in close collaboration with end-users and serving concrete use-cases
- Technical refactoring to minimise the cost of routine housekeeping and content development
- Implementation of tooling for routine maintenance to serve the needs of end-users, *e.g.* harvesting change requests and mappings between concepts

1.7 Principles

EDAM strives to uphold a few founding principles including:

- **Quality** - an ontology that is moderated
- **Openness** - development by the open community
- **Relevance** - prioritising use-case-driven development towards comprehensive but practical coverage
- **Practicality** - practical utility is valued over ontological “strictness” or any metaphysical doctrine
- **Clear scope** - respecting the scope of other complementary, well-developed ontologies and open linked data efforts
- **Familiarity** - including only concepts that are well established; familiar are prevalent and jargon is discouraged
- **Usability** - conceptual hierarchy with sufficient richness but only necessary complexity
- **Maintainability** - development must be efficient and sustainably up to date in the long term

EDAM is working towards implementing these principles fully and is open to suggestions.

1.8 Motivation

Scientists - professional and “citizen” - handle an increasingly large and diverse set of tools and data. Meanwhile, researchers demand ever more powerful and convenient means to organise, find, understand, compare, select, use and connect the available resources. These tasks often rely on consistent, machine-understandable descriptions of the underlying components, but these have been generally lacking in *ad hoc* resource descriptions. The urgent need - filled by EDAM - is for an ontology that unifies semantically the bioinformatics concepts in common use, provides the curator with a comprehensive controlled vocabulary that is broadly applicable, and supports new and powerful search, browse and query functions.

1.9 Applications

EDAM is suitable for large-scale semantic annotations and categorisation of diverse bioinformatics resources, including:

- Web APIs including RESTful APIs and SOAP/WSDL Web Services
- Application software
- Tool collections and packages
- Workflows / pipelines
- Databases
- XML Schemata and data objects
- Data syntax and file formats
- Web portals and pages
- Resource catalogues
- Training materials
- Courses, tutorials, and other events
- Documents, such as scientific publications

EDAM is suitable for diverse applications beyond annotation, for example within workbenches and workflow-management systems, software distributions, and resource registries.

1.10 Citing EDAM

An up-to-date description of EDAM is available in the following extended abstract and poster. If you use, or refer to EDAM or its part, please cite:

Melissa Black, Lucie Lamothe, Hager Eldakrouy, Mads Kierkegaard, Ankita Priya, Anne Machinda, Uttam Singh Khanduja, Drashti Patoliya, Rashika Rath, Tawah Peggy Che Nico, Gloria Umutesi, Claudia Blankenburg, Anita Op, Precious Chieke, Omodolapo Babatunde, Steve Laurie, Steffen Neumann, Veit Schwämmle, Ivan Kuzmin, Chris Hunter, Jonathan Karr, Jon Ison, Alban Gaignard, Bryan Brancotte, Hervé Ménager, Matúš Kalaš (2022). [EDAM: the bioscientific data analysis ontology](#) (update 2021) [version 1; not peer reviewed]. *F1000Research*, **11** (ISCB Comm J): 1. Poster. DOI: [10.7490/f1000research.1118900.1](https://doi.org/10.7490/f1000research.1118900.1) *Open access*

The “source code” and releases of EDAM have DOIs:

- DOI representing all stable versions, resolving to the latest: [10.5281/zenodo.822690](https://doi.org/10.5281/zenodo.822690)
- DOI of the latest stable version: <https://zenodo.org/badge/latestdoi/20960594>

The newest posters, available as PDF:

- [EDAM](#) (2022)
- [EDAM Bioimaging](#) (2020)
- [EDAM Browser](#) (2021)

CHAPTER 2

Getting involved

Important: EDAM is a community project: everyone is made to feel welcome and encouraged to get involved. The EDAM core developers have a limited capacity - we rely on your enthusiasm, contributions, and patience. Please help us build a better EDAM!

2.1 Mailing lists

To find the most efficient way to contribute to EDAM, to get help using EDAM for annotation and in software implementations, and for general discussions, e-mail:

edam@elixir-dk.org

We'll make every effort to be responsive given our limited resources. We will work with you to find the most efficient way to proceed depending on your requirements, expertise, and bandwidth.

To post or receive mail from the list above, subscribe here:

<http://elixirmail.cbs.dtu.dk/mailman/listinfo/edam>

For low-traffic announcements, subscribe here:

<http://elixirmail.cbs.dtu.dk/mailman/listinfo/edam-announce>

Note: Most discussion occurs on the [GitHub tracker](#)!

2.2 Suggestions & requests

Suggestions for addition, corrections, and other improvements to EDAM are always welcome!

If you have a GitHub account, you can make suggestions and requests by opening a GitHub issue. We strongly recommend you get an account, if needed, - it only takes a minute! Our tracker includes templates for various [types of suggestion](#):

- Go to <https://github.com/edamontology/edamontology/issues> and click on “New issue”
- If you’re not logged in, you’ll be asked first to log in or create an account
- Click the “Get started” next to the type of request you want to make
- Provide a title, and then follow instructions in the template, providing as much information as possible

Each template includes detailed instructions, which you’ll see when you create an issue, or you can read about them beforehand [here](#).

If you can’t find the template you need, click “Open a regular issue” and provide concise information, but sufficiently detailed to be actionable.

When requesting new concepts, the bare-bones required information (common to all subontologies) is:

- EDAM sub-ontology (Topic, Operation, Format, Data, Identifier)
- term (the most common term used to refer to the concept)
- short description (a couple of sentences)

More information will speed up the process, especially by including:

- URL of suggested parent
- exact synonyms (other commonly-used terms, acronyms *etc.* by which the concept is referred to)
- narrow synonyms (terms reflecting specialisations of the concept. Narrow synonyms can be defined to avoid creating overly-specialised concepts.)

If it turns out you need a lot of new concepts, we can find a more efficient way (*e.g.* shared Google sheet), and you can join us as an [EDAM Editor](#)

2.2.1 Suggestions form

If you’re unwilling to get a GitHub account, then simple requests for one or a few changes can be made using this form:

<http://tinyurl.com/EDAMChangeRequest>

If you require many changes and additions, there’s probably a more efficient way to proceed: please contact edam@elixir-dk.org. You’ll need to [subscribe](#) to the list first.

2.3 Joining the team

We very much welcome you to join the EDAM team. Please first read about the [Governance of EDAM](#) to find the level that is right for you. Then mail edam-dev@elixir-dk.org to get started.

2.4 Hackathons

EDAM developers participate in regular events, which you are welcome to attend. For more details see <https://bio.tools/events> - or host your own event and invite the EDAM team!

Note: An outline of the technical implementation of EDAM is below. EDAM makes light use of [OWL](#) logic/modelling and focuses on providing quality information for a basic set of *concepts*, with *relations* between these concepts and a very simple set of *rules* governing the conceptual relationships. Concepts have unique IDs and persistent URLs which resolve to a Web page providing all the information for that concept.

3.1 Concepts

3.1.1 Topic

“A general bioinformatics subject or category, such as a field of study, data, processing, analysis or technology.”

e.g. “Biology”, “Nucleic acid sites, features and motifs”, “Sequencing”.

Topic concepts provide very broad categories for annotation of diverse bioinformatics resources. There will only ever be a few hundred EDAM topics: biology or computer science is not exhaustively covered!

3.1.2 Operation

“A function or process performed by a tool; what is done, but not (typically) how or in what context.”

e.g. “Sequence alignment”, “Genome indexing”, “Sequence database search”.

Operation concepts enable the annotation of software tool functions, from quite broad to very specific. There will be as many EDAM Operations as required to support practical applications, *e.g.* description / finding tools in [bio.tools](#).

3.1.3 Data

“A type of data in common use in bioinformatics.”

e.g. “Sequence alignment”, “Comparison matrix”, “Phylogenetic tree”.

Data concepts enable the annotation of types of data, *e.g.* inputs and outputs of tools. There are some categories of data (for structuring EDAM), and some common tool parameters, but mostly they correspond to specific data formats (see below), *i.e.* complex biological data. Note: a single type of data can be serialised in multiple data formats!

3.1.4 Identifier

“A label that identifies (typically uniquely) something such as data, a resource or a biological entity.”

e.g. “UniProt accession”, “EC number”, “Gene symbol”.

Identifier concepts enable the annotation of identifiers of data. There are some categories (for structuring EDAM), but mostly they correspond to specific data identifiers. Identifiers are typically simple strings and EDAM includes regular expressions defining valid string values. Identifiers are formally related in EDAM to **Data** concepts (see [is_identifier_of](#)).

