# MPI for Python

*Release 4.0.0.dev0*

Lisandro Dalcin

Jun 30, 2023

## Contents

1. **Introduction**
   1.1 What is MPI? ................................................................. 4
   1.2 What is Python? ............................................................. 4
   1.3 Related Projects .......................................................... 4

2. **Overview**
   2.1 Communicating Python Objects and Array Data ............................................. 5
   2.2 Communicators ............................................................... 6
   2.3 Point-to-Point Communications ......................................................... 6
   2.4 Collective Communications ............................................................... 7
   2.5 Support for GPU-aware MPI .............................................................. 8
   2.6 Dynamic Process Management ............................................................... 8
   2.7 One-Sided Communications ............................................................... 9
   2.8 Parallel Input/Output ............................................................... 9
   2.9 Environmental Management ............................................................. 10

3. **Tutorial**
   3.1 Running Python scripts with MPI ....................................................... 12
   3.2 Point-to-Point Communication ............................................................ 12
   3.3 Collective Communication ............................................................... 14
   3.4 Input/Output (MPI-IO) ............................................................... 16
   3.5 Dynamic Process Management ............................................................. 17
   3.6 GPU-aware MPI + Python GPU arrays .................................................... 18
   3.7 One-Sided Communication (RMA) ......................................................... 18
   3.8 Wrapping with SWIG ................................................................. 19
   3.9 Wrapping with F2Py ................................................................. 20

4. **mpi4py**
   4.1 Runtime configuration options ............................................................. 21
   4.2 Environment variables ................................................................. 23
   4.3 Miscellaneous functions ................................................................. 25

5. **mpi4py.MPI**
   5.1 Classes ................................................................. 26
   5.2 Functions ................................................................. 27
   5.3 Attributes ................................................................. 29
Abstract

This document describes the MPI for Python package. MPI for Python provides Python bindings for the Message Passing Interface (MPI) standard, allowing Python applications to exploit multiple processors on workstations, clusters and supercomputers.

This package builds on the MPI specification and provides an object oriented interface resembling the MPI-2 C++ bindings. It supports point-to-point (sends, receives) and collective (broadcasts, scatters, gathers) communication of any picklable Python object, as well as efficient communication of Python objects exposing the Python buffer interface (e.g. NumPy arrays and builtin bytes/array/memoryview objects).

1 Introduction

Over the last years, high performance computing has become an affordable resource to many more researchers in the scientific community than ever before. The conjunction of quality open source software and commodity hardware strongly influenced the now widespread popularity of Beowulf class clusters and cluster of workstations.

Among many parallel computational models, message-passing has proven to be an effective one. This paradigm is specially suited for (but not limited to) distributed memory architectures and is used in today’s most demanding scientific and engineering application related to modeling, simulation, design, and signal processing. However, portable message-passing parallel programming used to be a nightmare in the past because of the many incompatible options developers were faced to. Fortunately, this situation definitely changed after the MPI Forum released its standard specification.

High performance computing is traditionally associated with software development using compiled languages. However, in typical applications programs, only a small part of the code is time-critical enough to require the efficiency of compiled languages. The rest of the code is generally related to memory management, error handling, input/output, and user interaction, and those are usually the most error prone and time-consuming lines of code to write and debug in the whole development process. Interpreted high-level languages can be really advantageous for this kind of tasks.

For implementing general-purpose numerical computations, MATLAB\(^1\) is the dominant interpreted programming language. In the open source side, Octave and Scilab are well known, freely distributed software packages providing compatibility with the MATLAB language. In this work, we present MPI for Python, a new package enabling applications to exploit multiple processors using standard MPI “look and feel” in Python scripts.

\(^1\) MATLAB is a registered trademark of The MathWorks, Inc.
1.1 What is MPI?

MPI, [mpi-using] [mpi-ref] the Message Passing Interface, is a standardized and portable message-passing system designed to function on a wide variety of parallel computers. The standard defines the syntax and semantics of library routines and allows users to write portable programs in the main scientific programming languages (Fortran, C, or C++).

