# Table of Contents

Colophon .................................................................................................................. 1  
Suggested citation .................................................................................................... 1  
Contributors ............................................................................................................ 1  
Licence ...................................................................................................................... 1  
Persistent URI ......................................................................................................... 1  
Document control .................................................................................................... 1  
Cover image .............................................................................................................. 1  
Background .............................................................................................................. 1  
  Current documents .................................................................................................. 2  
  1. Community peer-review process .................................................................... 3  
  2. Guidelines for document authors .................................................................... 4  
  3. Translations ....................................................................................................... 5  
  4. ‘Decommissioning’ old documents .................................................................. 5  
  5. Documentation Steering Panel ......................................................................... 6  
     5.1. Structure and operations ........................................................................... 6  
     5.2. Responsibilities ......................................................................................... 6  
     5.3. Panel members ........................................................................................ 6  
  6. Technical guidance ............................................................................................ 7  
     6.1. For authors ............................................................................................... 7  
     6.2. For editors ............................................................................................... 10  
     6.3. The documentation system software ........................................................ 12  
  7. Information for GBIF developers ....................................................................... 13  
     7.1. New documents ......................................................................................... 13
Background

GBIF—the Global Biodiversity Information Facility—has long produced technical documentation on a range of topics relating to biodiversity informatics and open biodiversity data with the aim of supporting a global community of practice.

With the help of the team from VertNet, the GBIF Secretariat started coordinating the development of such documentation as part of its 2019 work programme, with the aim of engaging GBIF communities of practice to work with subject-matter experts commissioned to create and update under the guidance of an editorial panel.

This goal of this approach is to provide consistent, reliable, reusable and versioned materials that can be easily updated, instilling community trust in the documentation and fostering its wider adoption and use.
Current documents


- Translation (status): French (ongoing), Spanish (ongoing)


- Translation (status): Spanish (ongoing)


- Translation (status): Spanish (ongoing)


- Translation (status): Spanish (ongoing)


- Translation (status): English (ongoing), French (ongoing)


- Translation (status): French (ongoing), Spanish (ongoing)

Rui Figueira, Pedro Beja, Cristina Villaverde, Miguel Vega, Katia Cezón, Tainan Messina, Anne-Sophie Archambeau, Rukaya Johaadien, Dag Endresen & Dairo Escobar (2020) *Guidance for private companies to become data publishers through GBIF: Template document to support the internal authorization process to become a GBIF publisher*. Copenhagen: GBIF Secretariat. https://doi.org/10.35035/doc-b8hq-me03
1. Community peer-review process

Community peer-review may be just a single step in GBIF's digital documentation workflow, but it provides an important opportunity for members of the GBIF community of practice—the intended users and beneficiaries of these documents—to guide the documents' development by offering direct input and feedback. While this process is first and foremost intended to ensure the quality of the documentation, it also serves as a mechanism for fostering community discussion and collaboration.
The process starts from the premise that authors and reviewers are part of the same community. The fact that their identities are not concealed at any point during the process, reviewers and authors should be encouraged toward open, honest and collegial exchanges, with a focus on constructive criticism even where difference of opinion exist. The focus of reviewers should be to help authors to improve their work in ways that benefit the broader biodiversity informatics community. Community members are responsible for ensuring that their actions encourage a “safe, hospitable, and productive environment” that is “professional, respectful and harassment-free for all participating,” in adherence with the GBIF Code of Conduct.

Each document’s source text is freely and openly available and maintained in a public GitHub repository, or “repo”. The use of GitHub enables reviewers and users to raise issues and track their resolution. Reviewers and users can offer comments, suggestions and corrections at any stage of the document’s life cycle, making it easier to make corrections to current versions and update future ones while ensuring community access to accurate, well-maintained guidance and information.

Staff from the GBIF Secretariat commits to two operational principles to ensure the transparency and effectiveness of the community review process:

1. Individual contributions by community members will be properly credited and acknowledged
2. Open issues will be resolved in timely fashion, either by the authors or by Secretariat staff, in agreement with the authors

Learn how to help improve these documents as a community peer-reviewer

2. Guidelines for document authors

We code documents in this system using a lightweight code called AsciiDoc. We apply United Nations editorial style conventions and spelling (tl;dr: British English with –ize and –yse endings).