3.1.5 Format

“A specific layout for encoding a specific type of data in a computer file or memory.”

e.g. “FASTA format”, “PDB format”, “mmCIF”.

Format concepts enable the annotation of specific data formats / syntaxes; some general purpose ones but mostly biological data formats.

To be included, a format must be in common use and formally specified or documented; EDAM includes links to these things. Formats are formally related in EDAM to **Data** concepts (see [is_format_of](#)).

3.2 Relations

3.2.1 Relation types

is_a

Defines a concept as a specialisation of another concept. If A *is_a* B, then A is a specialisation (child) of B, and B is a generalisation (parent) of A.

The *is_a* relation is transitive: if A *is_a* B and B *is_a* C then A *is_a* C.

All relations are transitive over *is_a*: *e.g.* if A *has_input* B and B *is_a* C then A *has_input* C, and if A *is_a* B and B *has_input* C then A *has_input* C.

e.g. “Pairwise sequence alignment” is_a “Sequence alignment”

has_input

Defines an **Operation** concept as reading (inputting) a **Data** concept.

e.g. “Sequence analysis” has_input “Sequence”

has_output

Defines an **Operation** concept as writing (outputting) a **Data** concept.

e.g. “Sequence alignment” has_output “Sequence alignment”

has_topic

Defines a **Data** or **Operation** concept as being within the scope of a **Topic** concept.

e.g. “Peptide identification” has_topic “Proteomics”

is_identifier_of

Defines that an **Identifier** concept identifies a **Data** concept.

e.g. “Sequence accession” is_identifier_of “Sequence”

is_format_of

Defines that a **Format** concept is the format of a **Data** concept.

e.g. “Sequence record format” is_format_of “Sequence record”

3.2.2 Rules

Rules define how concepts are related.

Rules by concept

Topic

- **Topic is_a Topic** (specialisation of a topic)

Operation

- **Operation is_a Operation** (specialisation of an operation)
- **Operation has_input Data** (inputs a type of data)
- **Operation has_output Data** (outputs a type of data)
- **Operation has_topic Topic** (within a topic)

Data

- **Data is_a Data** (specialisation of a type of data)
- **Data has_topic Topic** (within a topic)

Format

- **Format is_a Format** (specialisation of a data format)
- **Format is_format_of Data** (a format specification of a data type)

Identifier

- **Identifier is_identifier_of Data** (identifier of a data type)

Rules by relation

is_a

- **Topic** *is_a* **Topic**
- **Operation** *is_a* **Operation**
- **Data** *is_a* **Data**
- **Format** *is_a* **Format**

has_input

- **Operation** *has_input* **Data**

has_output

- **Operation** *has_output* **Data**

has_topic

- **Operation** *has_topic* **Topic**
- **Data** *has_topic* **Topic**

is_identifier_of

- **Identifier** *is_identifier_of* **Data**

is_format_of

- **Format** *is_format_of* **Data**

3.3 Concept types

EDAM concepts are defined internally (via `<oboInOwl:inSubset>`) as one of two types:

- **Placeholder concepts** are high-level (conceptually broad), and used primarily to structure EDAM, providing placeholders for *concrete concepts*. They're not intended to be used much, or at all, for annotation.
- **Concrete concepts** are lower-level (conceptually more narrow) and are intended for annotation. *All* leaf nodes are concrete.

These notions depend upon the subontology (see below).

Note: EDAM topics are conceptually very broad categories with no clearly defined borders between each other: the notion of placeholder and concrete concepts doesn't apply!

3.3.1 Placeholder concepts

- **Operation placeholders** include high-level (abstract) operations *e.g. Analysis, Prediction and recognition*, and sometimes variants *e.g. Sequence analysis*.
 - all Tier 1 (children of [Operation](#)) and some Tier 2 operations are placeholders
- **Data placeholders** include:
 - basic technical types, *e.g. Score*
 - broad biological types *e.g. Phylogenetic data*

- they mostly appear in Tier 1 (children of **Operation**), rarely in Tier 2, and never below that (note, not all Tier 1 **Data** concepts are placeholders)
- **Identifier placeholders** include:
 - *basic type* one of **Accession** or **Name**. All concrete identifiers are a child of one of these.
 - *type of data* under **Identifier (by type of data)** e.g. “*Sequence accession (protein)*”. These mirror the **Data** subontology. All concrete identifiers are a child of one of these.
 - **Identifier (hybrid)**: a concrete identifier is a child of this if it’s re-used for data objects of fundamentally different types (typically served from a single database)
- **Format placeholders** include:
 - *basic type* currently **Textual format**, **Binary format**, **XML**, **HTML**, **JSON**, **RDF format** and **YAML**. All concrete formats are a child of one of these.
 - *type of data* under **Format (by type of data)** e.g. *Alignment format*, *Image format* etc.. All concrete formats are a child of one of these.

Note: The *data type placeholders* used in the **Identifier** and **Format** subontologies reflect the EDAM **Data** subontology. They serve purely to aid navigation, by providing an additional axis over (the same set of) concepts under “*Accession*” and “*Name*” (**Identifier**) or “*Textual format*”, “*Binary format*”, etc. (**Format**). Once ontology browsers better support rendering of conceptual relationships, it may no longer be necessary to support in EDAM the **Format (by type of data)** and **Identifier (by type of data)** patterns.

3.3.2 Concrete concepts

- **Concrete operations**
 - have specific input(s) and/or output(s) (**Operation** *has_input* | *has_output* **Data** relation)
 - include low-level (specific) operations (e.g. **Protein feature detection**) and in some cases variants (e.g. **Protein binding site prediction**) and sub-variants (e.g. **Protein-nucleic acid binding prediction**)
- **Concrete data types**
 - have a specific serialisation format (**Format** *is_format_of* **Data** relation)
- **Concrete identifiers**
 - have a corresponding data type (**Identifier** *is_identifier_of* **Data** relation)
 - normally have a regular expression pattern defining valid syntax of identifier instances (<regex> attribute)
- **Concrete data formats**
 - have a formal, public syntax specification (<specification> attribute)
 - in some cases, as practical necessity, there are format variants and sub-variants, e.g. *EMBL-like (XML)* and *FASTA-like (text)*

Note: The notions of “placeholder”, “concrete”, “broad”, “narrow” etc. are of course not hard and fast. As a work in progress, all placeholders and concrete concepts will be formally annotated as such, this *under discussion*. The addition of *has_input* and *has_output* relations is also a work in progress.

3.4 Hierarchy depth

There are limitations on the number of placeholders, and concrete concepts, that are chained (via *is_a* relation) together. In practice, this results in a maximum depth of each subontology.

- **Topic:**
 - each concept must have a path to root not exceeding 4 levels, e.g. **Topic** -> “*Computational biology*” -> “*Molecular genetics*” -> “*Gene structure*” -> “*Mobile genetic elements*”
 - 5 levels deep max. (in exceptional circumstances, an alternative path to root may exist, but never deeper than 5 levels)
- **Operation:**
 - maximum chain of 3 placeholders
 - maximum chain of 3 concrete operations
 - thus 6 levels deep max.
- **Data:**
 - maximum chain of 2 placeholders
 - maximum chain of 2 concrete data types
 - thus 4 levels deep max.
- **Identifier:**
 - maximum chain of 4 placeholders, e.g. **Identifier (by type of data)** -> “*Sequence identifier*” -> “*Sequence accession*” -> “*Sequence accession (protein)*”
 - maximum chain of 2 concrete identifiers
 - thus 6 levels deep max.
- **Format:**
 - maximum chain of 4 placeholders, e.g. **Format (by type of data)** -> “*Sequence record format*” -> “*FASTA-like*” -> “*FASTA-like (text)*”
 - maximum chain of 2 concrete formats
 - thus 6 levels deep max.

3.5 Terms and synonyms

EDAM uses:

- **Exact** synonym - a standard synonym (same meaning) of the primary term
- **Narrow** synonym - specialisms of the primary term
- **Broad** synonym - generalisations of the primary term (used in exceptional cases only)

All terms (primary and synonyms) are unique within a subontology, and (with a few exceptions) are unique *between* subontologies, too.

3.6 Subsets

<todo>

3.7 Identifiers & persistent URLs

Each EDAM concept has an alphanumeric identifier that uniquely identifies that concept. The general form is:

- `<namespace>_<4-digit-ID>`

where `<namespace>` refers to an EDAM subontology, one of:

- `topic`
- `operation`
- `data`
- `format`

and `<4-digit-ID>` uses numbers from 0 to 9. Note this number is unique across all subontologies.