Since its release, the MPI specification [mpi-std1] [mpi-std2] has become the leading standard for message-passing libraries for parallel computers. Implementations are available from vendors of high-performance computers and from well known open source projects like MPICH [mpi-mpich] and Open MPI [mpi-openmpi].

1.2 What is Python?

Python is a modern, easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming with dynamic typing and dynamic binding. It supports modules and packages, which encourages program modularity and code reuse. Python’s elegant syntax, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. It is easily extended with new functions and data types implemented in C or C++. Python is also suitable as an extension language for customizable applications.

Python is an ideal candidate for writing the higher-level parts of large-scale scientific applications [Hinsen97] and driving simulations in parallel architectures [Beazley97] like clusters of PC’s or SMP’s. Python codes are quickly developed, easily maintained, and can achieve a high degree of integration with other libraries written in compiled languages.

1.3 Related Projects

As this work started and evolved, some ideas were borrowed from well known MPI and Python related open source projects from the Internet.

• OOMPI
  – It has no relation with Python, but is an excellent object oriented approach to MPI.
  – It is a C++ class library specification layered on top of the C bindings that encapsulates MPI into a functional class hierarchy.
  – It provides a flexible and intuitive interface by adding some abstractions, like Ports and Messages, which enrich and simplify the syntax.

• Pypar
  – Its interface is rather minimal. There is no support for communicators or process topologies.
  – It does not require the Python interpreter to be modified or recompiled, but does not permit interactive parallel runs.
  – General (picklable) Python objects of any type can be communicated. There is good support for numeric arrays, practically full MPI bandwidth can be achieved.

• pyMPI
  – It rebuilds the Python interpreter providing a built-in module for message passing. It does permit interactive parallel runs, which are useful for learning and debugging.
It provides an interface suitable for basic parallel programming. There is not full support for defining new communicators or process topologies.

General (picklable) Python objects can be messaged between processors. There is not support for numeric arrays.

- **Scientific Python**
  - It provides a collection of Python modules that are useful for scientific computing.
  - There is an interface to MPI and BSP (*Bulk Synchronous Parallel programming*).
  - The interface is simple but incomplete and does not resemble the MPI specification. There is support for numeric arrays.

Additionally, we would like to mention some available tools for scientific computing and software development with Python.

- **NumPy** is a package that provides array manipulation and computational capabilities similar to those found in IDL, MATLAB, or Octave. Using NumPy, it is possible to write many efficient numerical data processing applications directly in Python without using any C, C++ or Fortran code.

- **SciPy** is an open source library of scientific tools for Python, gathering a variety of high level science and engineering modules together as a single package. It includes modules for graphics and plotting, optimization, integration, special functions, signal and image processing, genetic algorithms, ODE solvers, and others.

- **Cython** is a language that makes writing C extensions for the Python language as easy as Python itself. The Cython language is very close to the Python language, but Cython additionally supports calling C functions and declaring C types on variables and class attributes. This allows the compiler to generate very efficient C code from Cython code. This makes Cython the ideal language for wrapping for external C libraries, and for fast C modules that speed up the execution of Python code.

- **SWIG** is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages like Perl, Tcl/Tk, Ruby and Python. Issuing header files to SWIG is the simplest approach to interfacing C/C++ libraries from a Python module.

## 2 Overview

MPI for Python provides an object oriented approach to message passing which grounds on the standard MPI-2 C++ bindings. The interface was designed with focus in translating MPI syntax and semantics of standard MPI-2 bindings for C++ to Python. Any user of the standard C/C++ MPI bindings should be able to use this module without need of learning a new interface.

### 2.1 Communicating Python Objects and Array Data

The Python standard library supports different mechanisms for data persistence. Many of them rely on disk storage, but *pickling* and *marshaling* can also work with memory buffers.

The **pickle** modules provide user-extensible facilities to serialize general Python objects using ASCII or binary formats. The **marshal** module provides facilities to serialize built-in Python objects using a binary format specific to Python, but independent of machine architecture issues.

*MPI for Python* can communicate any built-in or user-defined Python object taking advantage of the features provided by the **pickle** module. These facilities will be routinely used to build binary representations of objects to communicate (at sending processes), and restoring them back (at receiving processes).