We use AsciiDoc because it means that:

• Authors need not worry about maintaining formatting.
• Changes to both the document’s content and structure can be tracked and managed using general tools.
• The same source files can produce multiple outputs automatically, such as HTML and (as a convenience) PDF.
• The management and tracking translations is easier and more efficient.

While this system does require learning some new tools, the ones we have chosen are widely used in the publishing, software development and translation communities. AsciiDoc is also relatively simple to work with and understand. If you’ve written or edited a Wikipedia article, you’ll have no problem with this format.

We’ve pulled out details for a few of the more common uses of AsciiDoc markup in the technical guidelines below to give you a flavour of it. However, two resources provide more detailed guidance:

• The AsciiDoc Writer’s Guide, which provides a ‘gentle introduction’ to the format
• The AsciiDoctor User Guide, which provides a comprehensive reference on how to work with this language.

If you have questions, comments or concerns—either specifically about AsciiDoc or about the overall use and approach of this format—please email us at communication@gbif.org.

3. Translations

The GBIF network hosts a vibrant community of volunteer translators who work actively to reduce linguistic barriers to free and open biodiversity data. We also commission commercial translation of our materials when the Secretariat views such efforts as strategically important and/or our existing language communities lack the capacity to complete them.

We support translations based on the expressed needs and interests of our language communities. As such, we welcome efforts to extend usefulness and reach of our documentation through translation. If you wish to volunteer or to voice your support for making specific titles available in translation, please email us at communication@gbif.org. We will do what we can gauge interest from others in the community and provide coordination support.

If you’re interested in the details of how our documentation system implements translations, see the relevant section in the technical guidance below.

4. ‘Decommissioning’ old documents

As a matter of practice, the Secretariat will ‘decommission’ and remove earlier versions of documents from GBIF through the following series of steps:

1. Register a GBIF DOI via DataCite for the previous version of the document (provided that one does not already exist)
2. Produce an archival standard version (PDF/A) of the document—or documents, if translations are available
3. Deposit the file(s) in Zenodo with the assigned DOI
4. Update the DataCite metadata to resolve the DOI to the new Zenodo deposit
5. Include a reference to the earlier version in the current document’s metadata on GBIF.org (e.g. https://www.gbif.org/document/80925)

This approach achieves several key goals:

• Previous versions will be permanently discoverable using a persistent identifier
• GBIF will no longer have to manage either the old file or its URL (or, as is more often the case, URLs, plural)
• Users searching on GBIF.org will retrieve only the current documents, which then reference older versions
5. Documentation Steering Panel

The Steering Panel consists of a volunteer group of experts that, in coordination with GBIF Secretariat staff, provides oversight and guidance for the selection and community-review of GBIF documentation.

5.1. Structure and operations

The panel consists of 8-10 members comprised of individuals from GBIF regions and the Secretariat staff. The panel meets (no more than quarterly) to discuss existing documentation and to provide recommendations to the GBIF Secretariat on commissioning high-priority guidance from subject-matter experts.

5.2. Responsibilities

1. Assist with setting annual priorities for documentation needs and make recommendations for calls for documentation as and when appropriate.

2. Consult widely with other experts, institutions, initiatives and projects within the biodiversity informatics community at-large when considering updates to and for new documentation.

3. Review and make recommendations regarding the documentation system for future sustainability.

4. Participate in the vetting process to ensure that commissioned documentation is of high quality and serves the intended audiences.

5.3. Panel members

- **Chair**: Sharon Grant, North America
- **Vice-chair**: Patricia Mergen, Europe and Central Asia
- Pierre Radji, Africa
- Maofang Luo, Asia
- Paula Zermoglio, Latin American and the Caribbean
- Chantal Huijbers, Oceania
- Andrea Hahn, GBIF Secretariat
- Dmitry Schigel, GBIF Secretariat

*GBIF Secretariat staff liaisons*

- Kyle Copas
- Laura Anne Russell
6. Technical guidance

6.1. For authors

This section provides some basic technical orientation for authors commissioned to write documents. Both the AsciiDoc Writer’s Guide and AsciiDoctor User Guide provides much more detailed and comprehensive references for how to work with this lightweight markup format.

