# C API Stability

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 1)

Unknown directive type "highlight".

```
.. highlight:: c
```

Python's C API is covered by the Backwards Compatibility Policy, PEP 387. While the C API will change with every minor release (e.g. from 3.9 to 3.10), most changes will be source-compatible, typically by only adding new API. Changing existing API or removing API is only done after a deprecation period or to fix serious issues.

CPython's Application Binary Interface (ABI) is forward- and backwards-compatible across a minor release (if these are compiled the same way; see :ref:`stable-abi-platform` below). So, code compiled for Python 3.10.0 will work on 3.10.8 and vice versa, but will need to be compiled separately for 3.9.x and 3.10.x.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 15); *backlink*

Unknown interpreted text role "ref".

Names prefixed by an underscore, such as `_Py_InternalState`, are private API that can change without notice even in patch releases.

## Stable Application Binary Interface

Python 3.2 introduced the *Limited API*, a subset of Python's C API. Extensions that only use the Limited API can be compiled once and work with multiple versions of Python. Contents of the Limited API are :ref:`listed below <stable-abi-list>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 28); *backlink*

Unknown interpreted text role "ref".

To enable this, Python provides a *Stable ABI*: a set of symbols that will remain compatible across Python 3.x versions. The Stable ABI contains symbols exposed in the Limited API, but also other ones – for example, functions necessary to support older versions of the Limited API.

(For simplicity, this document talks about *extensions*, but the Limited API and Stable ABI work the same way for all uses of the API – for example, embedding Python.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 42)

Unknown directive type "c:macro".

```
.. c:macro:: Py_LIMITED_API

   Define this macro before including ``Python.h`` to opt in to only use
   the Limited API, and to select the Limited API version.

   Define ``Py_LIMITED_API`` to the value of :c:data:`PY_VERSION_HEX`
   corresponding to the lowest Python version your extension supports.
   The extension will work without recompilation with all Python 3 releases
   from the specified one onward, and can use Limited API introduced up to that
   version.

   Rather than using the ``PY_VERSION_HEX`` macro directly, hardcode a minimum
   minor version (e.g. ``0x030A0000`` for Python 3.10) for stability when
   compiling with future Python versions.

   You can also define ``Py_LIMITED_API`` to ``3``. This works the same as
   ``0x03020000`` (Python 3.2, the version that introduced Limited API).
```

On Windows, extensions that use the Stable ABI should be linked against `python3.dll` rather than a version-specific library such as `python39.dll`.

On some platforms, Python will look for and load shared library files named with the `abi3` tag (e.g. `mymodule.abi3.so`). It does not check if such extensions conform to a Stable ABI. The user (or their packaging tools) need to ensure that, for example, extensions built with the 3.10+ Limited API are not installed for lower versions of Python.

All functions in the Stable ABI are present as functions in Python's shared library, not solely as macros. This makes them usable from languages that don't use the C preprocessor.

### Limited API Scope and Performance

The goal for the Limited API is to allow everything that is possible with the full C API, but possibly with a performance penalty.

For example, while :c:func:`PyList_GetItem` is available, its "unsafe" macro variant :c:func:`PyList_GET_ITEM` is not. The macro can be faster because it can rely on version-specific implementation details of the list object.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 82); *backlink***
>
> Unknown interpreted text role "c:func".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 82); *backlink***
>
> Unknown interpreted text role "c:func".

Without `Py_LIMITED_API` defined, some C API functions are inlined or replaced by macros. Defining `Py_LIMITED_API` disables this inlining, allowing stability as Python's data structures are improved, but possibly reducing performance.

By leaving out the `Py_LIMITED_API` definition, it is possible to compile a Limited API extension with a version-specific ABI. This can improve performance for that Python version, but will limit compatibility. Compiling with `Py_LIMITED_API` will then yield an extension that can be distributed where a version-specific one is not available – for example, for prereleases of an upcoming Python version.

### Limited API Caveats

Note that compiling with `Py_LIMITED_API` is *not* a complete guarantee that code conforms to the Limited API or the Stable ABI. `Py_LIMITED_API` only covers definitions, but an API also includes other issues, such as expected semantics.

One issue that `Py_LIMITED_API` does not guard against is calling a function with arguments that are invalid in a lower Python version. For example, consider a function that starts accepting `NULL` for an argument. In Python 3.9, `NULL` now selects a default behavior, but in Python 3.8, the argument will be used directly, causing a `NULL` dereference and crash. A similar argument works for fields of structs.

Another issue is that some struct fields are currently not hidden when `Py_LIMITED_API` is defined, even though they're part of the Limited API.

For these reasons, we recommend testing an extension with *all* minor Python versions it supports, and preferably to build with the *lowest* such version.

We also recommend reviewing documentation of all used API to check if it is explicitly part of the Limited API. Even with `Py_LIMITED_API` defined, a few private declarations are exposed for technical reasons (or even unintentionally, as bugs).

Also note that the Limited API is not necessarily stable: compiling with `Py_LIMITED_API` with Python 3.8 means that the extension will run with Python 3.12, but it will not necessarily *compile* with Python 3.12. In particular, parts of the Limited API may be deprecated and removed, provided that the Stable ABI stays stable.

## Platform Considerations

ABI stability depends not only on Python, but also on the compiler used, lower-level libraries and compiler options. For the purposes of the Stable ABI, these details define a "platform". They usually depend on the OS type and processor architecture

It is the responsibility of each particular distributor of Python to ensure that all Python versions on a particular platform are built in a way that does not break the Stable ABI. This is the case with Windows and macOS releases from `python.org` and many third-party distributors.

## Contents of Limited API

Currently, the Limited API includes the following items:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]stable.rst, line 158)**

Unknown directive type "limited-api-list".

```
.. limited-api-list::
```