Although simple and general, the serialization approach (i.e., *pickling* and *unpickling*) previously discussed imposes important overheads in memory as well as processor usage, especially in the scenario of objects with large memory...
footprints being communicated. Pickling general Python objects, ranging from primitive or container built-in types to user-defined classes, necessarily requires computer resources. Processing is also needed for dispatching the appropriate serialization method (that depends on the type of the object) and doing the actual packing. Additional memory is always needed, and if its total amount is not known \textit{a priori}, many reallocations can occur. Indeed, in the case of large numeric arrays, this is certainly unacceptable and precludes communication of objects occupying half or more of the available memory resources.

\textit{MPI for Python} supports direct communication of any object exporting the single-segment buffer interface. This interface is a standard Python mechanism provided by some types (e.g., strings and numeric arrays), allowing access in the C side to a contiguous memory buffer (i.e., address and length) containing the relevant data. This feature, in conjunction with the capability of constructing user-defined MPI datatypes describing complicated memory layouts, enables the implementation of many algorithms involving multidimensional numeric arrays (e.g., image processing, fast Fourier transforms, finite difference schemes on structured Cartesian grids) directly in Python, with negligible overhead, and almost as fast as compiled Fortran, C, or C++ codes.

\subsection{2.2 Communicators}

In \textit{MPI for Python}, \texttt{Comm} is the base class of communicators. The \texttt{Intracomm} and \texttt{Intercomm} classes are subclasses of the \texttt{Comm} class. The \texttt{Comm.Is_inter} method (and \texttt{Comm.Is_intra}, provided for convenience but not part of the MPI specification) is defined for communicator objects and can be used to determine the particular communicator class.

The two predefined intracommunicator instances are available: \texttt{COMM_SELF} and \texttt{COMM_WORLD}. From them, new communicators can be created as needed.

The number of processes in a communicator and the calling process rank can be respectively obtained with methods \texttt{Comm.Get_size} and \texttt{Comm.Get_rank}. The associated process group can be retrieved from a communicator by calling the \texttt{Comm.Get_group} method, which returns an instance of the \texttt{Group} class. Set operations with \texttt{Group} objects like \texttt{Group.Union}, \texttt{Group.Intersection} and \texttt{Group.Difference} are fully supported, as well as the creation of new communicators from these groups using \texttt{Comm.Create} and \texttt{Intracomm.Create_group}.

New communicator instances can be obtained with the \texttt{Comm.Clone}, \texttt{Comm.Dup} and \texttt{Comm.Split} methods, as well methods \texttt{Intracomm.Create_intercomm} and \texttt{Intercomm.Merge}.

Virtual topologies (\texttt{Cartcomm}, \texttt{Graphcomm} and \texttt{Distgraphcomm} classes, which are specializations of the \texttt{Intracomm} class) are fully supported. New instances can be obtained from intracommunicator instances with factory methods \texttt{Intracomm.Create_cart} and \texttt{Intracomm.Create_graph}.

\subsection{2.3 Point-to-Point Communications}

Point to point communication is a fundamental capability of message passing systems. This mechanism enables the transmission of data between a pair of processes, one side sending, the other receiving.

\texttt{MPI} provides a set of \texttt{send} and \texttt{receive} functions allowing the communication of \textit{typed} data with an associated \textit{tag}. The type information enables the conversion of data representation from one architecture to another in the case of heterogeneous computing environments; additionally, it allows the representation of non-contiguous data layouts and user-defined datatypes, thus avoiding the overhead of (otherwise unavoidable) packing/unpacking operations. The tag information allows selectivity of messages at the receiving end.
**Blocking Communications**

MPI provides basic send and receive functions that are *blocking*. These functions block the caller until the data buffers involved in the communication can be safely reused by the application program.

In *MPI for Python*, the `Comm.Send`, `Comm.Recv` and `Comm.Sendrecv` methods of communicator objects provide support for blocking point-to-point communications within `Intracomm` and `Intercomm` instances. These methods can communicate memory buffers. The variants `Comm.send`, `Comm.recv` and `Comm.sendrecv` can communicate general Python objects.