Structuring the document

All documents whose primary language is English start from the file index.en.adoc. Using the include directive allows a single document to be spread across multiple files. This makes editing (especially collaborative editing) easier, helps translators, and simplifies reordering sections of a document.

Except for the primary file being called index.en.adoc, there are no hard restrictions on how a document must be structured. It is probably easiest for editors to structure documents with number-prefixed filenames, preferably with large intervals to allow new sections to be inserted.

![File structure example]

1 This file was presumably added later, between 200 and 300.

See the section on translating documents when adding, changing or deleting document files.

Writing in AsciiDoctor

AsciiDoctor is a text document format for writing (among other things) books, ebooks, and documentation. It is similar to wiki markup — if you can write a Wikipedia article, then you'll have no problem with AsciiDoctor.

AsciiDoctor User Guide


Here are the most common parts of AsciiDoctor markup:

Text

Regular paragraph text does not need any special markup in AsciiDoctor. Just add a blank line both above and below each paragraph, and the first word in the paragraph should not have a space before it. Here are some example paragraphs in AsciiDoctor:
This is an example paragraph written in AsciiDoctor. See, it's just plain text; no special markup necessary! Do make sure there aren't spaces or manual indentations at the beginning of your paragraph text.

This is a second example paragraph in AsciiDoctor. Note that there's a line break and a blank line between paragraphs.

**Chapters and headings**

The top of each chapter file should begin with a chapter title preceded by two equals signs. It's good practice to always include a unique ID string above the chapter title, surrounded in double brackets, for example:

```
[[unique_chapter_id]]
== Chapter Title
Chapter text begins here.
```

The unique ID string is used to link directly to a chapter or section, such as this link to this section. Readers can also link directly to a section, by using the § link that appears when the mouse hovers over a chapter heading.

**Top-level heading**

Within a chapter, the first and highest heading level uses three equals signs:

```
=== Top-Level Heading
```

Lower-level headings continue with additional = signs.

*Continue reading about section headings in the AsciiDoctor User Guide.*

**Inline Markup**

Here are some standard typographical conventions with explanations of how they're commonly used:

* _Italic_ One underscore character on either side of text marks it as italics in AsciiDoctor.

*Bold* **Bolded text** is used to emphasize a word or phrase. The AsciiDoctor markup is one asterisk on either side of the text to be bolded.

`Constant Width` Constant width, or monospaced, text is used for code, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords. The AsciiDoctor markup is one grave accent sign on either side of the text to monospaced.

Hyperlinks: For hyperlinks to external sources, just add the full URL string followed by brackets
containing the text you’d like to appear with the URL. The bracketed text will become a clickable link in web versions. In print versions, it will appear in the text, followed by the actual URL in parenthesis.

The markup looks like this:

Visit https://www.gbif.org/[GBIF.org].

Continue reading about text formatting in the AsciiDoctor User Guide.

Admonitions

AsciiDoctor allows authors to call out supplemental admonitions in the form of notes, tips, warnings and cautions.

For a note, the markup looks like this:

[NOTE]
=====
Past trends are no guarantee of future performance.
=====

And here’s how it renders:

Past trends are no guarantee of future performance.

There is also a short form, which is appropriate for a single sentence:

NOTE: Past trends are no guarantee of future performance

Continue reading about admonitions and other block formatting in the AsciiDoctor User Guide. The guide also covers other formatting, such as bulleted or numbered lists, tables and images.

6.1.1. GBIF extensions to AsciiDoctor

The document system recognizes some small additions to standard AsciiDoctor markup.