The IDs are used in URLs that resolve to information about the concept, *e.g.*:

- `topic_0121`
- http://edamontology.org/topic_0121

These identifiers (and the URLs) persist between versions: a given ID and URI are guaranteed to continue identifying the same concept. This does *not* imply that terms, definitions and other information remains constant, but the IDs *will* remain essentially true to the original concept.

Occasionally, concepts become *deprecated* - designated as not being recommended for use. Deprecated concepts also persist; they are removed and will maintain their ID and URI. EDAM developers adhere to rules on [deprecataation](#), *e.g.* a replacement concept, or suggested replacement is given for all deprecated concepts.

3.8 Color scheme

In order to propose a coherent experience in environments using the EDAM Ontology, a color sheme have been chosen and it is recommended to use it. Images the color scheme can be found [here](#) and [there](#).

Branch	Color	Light color
Data	#a7544d	#d6b1ae
Identifier	#d6b1ae	#e2cdcb
Topic	#5f903d	#b4cca1
Operation	#556ca9	#a3b0d1
Format	#e6834c	#f2c4a9
Deprecated	#999999	#bbb

Important: EDAM has many [applicatons](#). If using EDAM to annotate software, we strongly recommend you read the [bio.tools Curators Guidelines](#) which includes [detailed information](#) for this purpose. A few more general, and more specific guidelines are below.

You can email the [EDAM mailing list](#) for help (after first [subscribing](#)).

4.1 General guidelines

4.1.1 Which EDAM sub-ontology to use?

1. “*Topic*” for coarse-grained annotation of diverse entities
2. “*Operation*” for fine-grained annotation of tool functions
3. “*Data*” for annotation of data in semantic terms
4. “*Format*” for annotation of the syntax or format of data
5. “*Identifier*” (as a special type of “*Data*”) for annotation of identifiers (names and accessions) of data or other entities

EDAM thus provides different semantic ‘axes’ for annotation. For example, [annotation of a software tool](#) might include:

- *Topic* - general scientific domain the software serves, *e.g.* “*Structural biology*”
- *Operation* - the precise function of the tool, *e.g.* “*Homology modelling*”
- *Data* - the primary input and output, *e.g.* “*Protein structure*”
- *Format* - the supported format(s) of the input and output, *e.g.* “*PDB format*”

4.1.2 Use of other ontologies

The expectation is for EDAM to be used alongside other ontologies for annotation where possible and desirable. For example, an operation that predicts specific features of a molecular sequence could be annotated with concepts from [SO](#) (Sequence Ontology) for the features. Other notable relevant ontologies include [GO](#) (the Gene Ontology) and the [NCBI taxonomy](#).

4.1.3 Picking concepts

If you have many annotations to do, it will help to familiarise yourself with EDAM first using a [browser](#).

1. Identify the correct sub-ontology (“*Operation*”, “*Data*” *etc.*) of concepts considering what is being annotated (see above)
2. Search EDAM using keywords to find candidate concepts. Multiple searches using synonyms, alternative spellings *etc.* are advisable.
3. Pick the most specific concept(s) available, or if a broader concept really captures what you need, then use it. Bear in mind some concepts (especially EDAM topics, are necessarily overlapping or general.
4. Only pick a correct concept. If it doesn’t exist, [request](#) it’s added to EDAM

4.2 Specific use-cases

4.2.1 Annotation of software tools

If using EDAM to annotate software, we strongly recommend you read the [bio.tools Curators Guidelines](#) which includes [detailed information](#) for this purpose.

4.2.2 Annotation of Web APIs

There’s been some R&D into EDAM annotation of Web APIs described using the [OpenAPI specification](#). See the [update on progress and pre-publication](#).

4.2.3 Annotation of Web service (SOAP+WSDL)

A Web service built on SOAP and WSDL is considered as an arbitrary (but usually related) set of one or more operations, reducing the problem of Web service interoperation to one of compatibility between operations.

Operation

- Discrete unit of functionality performing (typically) one or more definite functions
- Reads an input
- Writes an output
- Uses zero or more data resources

Input

- Payload (*e.g.* of HTTP or SOAP message) passed in operation call
- Name and (ideally) description is given (*e.g.* in WSDL file)
- Input has one or more XML elements which must be set (input values)

Output

- Payload (*e.g.* of HTTP or SOAP message) returned from operation call
- Name and (ideally) description is given (*e.g.* in WSDL file)
- Output has one or more XML elements which are written (output values)

XML elements

- Correspond to the inputs (parameters) and outputs of a service
- Are simple or complex XSD types given in an XML Schema (within or referenced from a WSDL file)
- Have values that are instances of specific semantic type.
- Have values in a specific syntax, either fully specified by the schema, or (occasionally) text in a specific file format which is not specified by the schema.

Levels of annotation

Annotation of a WSDL file or associated XSD schema is possible at several levels. Assuming SAWSDL annotation (<http://www.w3.org/TR/sawSDL/>), the XML elements that may be annotated by EDAM concepts are:

1. **Web service** (as a whole) (`<wsi:portType></wsi>`)
 - One (or more) “*Topic*” concepts to describe the general area(s) the service concerns
 - If applicable, one (or more) “*Operation*” concepts to describe the functions of the service (if all operations perform essentially the same function)
2. **Operation** (`<wsi:operation></wsi>` inside `<wsi:portType></wsi>`)
 - One (or more) “*Operation*” concepts for each WSDL operation (more than one in exceptional circumstances)
3. **Input parameters and their sub-parts** (`<xsi:element></xsi>`, `<xsi:complexType></xsi>`, `<xsi:simpleType></xsi>`, `<xsi:attribute></xsi>`) * One (or more) “*Data*” concepts * One (or more) “*Format*” concepts
4. **Output parameters and their sub-parts** (`<xsi:element></xsi>`, `<xsi:complexType></xsi>`, `<xsi:simpleType></xsi>`, `<xsi:attribute></xsi>`)
 - One (or more) “*Data*” concepts
 - One (or more) “*Format*” concepts

NB. The input and output parameters should be annotated inside the XML Schema that defines them. In case of services that are not following the highly recommended *document/literal wrapped* SOAP-binding style, the `<wsi:part></wsi>` inside `<wsi:message></wsi>` can be annotated (the same applies to *faults*, but meanings of faults are not modelled by EDAM)

The following annotations might be useful but are not directly recommended by SAWSDL:

1. **Enumerated values of input/output parameters** (`<xsi:enumeration></xsi>`)
 - One (or more) “*Format*” or “*Data*” concepts defining the particular enumerated value

For details of incorporating the SAWSDL annotations into WSDLs and XSDs, see below.

EDAM URIs and SAWSDL annotation

SAWSDL mandates the use of (`<kbd>sawSDL:modelReference</kbd>`) attributes for annotation. The format of EDAM URIs used inside this attribute includes the ontology name (*`http://edamontology.org`*), main sub-ontology, and the unique identifier (ID) of the particular concept:

```
<xmp>
<xs:element name="elementName" sawSDL:modelReference="http://edamontology.org/
↳subontology_id">
</xmp>
```

Where ...

- `<kbd>xs:element</kbd>` is the XML element being annotated (can be also `<kbd>xs:attribute</kbd>`, `<kbd>xs:complexType</kbd>`, `<kbd>xs:simpleType</kbd>`, `<kbd>sawSDL:attrExtension</kbd>`, `<kbd>wsdl:portType</kbd>`, in special cases `<kbd>wsdl:part</kbd>`, or eventually `<kbd>xs:enumeration</kbd>`)
- `<kbd>elementName</kbd>` is the name of the XML element

The value of the `<kbd>sawSDL:modelReference</kbd>` attribute is a URI pointing to the concept definition. The URI to use in case of EDAM includes the concept's sub-ontology:

- `<kbd>sub-ontology</kbd>` is the **top-level sub-ontology** of the EDAM concept; one of `<kbd>topic</kbd>`, `<kbd>data</kbd>`, `<kbd>format</kbd>`, or `<kbd>operation</kbd>`
- `<kbd>id</kbd>` is the unique local identifier of the concept, *e.g.* `<kbd>"0295"</kbd>`

So for these 3 concepts:

```
<xmp>
EDAM_topic:0182
EDAM_operation:0292
EDAM_data:0863
</xmp>
```

We'd have

```
<xmp>
http://edamontology.org/topic_0182
http://edamontology.org/operation_0292
http://edamontology.org/data_0863
</xmp>
```

Which can be used in SAWSDL annotation, *e.g.*

```
<xmp>
<wsdl:portType name="myService" sawSDL:modelReference="http://edamontology.org/topic_
↳0182">
<sawSDL:attrExtension sawSDL:modelReference="http://edamontology.org/operation_0292">
<xs:element name="outfile" sawSDL:modelReference="http://edamontology.org/data_0863">
</xmp>
```

If more than one annotation of an element is required, these can be given in the `<kbd>sawSDL:modelReference</kbd>` attribute delimited by space characters:

```
<xmp><wsdl:portType name="myService" sawSDL:modelReference="http://edamontology.org/
↳topic_0182 http://edamontology.org/operation_0292">
</xmp>
```

NB. Such multiple annotations need not be in the same namespace, and need not at all to refer to the same ontology.