**Nonblocking Communications**

On many systems, performance can be significantly increased by overlapping communication and computation. This is particularly true on systems where communication can be executed autonomously by an intelligent, dedicated communication controller.

MPI provides *nonblocking* send and receive functions. They allow the possible overlap of communication and computation. Non-blocking communication always come in two parts: posting functions, which begin the requested operation; and test-for-completion functions, which allow to discover whether the requested operation has completed.

In *MPI for Python*, the `Comm.Isend` and `Comm.Irecv` methods initiate send and receive operations, respectively. These methods return a `Request` instance, uniquely identifying the started operation. Its completion can be managed using the `Request.Test`, `Request.Wait` and `Request.Cancel` methods. The management of `Request` objects and associated memory buffers involved in communication requires a careful, rather low-level coordination. Users must ensure that objects exposing their memory buffers are not accessed at the Python level while they are involved in nonblocking message-passing operations.

**Persistent Communications**

Often a communication with the same argument list is repeatedly executed within an inner loop. In such cases, communication can be further optimized by using persistent communication, a particular case of nonblocking communication allowing the reduction of the overhead between processes and communication controllers. Furthermore, this kind of optimization can also alleviate the extra call overheads associated to interpreted, dynamic languages like Python.

In *MPI for Python*, the `Comm.Send_init` and `Comm.Recv_init` methods create persistent requests for a send and receive operation, respectively. These methods return an instance of the `Prequest` class, a subclass of the `Request` class. The actual communication can be effectively started using the `Prequest.Start` method, and its completion can be managed as previously described.

### 2.4 Collective Communications

Collective communications allow the transmittal of data between multiple processes of a group simultaneously. The syntax and semantics of collective functions is consistent with point-to-point communication. Collective functions communicate *typed* data, but messages are not paired with an associated *tag*; selectivity of messages is implied in the calling order. Additionally, collective functions come in blocking versions only.

The more commonly used collective communication operations are the following.

- Barrier synchronization across all group members.
- Global communication functions
  - Broadcast data from one member to all members of a group.
  - Gather data from all members to one member of a group.
  - Scatter data from one member to all members of a group.
• Global reduction operations such as sum, maximum, minimum, etc.


Global reduction operations on memory buffers are accessible through the `Comm.Reduce`, `Comm.Reduce_scatter`, `Comm.Allreduce`, `Intracomm.Scan` and `Intracomm.Exscan` methods. The lower-case variants `Comm.reduce`, `Comm.allreduce`, `Intracomm.scan` and `Intracomm.exscan` can communicate general Python objects; however, the actual required reduction computations are performed sequentially at some process. All the predefined (i.e., `SUM`, `PROD`, `MAX`, etc.) reduction operations can be applied.

### 2.5 Support for GPU-aware MPI

Several MPI implementations, including Open MPI and MVAPICH, support passing GPU pointers to MPI calls to avoid explicit data movement between host and device. On the Python side, support for handling GPU arrays have been implemented in many libraries related GPU computation such as CuPy, Numba, PyTorch, and PyArrow. To maximize interoperability across library boundaries, two kinds of zero-copy data exchange protocols have been defined and agreed upon: DLPack and CUDA Array Interface (CAI).

**MPI for Python** provides an experimental support for GPU-aware MPI. This feature requires:

1. mpi4py is built against a GPU-aware MPI library.
2. The Python GPU arrays are compliant with either of the protocols.

See the **Tutorial** section for further information. We note that

- Whether or not a MPI call can work for GPU arrays depends on the underlying MPI implementation, not on mpi4py.
- This support is currently experimental and subject to change in the future.

### 2.6 Dynamic Process Management

In the context of the MPI-1 specification, a parallel application is static; that is, no processes can be added to or deleted from a running application after it has been started. Fortunately, this limitation was addressed in MPI-2. The new specification added a process management model providing a basic interface between an application and external resources and process managers.

This MPI-2 extension can be really useful, especially for sequential applications built on top of parallel modules, or parallel applications with a client/server model. The MPI-2 process model provides a mechanism to create new processes and establish communication between them and the existing MPI application. It also provides mechanisms to establish communication between two existing MPI applications, even when one did not start the other.

