

This page is aimed at any developers or coders interesting in understanding or extending the new Sequence Input/Output interface for BioPython, [SeqIO](#).

The code has now been checked into [CVS](#). Related [Bug 2059](#) has been resolved.

The code is already available in BioPython 1.43.

Contents

- [1 Reading new file formats](#)
- [2 Writing new file formats](#)
- [3 Possible additional formats](#)
 - ◆ [3.1 KEGG format](#)
 - ◆ [3.2 MEME format](#)
 - ◆ [3.3 BLAST results](#)
 - ◆ [3.4 COMPASS pairwise alignment format](#)

Reading new file formats

Note: The details are still subject to change

To add support for reading a new file format, you must implement an iterator that expects a just file handle and returns SeqRecord objects. You may do this using:

- An iterator class subclassing something from `Bio.SeqIO.Interfaces`
- A generator function (using the `yield` keyword; suitable for simple formats)
- An ordinary function which returns an iterator. For example, you could build a list of SeqRecords and then turn it into an iterator using the `iter()` function.

You may accept additional *optional* arguments (an alphabet for example). However there *must* be one and only one required argument (the input file handle).

What you use as the SeqRecord's id, name and description will depend on the file format. Ideally you would use the accession number for the id. This id should also be unique for each record (unless the records in the file are in themselves ambiguous).

When storing any annotations in the record's annotations dictionary follow the defacto standard laid down by the GenBank parser... I should try and document this more.

If the supplied file seems to be invalid, raise a ValueError exception.

Finally, the new format must be added to the relevant dictionary mapping in `Bio/SeqIO/__init__.py` so that the `Bio.SeqIO.parse()` and `Bio.SeqIO.read()` functions are aware of it.

Writing new file formats

Note: The details are still subject to change

To add support for writing a new file format you should write a sub class of one of the writer objects in `Bio.SeqIO.Interfaces`

Then, the new format must be added to the relevant dictionary mappings in `Bio/SeqIO/__init__.py` so that the `Bio.SeqIO.write()` function is aware of your code.

If the supplied records cannot be written to this file format, raise a `ValueError` exception. Where appropriate, please use the following wording:

```
raise ValueError("Must have at least one sequence")
raise ValueError("Sequences must all be the same length")
raise ValueError("Duplicate record identifier: %s" % ...)
...
```

ToDo - Defined standard exceptions in `Bio.SeqIO` itself?

Possible additional formats

There are existing parsers in BioPython for the following file formats, which could be integrated into `Bio.SeqIO` or `Bio.AlignIO` if appropriate.

KEGG format

Can `Bio.KEGG` parse files in [KEGG format](#)?

MEME format

`Bio.MEME` has a parser for this file format, which at first glance looks like it could be treated like an alignment format.

<http://meme.sdsc.edu>

BLAST results

Pairwise alignments from the BLAST suite could be turned into a pairwise `Alignment` object with `Bio.AlignIO`. Is this useful? Sample code on [Bug 2560](#)

COMPASS pairwise alignment format

SeqIO_dev

Bio.Compass can parse the pairwise alignments from COMPASS. The output is similar to BLAST in many ways. Again, is getting the results as SeqRecord or pairwise alignment objects useful?