Terms

Terms, including Darwin Core terms, can be shown in a special style and with a link to the definition as `decimalLatitude` or `dwc:eventDate`.

term:dwc[decimalLatitude] or term:dwc[dwc:eventDate]
Table cell wrapping

By applying the role `break-all`, the contents of a table cell will break (wrap) at any position, rather than only between words.

| DNA sequence example | TCTATCCTCAATTAGGTCATAAT TCACCATCAGTAGATTGATAATT TTCTCATATTGAGGTTGATCATCAATTAGGATCAATTATTATTATTGTAACATTTTTAAATATACCATACAAAAACTCATTCATTAAACTTTTTTACATTATTTTCACTGATCAGTCTCTTCTTACAGCAATTCTCTTTATTA TCTATTA | The markup is `[break-all]#TCTA...ATTA#`

Without it, the DNA sequence would stretch the table cell beyond the width of the page.

### 6.1.2. Outstanding issues

- Demonstrate embedding an image, and alternative (translated) images ([doc-effective-nodes-guidance](#) has this)
- Apply a custom style to the document ([doc-effective-nodes-guidance](#) also has this)
- Document a release process, possibly involving assigning DOIs.

### 6.2. For editors

#### 6.2.1. Document “source code”

The plain text files and other assets (images, data tables) that form each document comprises the *source code*.

These source files are stored in a *Git repository*, which (for GBIF) is managed by a commercial service, *GitHub*.

The source code for this document is stored at [https://github.com/gbif/doc-documentation-guidelines/](https://github.com/gbif/doc-documentation-guidelines/), the source code for this part of the document can be seen here.

Contributors can edit the source code either in a web browser using the GitHub interface or on a computer (including when offline) using Git. They may also submit issues that comment or flag problems for others to address, including outdated information, broken links, mispellings and the like.

Many tutorials for using both Git and Github are available on the web.

#### 6.2.2. Document versions

Some documents are published as multiple versions. This is done using *branches* in Git: the name of the branch, such as `1.0` or `2019`, is the identifier for the version. This allows for edits to old versions, such as updating a link or correcting a syntax error in the document.
The version (branch name) is used as part of the URL for the document, e.g. 
https://docs.gbif.org/effective-nodes-guidance/1.0/. This allows for multiple versions to be retained on the webserver.

6.2.3. Translated documents

The translation system uses .po “Portable Object” files, which are commonly used for translating software and websites.

1. A file po4a.conf needs to exist, as shown in Translation setup (po4a). Each *.en.adoc file needs an entry in po4a.conf:

   [type:asciidoc] 100.en.adoc $lang:100.$lang.adoc

The build system will warn if any *.en.adoc files are not present in po4a.conf. (This is why the README.adoc and LICENSE.adoc files, not part of the document, do not include .en in their filenames.)

   ◦ Whenever the document text is changed, the build server will update the translation template file translations/index.pot with the source (English) text.
   ◦ Crowdin will detect the change to translations/index.pot and notify translators.
   ◦ As translators add translations to the text, Crowdin will make a pull request on the repository. This should be merged.
   ◦ The build server will then rebuild the document with the translated text.

Alternatives to Crowdin

It is also possible to translate documents without Crowdin, using desktop tools instead. The translators then need to use Git/GitHub. These additional steps are needed:

1. For a new language, copy the generated index.pot (Portable Object Template) file to the new file xx.po, where xx is the language code. For example this would be da.po for a Danish translation.

2. To update a translation, open the xx.po file in a po-file editor and choose the option to “Update from POT file” or similar.

3. Use a po-file editor to make the translations. Examples are Poedit (software) or poeditor (website).

4. Use Git/GitHub to replace the old translation file with your updated translation file.

5. Push the changes, and the build server will rebuild the document

It is not recommended to use both methods on the same document. If translations conflict they would not be lost, but the resulting mess can be confusing to sort out using Git.
6.2.4. Publishing a document

Here, publishing a document means building the document for docs.gbif.org, rather than the test system docs.gbif-uat.org.

To publish a document, go to the GitHub repository in a web browser.

1. If required, review and merge any translation pull requests.
2. Check the most recent output from the document build in Jenkins. This is easily accessed using the "Build Status" button on the repository. Check for
   - Incorrect spelling
   - Warnings about broken crossreferences
   - Warnings about incomplete translation
3. Review the document on https://docs.gbif-uat.org/, including the PDF.
4. Use the GitHub interface to make a release.