In **MPI for Python**, new independent process groups can be created by calling the `Intracomm.Spawn` method within an intracomnicator. This call returns a new intercomnicator (i.e., an `Intercomm` instance) at the parent process group. The child process group can retrieve the matching intercomnicator by calling the `Comm.Get_parent` class method. At each side, the new intercomnicator can be used to perform point to point and collective communications between the parent and child groups of processes.

Alternatively, disjoint groups of processes can establish communication using a client/server approach. Any server application must first call the `Open_port` function to open a `port` and the `Publish_name` function to publish a provided
service, and next call the Intracomm. Accept method. Any client applications can first find a published service by calling the Lookup_name function, which returns the port where a server can be contacted; and next call the Intracomm. Connect method. Both Intracomm.Accept and Intracomm.Connect methods return an Intercomm instance. When connection between client/server processes is no longer needed, all of them must cooperatively call the Comm. Disconnect method. Additionally, server applications should release resources by calling the Unpublish_name and Close_port functions.

2.7 One-Sided Communications

One-sided communications (also called Remote Memory Access, RMA) supplements the traditional two-sided, send/receive based MPI communication model with a one-sided, put/get based interface. One-sided communication that can take advantage of the capabilities of highly specialized network hardware. Additionally, this extension lowers latency and software overhead in applications written using a shared-memory-like paradigm.

The MPI specification revolves around the use of objects called windows; they intuitively specify regions of a process’s memory that have been made available for remote read and write operations. The published memory blocks can be accessed through three functions for put (remote send), get (remote write), and accumulate (remote update or reduction) data items. A much larger number of functions support different synchronization styles; the semantics of these synchronization operations are fairly complex.

In MPI for Python, one-sided operations are available by using instances of the Win class. New window objects are created by calling the Win.Create method at all processes within a communicator and specifying a memory buffer. When a window instance is no longer needed, the Win.Free method should be called.

The three one-sided MPI operations for remote write, read and reduction are available through calling the methods Win.Put, Win.Get, and Win.Accumulate respectively within a Win instance. These methods need an integer rank identifying the target process and an integer offset relative the base address of the remote memory block being accessed.

The one-sided operations read, write, and reduction are implicitly nonblocking, and must be synchronized by using two primary modes. Active target synchronization requires the origin process to call the Win.Start and Win.Complete methods at the origin process, and target process cooperates by calling the Win.Post and Win.Wait methods. There is also a collective variant provided by the Win.Fence method. Passive target synchronization is more lenient, only the origin process calls the Win.Lock and Win.Unlock methods. Locks are used to protect remote accesses to the locked remote window and to protect local load/store accesses to a locked local window.

2.8 Parallel Input/Output

The POSIX standard provides a model of a widely portable file system. However, the optimization needed for parallel input/output cannot be achieved with this generic interface. In order to ensure efficiency and scalability, the underlying parallel input/output system must provide a high-level interface supporting partitioning of file data among processes and a collective interface supporting complete transfers of global data structures between process memories and files. Additionally, further efficiencies can be gained via support for asynchronous input/output, strided accesses to data, and control over physical file layout on storage devices. This scenario motivated the inclusion in the MPI-2 standard of a custom interface in order to support more elaborated parallel input/output operations.

The MPI specification for parallel input/output revolves around the use objects called files. As defined by MPI, files are not just contiguous byte streams. Instead, they are regarded as ordered collections of typed data items. MPI supports sequential or random access to any integral set of these items. Furthermore, files are opened collectively by a group of processes.

The common patterns for accessing a shared file (broadcast, scatter, gather, reduction) is expressed by using user-defined datatypes. Compared to the communication patterns of point-to-point and collective communications, this approach has the advantage of added flexibility and expressiveness. Data access operations (read and write) are defined for different kinds of positioning (using explicit offsets, individual file pointers, and shared file pointers), coordination (non-collective and collective), and synchronism (blocking, nonblocking, and split collective with begin/end phases).
In *MPI for Python*, all MPI input/output operations are performed through instances of the *File* class. File handles are obtained by calling the *File.Open* method at all processes within a communicator and providing a file name and the intended access mode. After use, they must be closed by calling the *File.Close* method. Files even can be deleted by calling method *File.Delete*.

