# Call Protocol

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

Unknown directive type "highlight".

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

CPython supports two different calling protocols: *tp_call* and vectorcall.

## The *tp_call* Protocol

Instances of classes that set :c:member:`~PyTypeObject.tp_call` are callable. The signature of the slot is:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 14); *backlink*

Unknown interpreted text role "c:member".

```
PyObject *tp_call(PyObject *callable, PyObject *args, PyObject *kwargs);
```

A call is made using a tuple for the positional arguments and a dict for the keyword arguments, similarly to `callable(*args, **kwargs)` in Python code. *args* must be non-NULL (use an empty tuple if there are no arguments) but *kwargs* may be *NULL* if there are no keyword arguments.

This convention is not only used by *tp_call*: :c:member:`~PyTypeObject.tp_new` and :c:member:`~PyTypeObject.tp_init` also pass arguments this way.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 25); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 25); *backlink*

Unknown interpreted text role "c:member".

To call an object, use :c:func:`PyObject_Call` or another :ref:`call API <capi-call>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 29); *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)call.rst, line 29); *backlink*

Unknown interpreted text role "ref".

## The Vectorcall Protocol

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 38)

Unknown directive type "versionadded".

```
.. versionadded:: 3.9
```

The vectorcall protocol was introduced in PEP 590 as an additional protocol for making calls more efficient.

As rule of thumb, CPython will prefer the vectorcall for internal calls if the callable supports it. However, this is not a hard rule. Additionally, some third-party extensions use *tp_call* directly (rather than using :c:func:`PyObject_Call`). Therefore, a class supporting vectorcall must also implement :c:member:`~PyTypeObject.tp_call`. Moreover, the callable must behave the same regardless of which protocol is used. The recommended way to achieve this is by setting :c:member:`~PyTypeObject.tp_call` to :c:func:`PyVectorcall_Call`. This bears repeating:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 43); *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)call.rst, line 43); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 43); *backlink*

Unknown interpreted text role "c:member".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 43); *backlink*

Unknown interpreted text role "c:func".

**Warning**

A class supporting vectorcall **must** also implement :c:member:`~PyTypeObject.tp_call` with the same semantics.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 57); *backlink*

Unknown interpreted text role "c:member".

A class should not implement vectorcall if that would be slower than *tp_call*. For example, if the callee needs to convert the arguments to an args tuple and kwargs dict anyway, then there is no point in implementing vectorcall.

Classes can implement the vectorcall protocol by enabling the :const:`Py_TPFLAGS_HAVE_VECTORCALL` flag and setting :c:member:`~PyTypeObject.tp_vectorcall_offset` to the offset inside the object structure where a *vectorcallfunc* appears. This is a pointer to a function with the following signature:

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 65); *backlink*

Unknown interpreted text role "const".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 65); *backlink*

Unknown interpreted text role "c:member".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 71)

Unknown directive type "c:type".

```
.. c:type:: PyObject *(*vectorcallfunc)(PyObject *callable, PyObject *const *args, size_t nargsf, PyObject *kwn
```

- *callable* is the object being called.
- *args* is a C array consisting of the positional arguments followed by the
  values of the keyword arguments. This can be *NULL* if there are no arguments.
- *nargsf* is the number of positional arguments plus possibly the

  :const:`PY_VECTORCALL_ARGUMENTS_OFFSET` flag. To get the actual number of positional arguments from *nargsf*, use :c:func:`PyVectorcall_NARGS`.

  **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 78); *backlink*

  Unknown interpreted text role "const".

  **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 78); *backlink*

  Unknown interpreted text role "c:func".

- *kwnames* is a tuple containing the names of the keyword arguments;
  in other words, the keys of the kwargs dict. These names must be strings (instances of `str` or a subclass) and they must be unique. If there are no keyword arguments, then *kwnames* can instead be *NULL*.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 87)

Unknown directive type "c:macro".

```
.. c:macro:: PY_VECTORCALL_ARGUMENTS_OFFSET

   If this flag is set in a vectorcall *nargsf* argument, the callee is allowed
   to temporarily change ``args[-1]``. In other words, *args* points to
   argument 1 (not 0) in the allocated vector.
   The callee must restore the value of ``args[-1]`` before returning.

   For :c:func:`PyObject_VectorcallMethod`, this flag means instead that
   ``args[0]`` may be changed.

   Whenever they can do so cheaply (without additional allocation), callers
   are encouraged to use :const:`PY_VECTORCALL_ARGUMENTS_OFFSET`.
   Doing so will allow callables such as bound methods to make their onward
   calls (which include a prepended *self* argument) very efficiently.
```

To call an object that implements vectorcall, use a :ref:`call API <capi-call>` function as with any other callable. :c:func:`PyObject_Vectorcall` will usually be most efficient.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 102); backlink

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 102); backlink

Unknown interpreted text role "c:func".