6.3. The documentation system software

The documents combine several small Linux tools:

- Git, for source control,
- AsciiDoctor, chosen with essentially the same reasoning as the KiCad documentation authors (and following their approach to translation),
- GNU Aspell, for spell checking,
- po4a, for translations,
- GBIF’s Jenkins server, for document compilation,
- Docker, to ensure consistent builds,
- Apache, to serve the finished documents.

The result is mostly contained in a Docker container, with some integration in the Jenkins build job.

6.3.1. Generating the document

The source .adoc files in the repository are converted into the finished HTML and PDF documents using the AsciiDoctor tool. Every time a change is made to the repository, the GBIF build server is notified. It retrieves the document source code, generates the document (in HTML and PDF, and in all available languages), then copies the formatted documents to a webserver.

A log file of recent builds is kept by the build server. If there is a syntax error preventing the document from being generated, you may need to inspect the log file to see what the problem is. The log file also contains a list of possible spelling errors.
6.3.2. Local document build

If you are familiar with software development tools you can build a document on your own computer — this is useful for previewing changes. You will first need to setup Docker. Then, open a terminal window and navigate using the cd command to the top-level directory of your document — for this document, it would be doc-documentation-guidelines. You can then build the HTML document with this command:

```
docker run --rm -it --user $(id -u):$(id -g) -v $PWD:/documents/ gbif/asciidoctor-toolkit
```

Assuming all is well, the resulting documents are in subdirectories coded by language (such as en), including both HTML and PDF files. The output from the command should provide clues if there are problems.

You can also add continuous to the end of this command. This will rebuild the document every time it is changed.

7. Information for GBIF developers

This section is technical information for GBIF software developers maintaining the system that powers these documents.

7.1. New documents

To make a new document:

- Create a new repository using the doc-template template repository (“Use this template”), with a name beginning with doc-, or course- for a training course.
- Edit the README.adoc to update the links, license, DOI etc.
- Set the branch name appropriately (1.0), if published versions of the document should be retained
- Add a new job to Jenkins,
- If required, create a po4a.conf file and add the document to Crowdin.

7.1.1. Jenkins setup

- Create a new job, based on:
  - the existing doc-template job, for unversioned documents
  - the existing doc-test-document job, for versioned documents
  You need to change the Git repository paths (“Source Code Management” section)
- Change the Authentication Token to something new (“Build Triggers” section)
• Within GitHub, set up a new webhook with the path:


° The secret text does not matter
° Select the individual events Pushes and Releases

**Full Jenkins configuration**

These things will have been copied across from the existing build:

• Discard old builds: 15
• GitHub project
• A payload parameter to receive information from GitHub.
• **Source Code Management**: Under advanced Git settings, set the branches to build to origin/* and Check out to specific local branch to **. This supports versioned documents, and updating the translation index.
• A build script, either VERSIONED=true /usr/local/bin/document-build-deploy or just /usr/local/bin/document-build-deploy.
• Git Publisher post-build action: to merge changes to the translation index.
• Set GitHub commit status (so users can see if they have committed invalid syntax).

### 7.1.2. Translation setup (po4a)

Do this before setting up Crowdin

• Create a po4a.conf file, based on this template:

```bash
# This is the translation configuration file.
#
# Any new file that requires translation must be added

[po_directory] translations
[options] opt:"-M utf-8 -A utf-8 -L utf-8 -k 0"

[type:asciidoc] index.en.adoc $lang:index.$lang.adoc
add_$lang:translations/$lang.add
[type:asciidoc] 100.en.adoc $lang:100.$lang.adoc
...
```

(This should be automated at some point.)

• Push the change. The build should generate a translations/index.pot file, the translation index.)
7.1.3. Crowdin setup

- First ensure appropriate version branches are set up, and the translation (po4a) set up.
- Add the gbif-crowdin GitHub user to the project, with “Admin” rights
- Use a private browser tab to log in to Crowdin, select the project, and add a new GitHub integration (GitHub authentication will be required).
  - Select the repository
  - Select the branch
  - Change the “Service Branch Name” to translation_*branchname* (thus avoiding the awkward abbreviation “i18n”)
  - Set the Branch Configuration:
    - Set the source to /translations/index.pot
    - Set the translation to /translations/%two_letters_code%.po
- Save all this.