After creation, files are typically associated with a per-process *view*. The view defines the current set of data visible and accessible from an open file as an ordered set of elementary datatypes. This data layout can be set and queried with the *File.Set_view* and *File.Get_view* methods respectively.

Actual input/output operations are achieved by many methods combining read and write calls with different behavior regarding positioning, coordination, and synchronism. Summing up, *MPI for Python* provides the thirty (30) methods defined in MPI-2 for reading from or writing to files using explicit offsets or file pointers (individual or shared), in blocking or nonblocking and collective or noncollective versions.

### 2.9 Environmental Management

**Initialization and Exit**

Module functions *Init* or *Init_thread* and *Finalize* provide MPI initialization and finalization respectively. Module functions *Is_initialized* and *Is_finalized* provide the respective tests for initialization and finalization.

**Note:** *MPI_Init()* or *MPI_Init_thread()* is actually called when you import the *MPI* module from the *mpi4py* package, but only if MPI is not already initialized. In such case, calling *Init* or *Init_thread* from Python is expected to generate an MPI error, and in turn an exception will be raised.

**Note:** *MPI_Finalize()* is registered (by using Python C/API function *Py_AtExit()*()) for being automatically called when Python processes exit, but only if *mpi4py* actually initialized MPI. Therefore, there is no need to call *Finalize* from Python to ensure MPI finalization.

**Implementation Information**

- The MPI version number can be retrieved from module function *Get_version*. It returns a two-integer tuple (version, subversion).
- The *Get_processor_name* function can be used to access the processor name.
- The values of predefined attributes attached to the world communicator can be obtained by calling the *Comm. Get_attr* method within the *COMM_WORLD* instance.

**Timers**

MPI timer functionalities are available through the *Wtime* and *Wtick* functions.
**Error Handling**

In order to facilitate handle sharing with other Python modules interfacing MPI-based parallel libraries, the predefined MPI error handlers `ERRORS_RETURN` and `ERRORS_ARE_FATAL` can be assigned to and retrieved from communicators using methods `Comm.Set_errhandler` and `Comm.Get_errhandler`, and similarly for windows and files. New custom error handlers can be created with `Comm.Create_errhandler`.

When the predefined error handler `ERRORS_RETURN` is set, errors returned from MPI calls within Python code will raise an instance of the exception class `Exception`, which is a subclass of the standard Python exception `RuntimeError`.

---

**Note:** After import, mpi4py overrides the default MPI rules governing inheritance of error handlers. The `ERRORS_RETURN` error handler is set in the predefined `COMM_SELF` and `COMM_WORLD` communicators, as well as any new `Comm`, `Win`, or `File` instance created through mpi4py. If you ever pass such handles to C/C++/Fortran library code, it is recommended to set the `ERRORS_ARE_FATAL` error handler on them to ensure MPI errors do not pass silently.

---

**Warning:** Importing with `from mpi4py.MPI import *` will cause a name clashing with the standard Python `Exception` base class.

---

**3 Tutorial**

**Warning:** Under construction. Contributions very welcome!

---

**Tip:** Rolf Rabenseifner at HLRS developed a comprehensive MPI-3.1/4.0 course with slides and a large set of exercises including solutions. This material is available online for self-study. The slides and exercises show the C, Fortran, and Python (mpi4py) interfaces. For performance reasons, most Python exercises use NumPy arrays and communication routines involving buffer-like objects.

---

**Tip:** Victor Eijkhout at TACC authored the book *Parallel Programming for Science and Engineering*. This book is available online in PDF and HTML formats. The book covers parallel programming with MPI and OpenMP in C/C++ and Fortran, and MPI in Python using mpi4py.

---

*MPI for Python* supports convenient, pickle-based communication of generic Python object as well as fast, near C-speed, direct array data communication of buffer-provider objects (e.g., NumPy arrays).