SAWSDL guidelines for annotating operations

One peculiarity of the SAWSDL specification is that annotations on `<wsdl:operation>` element inside `<wsdl:portType>` should be handled using a `<sawSDL:attrExtensions>` element. This is not a requirement for other elements.

Importantly, the `<sawSDL:attrExtension>` element inside the `wsdl:operation` **must be before** `<wsdl:input>`, `<wsdl:output>` and `<wsdl:fault>` elements (so typically after the `<wsdl:documentation>` element).

For example:

```
<?xml version='1.0'?>
<xmp> <wsdl:portType name="Clustalw2PortType" sawSDL:modelReference="http://
↪edamontology.org/topic_0186 http://edamontology.org/operation_0496">
  <wsdl:operation name="submitClustalw2">
    <wsdl:documentation>Submit a sequence and get a jobID</wsdl:documentation>
    <sawSDL:attrExtensions sawSDL:modelReference="http://edamontology.org/operation_0496
↪"/>
    <wsdl:input message="submitClustalw2Msg"/>
    <wsdl:output message="submitClustalw2ResponseMsg"/>
  </wsdl:operation>
</xmp>
```

Some WSDL/XSD validators or SOAP libraries do not check for it, but some do require the strict order of these elements.

Welcome to the EDAM Editors Guide. It contains general best-practice (technical and scientific) guidelines when modifying EDAM; adding or changing concepts, concept metadata, crosslinking, *etc.*

If you're not sure how to do something please ask edam@elixir-dk.org. You'll need to [subscribe](#) to the list first.

5.1 General considerations

5.1.1 Terminology

We use the following terms when talking about EDAM:

- *EDAM* refers to the ontology in totality (all subontologies).
- *Subontology* and occasionally *branch* refers to an EDAM subontology, *i.e.* **Topic**, **Operation**, **Data** or **Format**. Also **Data->Identifier** (a branch of **Data**).
- *Concept* is the basic unit of information in EDAM: including concept definition, terms and other metadata
- *Placeholder* is a concept intended primarily to organise the EDAM tree (not normally used for annotation).
- *Concrete concepts* are intended primarily for annotation purposes.
- *Primary term*, *Primary label* or simply *Label* is the primary term by which the concept is referred to. They're used for annotation purposes.
- *Synonym* means an exact, narrow, broad or related synonym (see [todo](#)). They can also be used for annotation.
- *Term* and *terms* refer to primary labels and synonyms collectively.
- *Hierarchy* refers to the EDAM tree structure, resulting from EDAM concepts being defined as specialisations/generalisations of one another (PS. EDAM isn't a tree, it's a [DAG](#).)
- *Root* refers to the top-most concept in a subontology, *i.e.* **Topic**, **Operation**, **Data**, and **Format**. And (depending on context) **Identifier**.

- *Tier* refers to a particular level in the hierarchy, excluding the subontology root, *e.g.* “Tier 1 data concepts” means everything immediately under [Data](#).
- *Top level* and *Top tier* refers to Tier 1 concepts.
- *Child*, *Children of*, *Kids etc.* refers to concept(s) defined as an immediate specialisation of another (*i.e.* “is_a”, or the OWL geek “subClass”). Conversely *Parent* means the opposite (a generalisation of a concept).
- *Ancestor* means *Parent* or the parent’s parent *etc.* Conversely *Descendant* means *Child* or the children’s children *etc.*
- *related to* means a concept in one subontology is formally defined as related (in various ways) to a concept in another, but excluding basic specialisation/generalisation relationships.
- *Node* refers to a concept when it’s being discussed in context of the hierarchy.
- *Leaf* refers to a concept at the bottom of the tree (without children).

For a technical definition of these things, see [Technical details](#).

5.2 Rules of thumb for EDAM development

These rules of thumb are to guide the technical and scientific development of EDAM, to help ensure structural and conceptual simplicity and that EDAM is fit for purpose and will scale for annotation applications, especially [bio.tools](#).

Note: 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.

Important: If you are editing the EDAM.owl file directly (in a text editor or Protege) there are additional things you need to consider and do, which are not covered by these guidelines. For further information see [Developers Guide](#).

5.2.1 General

Important: Before proposing or making any major changes, make sure you understand the [principles](#) on which EDAM is based.

Concepts & Terms

Concepts:

- **MUST** be conceptually clearly distinct from one another. The exception is **Topic** ontology where most concepts are overlapping.
- **MUST** be genuine specialisms, wherever a concept is defined as the child of another.
- **MUST** have a primary term (preferred label) and definition (see below)
- **MAY** have a comment and one or more synonyms (see below)

Primary term and synonyms:

- **MUST** be a short name or phrase in common use
- **MUST** be unique within a sub-ontology
- **SHOULD** be unique across all sub-ontologies (rare exceptions are allowed)

Primary term:

- **MUST** reflect the vernacular, *i.e.* the term that's most commonly used when referring to the concept; you **SHOULD** use google (number of hits) to help you choose, where necessary
- **MUST** use British spelling
- **MUST** not include buzzwords and marketing-spiel *e.g.* "Big data", "NGS" *etc.*

Synonyms:

- **MUST NOT** overlap conceptually, to a significant extent, with an already existing concept; be especially mindful of ancestors and descendants of the concept for which a synonym is defined.
- **SHOULD** use British spelling
- **MAY** capture spelling variations, including American spellings, case and hyphenation variants *etc* (as exact synonyms)
- **MAY** include buzzwords if really prevalent and relevant

Definitions and comments:

- **SHOULD** use British spelling

Definitions:

- **MUST** be a concise and lucid description of the concept, without acronyms, and avoiding jargon.
- **MUST** reflect the primary term.

Comments:

- **MAY** include peripheral but important information not captured by the definition.
- **MAY** reflect narrow and broad synonyms of the primary term.

When adding a new concept, in addition to above:

- **SHOULD** provide all common *exact synonyms* of the primary term
- **MAY** provide any number of *narrow synonyms* (but be wary of conceptual overlap with child concepts). The exception is **Format** subontology which **MUST NOT** include any narrow synonyms at all.
- **SHOULD NOT** provide any *broad synonyms* unless these are really needed (but be wary of conceptual overlap with parent concepts)

Note: EDAM must always evolve, which means additions, edits, and occasionally *deprecations*: marking-up concepts as not recommended for use: the EDAM developers follow special [deprecation guidelines](#) for this.

Hierarchy

Important: EDAM has the notion of *placeholder* and *concrete* concepts (see [todo](#)):

- *placeholders* are conceptually very broad, and are intended primarily to organise the EDAM tree
- *concrete* concepts conceptually more narrow, and are intended primarily for annotation purposes

There are rules for how many *placeholders* and *concrete* concepts can be chained together (via *is_a*) relationships, and thus, the maximum depths of the subontology hierarchies (see [Developers Guide](#)).

In practice, as an Editor, you should be aware of the general structure of EDAM and the conceptual granularity in each subontology. If in doubt, mail the [EDAM developers](#) for advice.

When adding a new concept:

- **MUST** be sure - if an addition introduces a new level of depth - that it's realistic to also add and maintain (in due course) all relevant siblings, *i.e.* related concepts with the same parent. This is to ensure EDAM coverage does not get patchy.
- **SHOULD NOT** introduce any “single childs” (concepts without siblings) unless you already know of potential siblings (to add in due course), or think it's likely such sibling concepts will appear in the future
- **SHOULD NOT** add (or imply the addition, as per above) multiple concepts if this would mean a big overlap with an existing, well-developed ontology. If in doubt, discuss this first with the [EDAM developers](#).
- **SHOULD NOT** define multiple parents of a concept (except where indicated [below](#)) unless there is a very unambivalent case. This rule is even stronger for **Topics** (many topics overlap with each other, but as a rule you must pick one parent only)

5.2.2 Subontology-specific

Topic

Note: EDAM **topics** are conceptually very broad (see [Scope](#)). There will only ever include a few hundred concepts in total, semantic richness is captured through synonyms (which are unlimited in number). This ensures sustainability and practical applications. In contrast see *e.g.* [MeSH](#).