**Note**

In CPython 3.8, the vectorcall API and related functions were available provisionally under names with a leading underscore: `_PyObject_Vectorcall`, `_Py_TPFLAGS_HAVE_VECTORCALL`, `_PyObject_VectorcallMethod`, `_PyVectorcall_Function`, `_PyObject_CallOneArg`, `_PyObject_CallMethodNoArgs`, `_PyObject_CallMethodOneArg`. Additionally, `PyObject_VectorcallDict` was available as `_PyObject_FastCallDict`. The old names are still defined as aliases of the new, non-underscored names.

## Recursion Control

When using *tp_call*, callees do not need to worry about :ref:`recursion <recursion>`: CPython uses :c:func:`Py_EnterRecursiveCall` and :c:func:`Py_LeaveRecursiveCall` for calls made using *tp_call*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 123); backlink

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 123); 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)call.rst, line 123); backlink

Unknown interpreted text role "c:func".

For efficiency, this is not the case for calls done using vectorcall: the callee should use *Py_EnterRecursiveCall* and *Py_LeaveRecursiveCall* if needed.

## Vectorcall Support API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 136)

Unknown directive type "c:function".

```
.. c:function:: Py_ssize_t PyVectorcall_NARGS(size_t nargsf)

   Given a vectorcall *nargsf* argument, return the actual number of
   arguments.
   Currently equivalent to::

      (Py_ssize_t)(nargsf & ~PY_VECTORCALL_ARGUMENTS_OFFSET)

   However, the function ``PyVectorcall_NARGS`` should be used to allow
   for future extensions.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 151)

Unknown directive type "c:function".

```
.. c:function:: vectorcallfunc PyVectorcall_Function(PyObject *op)

   If *op* does not support the vectorcall protocol (either because the type
   does not or because the specific instance does not), return *NULL*.
   Otherwise, return the vectorcall function pointer stored in *op*.
   This function never raises an exception.

   This is mostly useful to check whether or not *op* supports vectorcall,
   which can be done by checking ``PyVectorcall_Function(op) != NULL``.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 165)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyVectorcall_Call(PyObject *callable, PyObject *tuple, PyObject *dict)

   Call *callable*'s :c:type:`vectorcallfunc` with positional and keyword
   arguments given in a tuple and dict, respectively.