- Communication of generic Python objects
  - You have to use methods with all-lowercase names, like `Comm.send`, `Comm.recv`, `Comm.bcast`, `Comm.scatter`, `Comm.gather`. An object to be sent is passed as a parameter to the communication call, and the received object is simply the return value.
  - The `Comm.isend` and `Comm.irecv` methods return `Request` instances; completion of these methods can be managed using the `Request.test` and `Request.wait` methods.
  - The `Comm.recv` and `Comm.irecv` methods may be passed a buffer object that can be repeatedly used to receive messages avoiding internal memory allocation. This buffer must be sufficiently large to accommodate the transmitted messages; hence, any buffer passed to `Comm.recv` or `Comm.irecv` must be at least as long as the pickled data transmitted to the receiver.
Collective calls like `Comm.scatter`, `Comm.gather`, `Comm.allgather`, `Comm.alltoall` expect a single value or a sequence of `Comm.size` elements at the root or all process. They return a single value, a list of `Comm.size` elements, or `None`.

**Note:** MPI for Python uses the highest protocol version available in the Python runtime (see the `HIGHEST_PROTOCOL` constant in the `pickle` module). The default protocol can be changed at import time by setting the `MPI4PY_PICKLE_PROTOCOL` environment variable, or at runtime by assigning a different value to the `PROTOCOL` attribute of the `pickle` object within the `MPI` module.

- Communication of buffer-like objects


  In general, buffer arguments to these calls must be explicitly specified by using a 2/3-list/tuple like `[data, MPI.DOUBLE]`, or `[data, count, MPI.DOUBLE]` (the former one uses the byte-size of `data` and the extent of the MPI datatype to define `count`).

  For vector collectives communication operations like `Comm.Scatterv` and `Comm.Gatherv`, buffer arguments are specified as `[data, count, displ, datatype]`, where `count` and `displ` are sequences of integral values.

  Automatic MPI datatype discovery for NumPy/GPU arrays and PEP-3118 buffers is supported, but limited to basic C types (all C/C99-native signed/unsigned integral types and single/double precision real/complex floating types) and availability of matching datatypes in the underlying MPI implementation. In this case, the buffer-provider object can be passed directly as a buffer argument, the count and MPI datatype will be inferred.

  If mpi4py is built against a GPU-aware MPI implementation, GPU arrays can be passed to upper-case methods as long as they have either the `__dlpack__` and `__dlpack_device__` methods or the `__cuda_array_interface__` attribute that are compliant with the respective standard specifications. Moreover, only C-contiguous or Fortran-contiguous GPU arrays are supported. It is important to note that GPU buffers must be fully ready before any MPI routines operate on them to avoid race conditions. This can be ensured by using the synchronization API of your array library. mpi4py does not have access to any GPU-specific functionality and thus cannot perform this operation automatically for users.

### 3.1 Running Python scripts with MPI

Most MPI programs can be run with the command `mpiexec`. In practice, running Python programs looks like:

```
$ mpiexec -n 4 python script.py
```

to run the program with 4 processors.

### 3.2 Point-to-Point Communication

- Python objects (`pickle` under the hood):

  ```python
  from mpi4py import MPI
  comm = MPI.COMM_WORLD
  rank = comm.Get_rank()

  if rank == 0:
    data = {'a': 7, 'b': 3.14}
  ```

  (continues on next page)
comm.send(data, dest=1, tag=11)

```python
elif rank == 1:
    data = comm.recv(source=0, tag=11)
```

- Python objects with non-blocking communication:

```python
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    data = {'a': 7, 'b': 3.14}
    req = comm.isend(data, dest=1, tag=11)
    req.wait()

elif rank == 1:
    req = comm.irecv(source=0, tag=11)
    data = req.wait()
```

- NumPy arrays (the fast way!):

```python
from mpi4py import MPI
import numpy

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

# passing MPI datatypes explicitly
if rank == 0:
    data = numpy.arange(1000, dtype='i')
    comm.Send([data, MPI.INT], dest=1, tag=77)

elif rank == 1:
    data = numpy.empty(1000, dtype='i')
    comm.Recv([data, MPI.INT], source=0, tag=77)

# automatic MPI datatype discovery
if rank == 0:
    data = numpy.arange(100, dtype=numpy.float64)
    comm.Send(data, dest=1, tag=13)

elif rank == 1:
    data = numpy.empty(100, dtype=numpy.float64)
    comm.Recv(data, source=0, tag=13)
```
3.3 Collective Communication