- Respect the [scope](#), specifically:
 - **MUST NOT** include fine-grained operations or types of data. As a rare exception, very high-level operations *e.g.* *Sequence analysis* **MAY** be included.
 - **MUST NOT** include any topic tied to a concrete project or product.
 - **SHOULD NOT** include anything that is more tangible than a very general topic. For example specific cell types, diseases, biological processes, environment types *etc* belong in their own ontology, but **MAY** be captured, where desirable, as synonyms in EDAM. Rare exceptions are allowed where a term really is in extremely prevalent usage (pragmatism rules!)

- **MUST NOT** define multiple parents of a topic, with the exception of the strongest cases only, where it would be incongruous not to do so *e.g.* *Biochemistry* is a child of both *Biology* and *Chemistry*.
- **MUST NOT** conflate terms in a concept label where these terms exist as independent topics already, *e.g.* *Disease pathways* is disallowed because there are already concepts for *Disease* (synonym of *Pathology*) and *Pathways* (synonym of *Molecular interactions, pathways and networks*). Instead, if such confluations are required, they **MAY** be added as synonyms of one concept or the other.
- **SHOULD** provide a link to [Wikipedia](#) if a relevant page exists. Most EDAM topics are sufficiently broad to already have Wikipedia pages. Exceptions are OK, but if a Wikipedia page does not exist, consider carefully whether the concept is too fine-grained.

Note: Links to Wikipedia are desirable, for the primary term but also synonyms. In a future refactoring, EDAM may distinguish such cases.

Operation

Note: EDAM **operations** (see [Scope](#)) range from conceptually quite broad to quite narrow. There will be as many as required to capture the *essential functions* of current bioinformatics software tools. Note “*essential*”: the Operation subontology will not descend to a level of conceptual granularity that is impractical from a maintenance or usage perspective.

Operation concepts are formally related to **Data**

- **Operation** *has_input* **Data**
- **Operation** *has_output* **Data**

-
- **MUST** never be more fine-grained than is useful for practical search purposes. This excludes fine-grained specialisations of a basic function, individual algorithms *etc.* (a few exceptions are allowed for very highly prevalent concepts). If in doubt, speak to the [EDAM developers](#))
 - **MUST** state in the definition *what* is done by the operation but not *how*
 - **SHOULD** provide a link to [Wikipedia](#) if a relevant page exists.
 - **SHOULD** have concepts in the **Data** subontology corresponding to the typical inputs/outputs of the operation (these can be added, if needed).

Data

Note: EDAM **data** concepts range from conceptually quite broad to quite narrow. There will be as many as required to capture the *basic types* of bioinformatics data. The Data subontology does (and will) not reflect individual data structures, and like **Operation**, will maintain a level of conceptual granularity that is maintainable and usable.

Data concepts are formally related to **Identifier** and **Format** concepts:

- **Identifier** *is_identifier_of* **Data**
- **Format** *is_format_of* **Data**

-
- **MUST** have a corresponding concept in the **Format** subontology, *i.e.* the serialisation format(s) of the data. New formats can be added, if needed.

- **MUST** include in the definition a very basic description of the data, usually in biological terms.
- **SHOULD** have a corresponding concept in the **Identifier** subontology, *i.e.* identifier(s) of the data, if these exist. New identifiers can be added, if needed.

Data->Identifier

Note: EDAM **identifiers** are very specific. There will be as many as required to capture the unique types of identifiers in use. Uniqueness means that a regular expression pattern can, in principle, meaningfully be created describing the identifier instance syntax.

Identifier and data concepts are formally related:

- **Identifier** *is_identifier_of* **Data**

-
- **MUST** have a corresponding concept in the **Data** subontology, *i.e.* the type of data that is identified. New data concepts can be added, if needed.
 - **MUST** include in the definition what type of data and/or name of database the identifier is used for.
 - **SHOULD** include a link to relevant documentation for the identifier, if available
 - **SHOULD** specify a regular expression pattern, defining valid values of instances of that identifier

Format

Note: EDAM **formats** are very specific. There will be as many as required to capture all of the data formats currently in use. A format is only included if a comprehensive description of the syntax is available, typically either a formal specification such as an XML Schema (XSD), or comprehensive documentation.

Format and data concepts are formally related:

- **Format** *is_format_of* **Data**

-
- **MUST NOT** include formats which are specific to single tools only, unless heavily used (EDAM formats are generally in common use by multiple tools or public databases)
 - **MUST NOT** include formats for which a comprehensive description (formal specification, or documentation) does not exist
 - **MUST** include a link to the formal specification (*e.g.* an XML Schema (XSD)) or documentation of the format syntax
 - **MUST** have a corresponding concept in the **Data** subontology, *i.e.* the type of data that the format applies to. The exception is formats *e.g.* **TSV** or **RDF/XML** which are generic to any type of data. New data concepts can be added, if needed.
 - **MUST** mention in the definition the type of data the format is used for.
 - **SHOULD NOT** include any narrow synonyms (specialisations are normally handled by adding new sub-concepts, then with their distinct specifications).
 - **SHOULD** record the following attributes, if available:
 - File extension(s) where in common use; these **SHOULD** be given in lower case (unless a specific capitalisation is required) and **MUST NOT** include period ('.'), *e.g.* “txt” not “.txt”. If a matching **filext** record exists, it **SHOULD** be linked (*e.g.* <http://filext.com/file-extension/FASTA>).

- Wikipedia link if exists (*e.g.* http://en.wikipedia.org/wiki/FASTA_format).
- [Media type](#) (MIME type) if standardised.
- Link(s) to example(s) of the format.
- Link(s) to ontologies used in this data format.
- Link(s) (ideally DOI) to article about the format.
- Link to a public source-code repository where this format is developed.
- Link(s) to supported information standard(s).
- Link(s) to organisation that formally governs the format, if exists.

Welcome to the EDAM Developers Guide. It contains best-practice guidelines for the technical processes of EDAM development; modifying EDAM files on GitHub, creation of releases, deprecation of concepts *etc.*

If you're not sure how to do something please ask edam@elixir-dk.org. You'll need to [subscribe](#) to the list first.

6.1 Technical recipes

Note: 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.
-

6.1.1 General guidelines

1. As much as you can, try to make atomic changes and commit them independently. this improves greatly traceability in the long term

2. Make trivial modifications using a text editor if possible, rather than Protege, because the actual modification is not hidden in haystack of Protege reformatting
3. Include an informative message when you commit a change and (ideally) add a description of your modifications in the changelog.
4. Check and double-check your changes: errors can be hard to track and fix later

6.1.2 Adding concepts

Mandatory attributes

When adding new concepts, you **MUST** specify the following:

Attribute	OWL attribute	Note
Concept URI	<code>rdf:about</code>	In the right namespace and with the latest numerical ID.
Primary term	<code>rdfs:label</code>	See Editors Guide .
Definition	<code>oboInOwl:hasDefinition</code>	See Editors Guide .
Parent(s)	<code>rdfs:subClassOf</code>	Immediate parent(s) of the concept (normally one only)."
Version	<code>created_in</code>	Current EDAM dev version, <i>e.g.</i> 1.21.
Type subset	<code>oboInOwl:inSubset</code>	One of concrete or placeholder, see Technical details .
EDAM subset	<code>oboInOwl:inSubset</code>	Always edam.
Branch subset	<code>oboInOwl:inSubset</code>	One of topic, data, format or operation.
Next ID	<code><next_id></code>	Increment the current count by 1.

For **Format** additions you **MUST** also specify:

Attribute	OWL attribute	Note
Type of data	<code><is_format_of></code>	URI of EDAM Data concept.
Specification	<code><documentation></code>	URL of formal (machine-readable) specification. See Editors Guide .
Basic type	<code>rdfs:subClassOf</code>	One of Textual format , Binary format , <i>etc.</i> . See Technical details .
Type of data	<code>rdfs:subClassOf</code>	Some child of Format (by type of data). See Technical details .

For **Identifier** additions you **MUST** also specify:

Attribute	OWL attribute	Note
Type of data	<code><is_identifier_of></code>	URI of EDAM Data concept.
Basic type	<code>rdfs:subClassOf</code>	One of Accession or Name . See Technical details .
Type of data	<code>rdfs:subClassOf</code>	Some child of Identifier (hybrid). See Technical details .

Optional attributes

When adding new concepts, you **SHOULD** specify the following:

Attribute	OWL attribute	Note
Exact synonym	oboInOwl:hasExactSynonym	See Technical details .
Narrow synonym [1]	oboInOwl:hasNarrowSynonym	See Technical details .
Broad synonym [1]	oboInOwl:hasBroadSynonym	See Technical details .
Comment	rdfs:comment	See Editors Guide .
Wikipedia	<documentation>	URL of Wikipedia page.
Usage guideline	<notRecommendedForAnnotation>	Set to `true` for placeholder concepts.