   This is a specialized function, intended to be put in the
   :c:member:`~PyTypeObject.tp_call` slot or be used in an implementation of ``tp_call``.
   It does not check the :const:`Py_TPFLAGS_HAVE_VECTORCALL` flag
   and it does not fall back to ``tp_call``.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.8
```

## Object Calling API

Various functions are available for calling a Python object. Each converts its arguments to a convention supported by the called object â€“ either *tp_call* or vectorcall. In order to do as little conversion as possible, pick one that best fits the format of data you have available.

The following table summarizes the available functions; please see individual documentation for details.

| Function | callable | args | kwargs |
|---|---|---|---|
| :c:func:`PyObject_Call`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 198); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | tuple | dict/`NULL` |
| :c:func:`PyObject_CallNoArgs`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 200); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | --- | --- |
| :c:func:`PyObject_CallOneArg`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 202); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | 1 object | --- |
| :c:func:`PyObject_CallObject`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 204); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | tuple/`NULL` | --- |
| :c:func:`PyObject_CallFunction`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 206); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | format | --- |

| Function | callable | args | kwargs |
| --- | --- | --- | --- |
| :c:func:`PyObject_CallMethod`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 208); *backlink*<br>Unknown interpreted text role "c:func". | obj + `char*` | format | --- |
| :c:func:`PyObject_CallFunctionObjArgs`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 210); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | variadic | --- |
| :c:func:`PyObject_CallMethodObjArgs`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 212); *backlink*<br>Unknown interpreted text role "c:func". | obj + name | variadic | --- |
| :c:func:`PyObject_CallMethodNoArgs`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 214); *backlink*<br>Unknown interpreted text role "c:func". | obj + name | --- | --- |
| :c:func:`PyObject_CallMethodOneArg`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 216); *backlink*<br>Unknown interpreted text role "c:func". | obj + name | 1 object | --- |
| :c:func:`PyObject_Vectorcall`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 218); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | vectorcall | vectorcall |
| :c:func:`PyObject_VectorcallDict`<br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 220); *backlink*<br>Unknown interpreted text role "c:func". | `PyObject *` | vectorcall | dict/`NULL` |

| Function | callable | args | kwargs |
| --- | --- | --- | --- |
| :c:func:`PyObject_VectorcallMethod`<br><br>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 222); *backlink*<br>Unknown interpreted text role "c:func". | arg + name | vectorcall | vectorcall |

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 225)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Call(PyObject *callable, PyObject *args, PyObject *kwargs)

   Call a callable Python object *callable*, with arguments given by the
   tuple *args*, and named arguments given by the dictionary *kwargs*.

   *args* must not be *NULL*; use an empty tuple if no arguments are needed.
   If no named arguments are needed, *kwargs* can be *NULL*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This is the equivalent of the Python expression:
   ``callable(*args, **kwargs)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 240)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallNoArgs(PyObject *callable)

   Call a callable Python object *callable* without any arguments. It is the
   most efficient way to call a callable Python object without any argument.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 251)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallOneArg(PyObject *callable, PyObject *arg)

   Call a callable Python object *callable* with exactly 1 positional argument
   *arg* and no keyword arguments.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 264)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallObject(PyObject *callable, PyObject *args)

   Call a callable Python object *callable*, with arguments given by the
   tuple *args*.  If no arguments are needed, then *args* can be *NULL*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This is the equivalent of the Python expression: ``callable(*args)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)call.rst, line 275)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallFunction(PyObject *callable, const char *format, ...)
```

```
Call a callable Python object *callable*, with a variable number of C arguments.
The C arguments are described using a :c:func:`Py_BuildValue` style format
string.  The format can be *NULL*, indicating that no arguments are provided.

Return the result of the call on success, or raise an exception and return
*NULL* on failure.

This is the equivalent of the Python expression: ``callable(*args)``.

Note that if you only pass :c:type:`PyObject *` args,
:c:func:`PyObject_CallFunctionObjArgs` is a faster alternative.

.. versionchanged:: 3.4
   The type of *format* was changed from ``char *``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 293)**

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallMethod(PyObject *obj, const char *name, const char *format, ...)

   Call the method named *name* of object *obj* with a variable number of C
   arguments.  The C arguments are described by a :c:func:`Py_BuildValue` format
   string that should produce a tuple.

   The format can be *NULL*, indicating that no arguments are provided.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This is the equivalent of the Python expression:
   ``obj.name(arg1, arg2, ...)``.

   Note that if you only pass :c:type:`PyObject *` args,
   :c:func:`PyObject_CallMethodObjArgs` is a faster alternative.

   .. versionchanged:: 3.4
      The types of *name* and *format* were changed from ``char *``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 314)**

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallFunctionObjArgs(PyObject *callable, ...)

   Call a callable Python object *callable*, with a variable number of
   :c:type:`PyObject *` arguments.  The arguments are provided as a variable number
   of parameters followed by *NULL*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This is the equivalent of the Python expression:
   ``callable(arg1, arg2, ...)``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 327)**

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallMethodObjArgs(PyObject *obj, PyObject *name, ...)

   Call a method of the Python object *obj*, where the name of the method is given as a
   Python string object in *name*.  It is called with a variable number of
   :c:type:`PyObject *` arguments.  The arguments are provided as a variable number
   of parameters followed by *NULL*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)call.rst, line 338)**

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallMethodNoArgs(PyObject *obj, PyObject *name)

   Call a method of the Python object *obj* without arguments,
   where the name of the method is given as a Python string object in *name*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api) call.rst, line 351)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_CallMethodOneArg(PyObject *obj, PyObject *name, PyObject *arg)

   Call a method of the Python object *obj* with a single positional argument
   *arg*, where the name of the method is given as a Python string object in
   *name*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api) call.rst, line 365)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Vectorcall(PyObject *callable, PyObject *const *args, size_t nargsf, PyObjec

   Call a callable Python object *callable*.
   The arguments are the same as for :c:type:`vectorcallfunc`.
   If *callable* supports vectorcall_, this directly calls
   the vectorcall function stored in *callable*.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api) call.rst, line 379)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_VectorcallDict(PyObject *callable, PyObject *const *args, size_t nargsf, PyO

   Call *callable* with positional arguments passed exactly as in the vectorcall_ protocol,
   but with keyword arguments passed as a dictionary *kwdict*.
   The *args* array contains only the positional arguments.

   Regardless of which protocol is used internally,
   a conversion of arguments needs to be done.
   Therefore, this function should only be used if the caller
   already has a dictionary ready to use for the keyword arguments,
   but not a tuple for the positional arguments.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api) call.rst, line 395)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_VectorcallMethod(PyObject *name, PyObject *const *args, size_t nargsf, PyObj

   Call a method using the vectorcall calling convention. The name of the method
   is given as a Python string *name*. The object whose method is called is
   *args[0]*, and the *args* array starting at *args[1]* represents the arguments
   of the call. There must be at least one positional argument.
   *nargsf* is the number of positional arguments including *args[0]*,
   plus :const:`PY_VECTORCALL_ARGUMENTS_OFFSET` if the value of ``args[0]`` may
   temporarily be changed. Keyword arguments can be passed just like in
   :c:func:`PyObject_Vectorcall`.

   If the object has the :const:`Py_TPFLAGS_METHOD_DESCRIPTOR` feature,
   this will call the unbound method object with the full
   *args* vector as arguments.

   Return the result of the call on success, or raise an exception and return
   *NULL* on failure.

   This function is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.9
```

## Call Support API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api) call.rst, line 421)

Unknown directive type "c:function".

```
.. c:function:: int PyCallable_Check(PyObject *o)

   Determine if the object *o* is callable.  Return ``1`` if the object is callable
   and ``0`` otherwise.  This function always succeeds.
```