[1] narrowSynonym and broadSynonym **MUST NOT** be specified on EDAM Format concepts.

For **Operation** additions you **MAY** also specify:

Attribute	OWL attribute	Note
Top-level operation	rdfs:subClassOf	One of the Tier 1 operations (see technical docs) <i>unless</i> this already subsumed adequately by the parent.

For **Format** additions you **SHOULD** also specify:

Attribute	OWL attribute	Note
Documentation	<documentation>	URL of documentation about the format.
Publication	<documentation>	DOI of publication about the format.
File extension [1,2]	<file_extension>	File extension without period character), one extension / <file_extension> annotation. Must contain lowercase alphanumeric characters only.
Media type	<media_type>	media type (MIME type) if available.
Example	<example>	Link to example of the format, if available.
Information standard	<information_standard>	Link to relevant information standard which the format supports.
Ontology used	<ontology_used>	Link to an ontology used by this format (one link per <ontology_used> annotation).
Governing organisation	<organisation>	Link to an organisation that formally governs the format, one link link per <organisation> annotation.

[1] File extension values **MUST** be in lowercase. [2] If a file extension is specified, then this **MUST** also be given either as “exact synonyms” or as the concept “preferred label” (term). Exact synonyms **MAY** include variants both with and without full stop (period) character, *e.g.* both `.txt` and `txt`.

For **Identifier** additions you **SHOULD** also specify:

Attribute	OWL attribute	Note
Regex	<regex>	Regular expression pattern for identifier instances.
Documentation	<documentation>	URL of documentation about the identifier.

Hierarchy

The following rules maintain the integrity of the conceptual hierarchy and ensure a consistent level of conceptual granularity. See [Technical details](#) for definition of *concrete* and *placeholder* concepts.

- All subontologies

- leaf nodes **MUST** be concrete concepts
- **Topic:**
 - **MUST** have a path to root of 4 levels deep maximum
 - **MUST NOT** have a path to root exceeding 5 levels deep
- **Operation:**
 - **MUST** ensure placeholders appear in Tiers 1 and 2 (usually) and 3 (rarely - in exceptional cases) only
 - **MUST NOT** chain more than 3 placeholders
 - **MUST NOT** chain more than 3 concrete operations
- **Data:**
 - **MUST NOT** chain more than 2 placeholders
 - **MUST NOT** chain more than 2 concrete data concepts
 - **MUST** ensure placeholders occur in Tier 1 (usually) and 2 (rarely) only
- **Identifier:**
 - **MUST NOT** chain more than 4 placeholders
 - **MUST NOT** chain more than 2 concrete identifiers
 - **MUST** be related (via *is_identifier_of*) to a **Data** concept, but **MUST NOT** duplicate this annotation if it's already stated on an ancestor concept.
 - concrete identifiers **MUST** descend (via *subClassOf* relations) from:
 - * *Accession* or *Name* and
 - * *Identifier (typed)* (or its kids)but **MUST NOT** duplicate these relations if already stated on an ancestor concept.
 - Additionally, concrete identifier re-used for data objects of fundamentally different types (typically served from a single database) **MUST** descend from:
 - * “Identifier (hybrid)” (http://edamontology.org/data_2109) may also be given.
- **Format:**
 - **MUST NOT** chain more than 4 placeholders
 - **MUST** be related (via *is_format_of*) to a **Data** concept, but **SHOULD NOT** duplicate this annotation if it's already stated on an ancestor concept.
 - **MUST** descend (via *subClassOf*) concrete formats from *Textual format*, *Tabular format*, *Binary format*, *XML*, *HTML*, *JSON*, *RDF format* or *YAML*, but you **MUST NOT** duplicate this ancestry in format variants. For example *FASTA-like (text)* is defined as a child of *Textual format*, but the kids of *FASTA-like (text)* format are not.
 - **MUST** descend (via *subClassOf*) concrete formats from *Format (by type of data)* (or its kids), but again, you **MUST NOT** duplicate this ancestry in format variants. For example *FASTA-like (text)* is defined as a child of *Sequence record format* -> *FASTA-like*, but the kids of *FASTA-like (text)* format are not.
 - **MUST NOT** add new placeholder concepts (kids of *Format (by type of data)*) unless there is a corresponding concrete data format descending from it.

If you add a concept which you expect to remain a leaf node, *i.e.* EDAM will not include finer-grained concepts, then - if other well-developed ontologies exist that serve this conceptual niche - you **SHOULD** annotate this junction (see **‘todo <>_’**).

6.1.3 Deprecating concepts

When deprecating concepts, you **MUST** (unless otherwise stated) specify the following:

Attribute	OWL attribute	Note
EDAM version	obsolete_since	Current version <i>e.g. 1.21</i>
Subset	oboInOwl:inSubset	Set this to <code>obsolete</code> (pick the value)
Deprecation flag	owl:deprecated	Type the value of <code>true</code>
Replacement concept [1]	oboInOwl:replacedBy	The alternative ‘replacement’ concept to firmly use. Pick one.
Replacement concept [1]	oboInOwl:consideredReplacedBy	Replacement concept when less certain. Pick one.
Old parent	oldParent	Specify the URI(s) of the erstwhile parent(s) of the now-deprecated concept (using one or more attributes as needed).
Comment [2]	deprecation_comment	Optional comment as to why the concept is deprecated.
New parent	rdfs:subClassOf	Set the parent concept to be <code>ObsoleteClass</code>

[1] One of `replacedBy` or `consideredReplacedBy` **MUST** be specified. [2] `deprecation_comment` is **OPTIONAL**

Also:

1. **MUST** remove all other class annotations (subsets, comments, synonyms *etc.*) and axioms (including parent concepts), apart from `rdfs:definition` (which **MUST** be preserved) and `rdfs:comment` (which **MAY** be preserved).
2. **MUST** refactor all references (*e.g.* `SubClassOf`) to the concept being deprecated from other concepts (you can see these using Protege)
3. **SHOULD** preserve comments and synonyms, as new annotations either in the old parent(s), or the replacement(s) of the deprecated concept, as appropriate.

You **MAY** specify the following on concepts which are candidates for deprecation:

Attribute	OWL attribute	Note
Candidate for deprecation	is_deprecation_candidate	Set this to <code>true</code>

Note: You can see all references to a concept in Protege in the “Class Usage” window; each reference will need updating in turn: in case of very many such references, this can be easier to do globally in a text editor rather than Protege.

6.1.4 Use of Protege

Protege is a nice OWL Editor, but has it’s quirks, so it’s recommended you first get a crash course from the [EDAM Developers](#) before using it. A commercial alternative is [TopBraid Composer](#).

Editing

Important: When editing EDAM using Protege:

- URLs should be entered using the Protege IRI editor.

- general text is entered using the Protege ‘Constant’ editor.
- subsets (`oboInOwl:inSubset` annotation): you must pick (don’t type!) an appropriate value.
- when **saving** the file be sure to use **File...Save as** and select “RDF/XML” format.

Don’t deviate from the above advice. The EDAM CI (and other) systems rely upon EDAM being saved in RDF/XML format, following the patterns specified.

Ensuring logical consistency

Before committing changes, to ensure logical consistency of EDAM, please do the following within Protege:

1. Click *Reasoner->Hermit*
2. Click *Reasoner->Start reasoner* (it may take a few seconds)
3. In the *Entities* tab, select the *Class hierarchy (inferred)* tab
4. Select the *nothing* branch

If nothing (no classes) are shown under the *nothing* branch, then all is well. If one or more classes are shown, then there is a logical inconsistency which must be fixed. You might see lots of classes, but usually the problem is in one or a few classes.

Common problems include:

- classes assigned as a `subClass` of some deprecated concept
- end-point of relations are in the wrong branch, *e.g. class has_topic some operation*. These can easily occur if you use the *Class expression editor* in Protege to define such axioms: this is **NOT** EDAM namespace-aware, and in cases where a concept with the same primary label exists in both classes, can easily pick the wrong one.

The problems are easily fixed within Protege: ask on the [mailing list](#) if you’re not sure how.

Caution: Do not be tempted to click *Reasoner->Synchronise reasoner* between changes: it tends to hang Protege. Instead, use *Reasoner->Stop reasoner* than *Reasoner->Start reasoner*.

6.2 EDAM release process

6.2.1 Modifying GitHub main repo.

EDAM Developers can edit the main repository. The workflow is:

1. Get the “editing token”
 - contact edam-dev@elixir-dk.org and claim the “editing token” after first checking that it is not currently taken :)
 - say briefly what you are doing, why, and about how long it will take
2. Update your local repo with the latest files from the GitHub *main* branch:
`git pull` (or “Synch” from the Desktop client)

If you’ve not already done so, you will first need to clone the repo:

```
git clone https://github.com/edamontology/edamontology.git (or “Clone”  
from the Desktop client)
```

3. Configure Git hooks by running the following from your `edamontology git` directory

```
git config core.hooksPath .githooks
```

Git hooks are scripts defined in <https://github.com/edamontology/edamontology/tree/main/.githooks>. They currently detect and prevent (at pre-commit stage) commits of `EDAM_dev.owl` which are not in RDF/XML format.

4. Make and commit your local changes. You **must** be working with the “dev” version, `EDAM_dev.owl`.

- check your changes and that the OWL file looks good in Protege
- ensure the `next_id` attribute is updated
- ensure that `oboOther:date` is updated to the current GMT/BST before the commit
- add the edited file to the commit

```
git add <filepath>
```

- Commit your local changes, including a concise but complete summary of the major changes:

```
git commit -m ;±commit message here;±
```

5. Push your changes to GitHub (*main* branch):

```
git push origin
```

6. Release the editing token for the other developers:

- contact edam-dev@elixir-dk.org and release the “editing token”
- summarise what you actually did and why

Important: Please provide a **meaningful report** on changes so that we can easily generate the ChangeLog upon next release

- in the Git commit message, including the GitHub issue number of any issues addressed (use `fix #xxx` syntax, see [GitHub docs](#))
 - directly in the [changelog.md](#)
-

6.2.2 Creating a new official EDAM release

EDAM release schedule

We aim to follow a bi-monthly release cycle to this schedule:

1. First Wed of every month
 - EDAM team skype to discuss plans for this month. Announcement (to `edam-announcement`) including short summary of plans, invitation for suggestions.
2. Last Mon of every month
 - Announcement (to `edam-announcement`) saying that release is imminent, invitation for last-minute suggestions.
3. Last Wed of every month
 - Complete the work for the release. Make the release. Ensure it works in BioPortal, OLS, AgroPortal and in `bio.tools`.

4. Last Fri of every month


- Announce the release, including summary of changes.

Note: Releases have been mostly quarterly but more regular (bi-monthly or even monthly) remains the aspiration. Please help out move faster by [getting involved](#).

Process

Before creating a new release, please make sure you have the approval of leader of EDAM-dev, and that the [changelog.md](#) and [changelog-detailed.md](#) files are up-to-date with the changes of the new release. See [Editing the ChangeLog](#) below. Once you're clear to go, do the following:

0. fix any known bugs in EDAM: at the very least, the EDAM build tests should pass as indicated by:

Current status of the 'master' development file: 

1. update your local version of the repository:

`git pull` (or “Synch” in desktop client)

2. assuming you are releasing version $n+1$, n being the current version:

- you initially have `EDAM_dev.owl` in the repository
- make sure to update `oboOther:date` in this file
- copy the file `EDAM_dev.owl` to `releases/EDAM.owl` and `releases/EDAM_n+1.owl`
 - `cp EDAM_dev.owl releases/EDAM.owl`
 - `cp EDAM_dev.owl releases/EDAM_n+1.owl`
 - `git add releases/EDAM_n+1.owl`
- modify the `doap:version` property to **$n+1$** in `releases/EDAM_n+1.owl` and to **$n+2_dev$** in `EDAM_dev.owl`

3. commit and push your changes

- `git commit -a` (or “Commit to main” in the desktop client)
- `git push origin` (or “Synch” in the desktop client)

4. update the [detailed changelog](#) by running [Bubastis](#) to compare the release against the previous version.

5. update the [changelog](#) with a summary of the major changes.

6. create the release on GitHub (use the [_draft a new release_](#) button of the [_releases_](#) tab).

- from the main page of the EDAM repository, click Releases.
- click Draft a new release
- enter the version number *e.g.* 1.24 in the Tag version box
- enter a title *e.g.* EDAM 1.24 release
- check the This is a pre-release box if applicable
- paste an excerpt from `changelog.md` into

7. submit this new release to BioPortal. OLS will pull the file automatically from edamontology.org every night.
8. download the `EDAM.csv` file from BioPortal and copy this to <https://github.com/edamontology/edamontology/tree/main/releases>
9. create a tsv equivalent of `EDAM.csv` (e.g. by hacking in a text editor) and copy the resulting “EDAM.tsv” file to <https://github.com/edamontology/edamontology/tree/main/releases>
10. close GitHub issues labelled *done - staged for release*.
11. create the next milestone tag in GitHub, e.g. “1.25”
12. review any GitHub issues tagged for the release milestone which we’re not acted upon; remove the milestone and (if applicable) tag them with the next milestone tag
13. confirm everything is working in bio.tools by mailing [bio.tools Lead Curator](mailto:bio.tools@elixir-dk.org).
14. let the developers of the EDAM browser know a new release is available by posting [here](#)
15. Update the content of <https://github.com/edamontology/edamontology.org/blob/main/page.html> (add a line linking to the download of the latest release)
16. ensure <http://edamontology.org> is updated
17. announce the new release on Twitter and mailing lists (edam-announce@elixir-dk.org, edam@elixir-dk.org) including thanks and a summary of changes.
18. help applications that implement EDAM to update to the new version.

6.2.3 Editing the ChangeLog

The ChangeLog includes:

1. [changelog](#) - a summary of the major changes and what motivated them
2. [detailed changelog](#) - fine-grained details obtained using [Bubastis](#)

The changelog should include:

1. (as 1st paragraph) an “executive summary” suitable for consumption by technical managers, describing the motivation for major changes, including e.g. requests at recent hackathons, requests via GitHub, strategic directions etc.
2. summary of changes distilled from the output of [Bubastis](#) (see below).
3. summary of GitHub commit messages. **please ensure meaningful commit messages are provided on every commit**

Some hacking of bubastis output is needed to identify (at least):

- number of new concepts
- number of deprecations
- summary of activity, i.e. in which branches was most work focussed ?

6.3 Continuous Integration

Every modification on the ontology pushed to GitHub triggers an automated test in Travis CI. It checks:

- a few rules using the [edamxpathvalidator](#) tool.

- the consistency of the ontology by running the Hermit reasoner automatically.

The Travis-CI website shows you the current status [here](#). The fact that the continuous integration task succeeds does not guarantee there are no remaining bugs, but a failure means that you must take action to correct the problem, either fix it, fix the `edamxpathvalidator` program, or ask the mailing list if you're unsure.

6.4 Modifications in a GitHub fork

GitHub makes it possible for any developer to make modifications in a copy of EDAM and suggest these modifications are included in the original. Please note that we discourage using this mechanism for large modifications made using Protege, because merging OWL files which have been reformatted by Protege is notoriously unreliable (see “Best practices for edition” below).

The workflow is:

- Fork the edamontology repository in your own account.
- Make the modifications you want to suggest for inclusion in EDAM in this forked repository.
- Open pull requests for each modification you make.

Please make sure to:

- Keep your forked repository synchronized with the core repository, to avoid inconsistencies.
- Make sure to follow the “Best practices for edition” below.

EDAM follows a model with four tiers of governance:

1. **EDAM Advisory Group** advises the EDAM Developers on how best to uphold the EDAM principles and achieve its current aims. It includes (and is open to) representatives of institutes and projects that are (or which are seriously considering) adopting EDAM or committing significant resources to its development. Advisory Group **members** have five primary responsibilities:

- Help to set priorities in consultation with the Developers
- Verify whether stated aims are coherent and wise
- Monitor progress and provide feedback
- Help arrange funding for EDAM
- Advocate EDAM

The EDAM Developers will respect advice of the Advisory Group and give progress reports by email. The Advisory Group will be reconstituted (by the Developers) as and when required to ensure membership reflects current activity and interest.

2. **EDAM Developers** are typically funded to develop EDAM. The group is quasi-democratic with a leader (currently Jon Ison) having the final say where necessary (it seldom is). The leader ensures the Advisory Group, and all developers, editors and contributors, are listened to, respected and informed. The leader may be temporarily appointed from the developers as necessary, *e.g.* during holidays. **Developers** must have the intent and some capacity to develop EDAM in the long-term. They have 3 primary responsibilities:

- Agree aims and general good practice to uphold the EDAM principles
- Oversee and approve developments and routine maintenance
- Develop EDAM as bandwidth permits

3. **EDAM Editors** represent the broad life science community, especially scientific experts and end-users. They include anyone who makes significant contributions to EDAM scientific content, by whatever means, but have none of the commitments or responsibilities of the developers. **EDAM Editors** are expected to:

- Actively offer constructive advice based on their practical experience, requirements and expertise

- Advocate EDAM
- Contribute to EDAM as bandwidth permits

Note: The ELIXIR Tools Platform is working to support Thematic Editors to oversee coverage and quality of bio.tools and EDAM in specific thematic areas. The editorships will enable expanding accurate high-standard software annotations in bio.tools to most scientific topics in life science. Please see the [EDAM Editors Guide](#) (instructions for EDAM Editors) and [bio.tools Editors Guide](#) (for a description of the Thematic Editor scheme).

4. **Other contributors** do not have GitHub commit rights but can still make comments, contribute suggestions for new terms and other changes. New [contributors](#) are strongly encouraged.

We very much welcome new members of the team at all levels above. Representatives of projects who plan to adopt EDAM are encouraged to join the EDAM Advisory Group. For further information please see the [documentation](#) or mail edam-dev@elixir-dk.org.

8.1 EDAM Developers

- Jon Ison (DTU, DK) **lead developer**
- Matúš Kalaš (University of Bergen, NO)
- Hervé Ménager (Institut Pasteur, FR)
- Veit Schwämmle (SDU, DK)

8.2 EDAM Editors

Official EDAM Editors are now part of the [Thematic Editors](#) group serving both EDAM and [bio.tools](#).

8.3 EDAM Advisory Group

- Alfonso Valencia (ELIXIR ES)
- Anna-Lena Lamprecht (University of Potsdam, DE)
- Cath Brooksbank (ELIXIR EMBL-EBI)
- Christophe Blanchet (ELIXIR FR)
- Hedi Peterson (ELIXIR EE)
- Heinz Stockinger (ELIXIR CH)
- Inge Jonassen (ELIXIR NO)
- Karel Berka (ELIXIR CZ)
- Michael Crusoe

- Søren Brunak (ELIXIR DK)
- Steven Newhouse (ELIXIR EMBL-EBI)

Note: We hope soon to expand the Advisory Group to include representatives of all the projects and groups which are using, or seriously considering using EDAM, including (non-exhaustive list):

Project	Description	Possible representative
bio.tools	ELIXIR Tools and Data Services Registry.	Hans-Ioan Lenasescu?
Bioschema	Fosters the use of schema.org within the life sciences.	Alasdair Gray?, Giuseppe Profiti?
BISE	Bioimage Informatics Search Engine (uses EDAM-Bioimaging).	Perrine Paul-Gilloteaux?
CWL / CWLProv	Open standard for describing analysis workflows and tools in a way that makes them portable and scalable.	Michael Crusoe
Debian Med	Debian flavour well fit for medical practice and biomedical research.	Steffen Möller?
DSEO	Data Science Education Ontology	Lily Fierro
EMBL EBI (tools)	Tools and data resources from EMBL-EBI	Jonathon Hickford? Rodrigo Lopez?
EMBL EBI (training)	Training resources from EMBL-EBI	Laura Emery?
Galaxy	Open source, web-based platform for data intensive biomedical research.	Björn Gruening? John Chilton?
H3Africa eGenomics Catalogue	Consortium to empower African researchers in genomic sciences.	tbd
Identifiers.org	Resolving system for referencing of data for the scientific community.	Sarala Wimalaratne?
IFB RAINBIO Bioinformatics Cloud Appliances	Bioinformatics cloud appliances registry from the IFB.	Christophe Blanchet
IFB catalogue	Bioinformatics resource catalogue from the IFB.	Jacques Van Helden, Hervé Menager
PROPHETS workflow synthesis	Workflow synthesis using EDAM.	Anna-Lena Lamprecht
TeSS	ELIXIR's training portal; browse and discover life sciences training resources.	Niall Beard?
The flora phenotype ontology (FLOPO)		Robert Hoehndorf
Pathogen-Host Interaction Database (PHI-base)		Robert Hoehndorf
Subject Resource Application Ontology		Allyson Lister
Software ontology		Allyson Lister
BIOLITMAP		Bazaga, A
APE		Anna-Lena Lamprecht

8.4 Contributors

Thanks to the many people who have contributed - if you're not listed below, please let us know!

- Marie Grosjean (IFB, FR)
- Nathalie Conte (EMBL-EBI, UK)
- Victor de la Torre (ELIXIR-ES)
- Ray Fergerson (Stanford University, USA)
- Carole Goble (ELIXIR-UK)
- Simon Jupp (EMBL-EBI, UK)
- Peter Løngreen (CBS-DTU, DK)
- Allyson Lister (Newcastle University, UK)
- Rodrigo Lopez (EMBL-EBI, UK)
- James Malone (EMBL-EBI, UK)
- Julie McMurry (EMBL-EBI, UK)
- Hamish McWilliam (formerly EMBL-EBI, UK)
- Helen Parkinson (EMBL-EBI, UK)
- Steve Pettifer (University of Manchester, UK)
- Kristoffer Rapacki (CBS-DTU, DK)
- Peter Rice (Imperial College, UK)
- Mahmut Uludag (EMBL-EBI, UK)
- Jiří Vondrášek (IOCB AS, CZ)
- Gert Vriend (CMBI, NL)
- Trish Whetzel (University of California, USA)
- David Sehnal (MU, CZ)
- Dmitry Repchevsky (BSC, ES)
- Ivan Mičetić (University of Padova, IT)
- Kristian Davidsen (DTU, DK)
- Laura Emery (EMBL-EBI, UK)
- Lukáš Pravda (MU, CZ)
- Stanislav Geidl (MU, CZ)
- Wouter Touw (CMBI, NL)

8.5 Recent workshops (2014 -)

Thank you to all of the participants of various meetings and workshops organised by ELIXIR, BioMedBridges and others. See the complete list of past and forthcoming [workshops](#).

CHAPTER 9

Hangouts

Monthly informal hangouts to discuss all matters around EDAM development. Meetings are scheduled for the 1st Wed of each month at 10AM UK == 11AM CE(S)T.

If you'd like to attend a hangout, please mail [edam-dev](#).

10.1 ELIXIR Tools & Data Services Registry

The ELIXIR Tools & Data Services Registry is working towards a comprehensive, consistent and maintained catalogue for life science software. All bio.tools entries are annotated using EDAM.

- bio.tools

10.2 EBI Train online

Train online provides free courses on Europe's most widely used data resources, created by experts at EMBL-EBI and collaborating institutes. Training courses are annotated using EDAM Topics.

- <http://www.ebi.ac.uk/training/online/>

10.3 RAINBio registry of bioinformatics cloud appliances

The appliances of the IFB cloud come with preconfigured bioinformatics tools, and both are annotated with terms from the topics dictionary of the EDAM ontology. To help you to find the right tools or appliances for your needs, you can browse the list of appliances and tools below and filter it with any terms.

- <https://biosphere.france-bioinformatique.fr/catalogue/>

10.4 SEQwiki

The SEQanswers wiki is an open catalogue of bioinformatics software tools, non-exclusively focussed on sequencing data analysis. SEQanswers tool wiki uses EDAM for annotation of the listed tools where applicable.

- <http://seqanswers.com/wiki/Software>

10.5 eSysbio

eSysbio is a proof-of-concept prototype workbench for sharing and analysing bioinformatics data using public or private Web services and R scripts. eSysbio used EDAM to annotate and denote the type and format of data items submitted to the system.

- <http://esysbio.org/about>

CHAPTER 11

License

EDAM is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License ([CC BY-SA 4.0](#)).

We recommend, however, that while EDAM is being actively maintained by its authors, substantial derived work, major modifications (especially conceptual and semantic), and re-definitions of concepts and other content (*e.g.* additional constraints on EDAM concepts/OWL classes within `owl:imports` meant with universal validity, that would “close” some desired options of the open-world assumption) are consulted with the EDAM developers beforehand at the time of consideration, and consistent solutions are sought in collaboration.

Frequently asked questions

12.1 Where do I ask questions about EDAM?

You can contact the [mailing list](#). Alternatively (and especially if you think your question is of general interest), you can post your question on [BioStars](#).

12.2 How do I request new terms?

First read the [guidelines for requests](#). Then (usually) you can make your request via [GitHub](#).

12.3 How do I cite EDAM?

See the [citation instructions](#).

12.4 I want to add loads of terms - what do I do?

Please mail the [EDAM developers](#) and we'll advise the best way to proceed.

12.5 Can I join the EDAM team?

Yes please! See [Joining the team](#).