- Broadcasting a Python dictionary:

```python
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
data = {'key1': [7, 2.72, 2+3j],
        'key2': ('abc', 'xyz')}
elset:
data = None
data = comm.bcast(data, root=0)
```

- Scattering Python objects:

```python
from mpi4py import MPI
comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()

if rank == 0:
data = [(i+1)**2 for i in range(size)]
elset:
data = None
data = comm.scatter(data, root=0)
assert data == (rank+1)**2
```

- Gathering Python objects:

```python
from mpi4py import MPI
comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()
data = (rank+1)**2
data = comm.gather(data, root=0)
if rank == 0:
    for i in range(size):
        assert data[i] == (i+1)**2
else:
    assert data is None
```

- Broadcasting a NumPy array:

```python
from mpi4py import MPI
import numpy as np
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
```
if rank == 0:
    data = np.arange(100, dtype='i')
else:
    data = np.empty(100, dtype='i')
comm.Bcast(data, root=0)
for i in range(100):
    assert data[i] == i

• Scattering NumPy arrays:

```python
from mpi4py import MPI
import numpy as np

comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()

sendbuf = None
if rank == 0:
    sendbuf = np.empty([size, 100], dtype='i')
    sendbuf.T[:,::] = range(size)
recvbuf = np.empty(100, dtype='i')
comm.Scatter(sendbuf, recvbuf, root=0)
assert np.allclose(recvbuf, rank)
```

• Gathering NumPy arrays:

```python
from mpi4py import MPI
import numpy as np

comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()

sendbuf = np.zeros(100, dtype='i') + rank
recvbuf = None
if rank == 0:
    recvbuf = np.empty([size, 100], dtype='i')
comm.Gather(sendbuf, recvbuf, root=0)
if rank == 0:
    for i in range(size):
        assert np.allclose(recvbuf[i,::], i)
```

• Parallel matrix-vector product:

```python
from mpi4py import MPI
import numpy

def matvec(comm, A, x):
    m = A.shape[0]  # local rows
    p = comm.Get_size()
    xg = numpy.zeros(m*p, dtype='d')
```
comm.Allgather([x, MPI.DOUBLE],
   [xg, MPI.DOUBLE])
y = numpy.dot(A, xg)
return y

3.4 Input/Output (MPI-IO)

• Collective I/O with NumPy arrays:

```python
from mpi4py import MPI
import numpy as np

amode = MPI.MODE_WRONLY|MPI.MODE_CREATE
comm = MPI.COMM_WORLD
fh = MPI.File.Open(comm, "./datafile.contig", amode)

buffer = np.empty(10, dtype=np.int)
buffer[:] = comm.Get_rank()

offset = comm.Get_rank()*buffer.nbytes
fh.Write_at_all(offset, buffer)

fh.Close()
```

• Non-contiguous Collective I/O with NumPy arrays and datatypes:

```python
from mpi4py import MPI
import numpy as np

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

amode = MPI.MODE_WRONLY|MPI.MODE_CREATE
fh = MPI.File.Open(comm, "./datafile.noncontig", amode)

item_count = 10
buffer = np.empty(item_count, dtype='i')
buffer[:] = rank

filetype = MPI.INT.Create_vector(item_count, 1, size)
filetype.Commit()

displacement = MPI.INT.Get_size()*rank
fh.Set_view(displacement, filetype=filetype)

fh.Write_all(buffer)
filetype.Free()
fh.Close()
```
3.5 Dynamic Process Management

- Compute Pi - Master (or parent, or client) side:

```python
#!/usr/bin/env python
from mpi4py import MPI
import numpy
import sys

comm = MPI.COMM_SELF.Spawn(sys.executable,
                             args=['cpi.py'],
                             maxprocs=5)

N = numpy.array(100, 'i')
comm.Bcast([N, MPI.INT], root=MPI.ROOT)
PI = numpy.array(0.0, 'd')
comm.Reduce(None, [PI, MPI.DOUBLE],
             op=MPI.SUM, root=MPI.ROOT)
print(PI)

comm.Disconnect()
```

- Compute Pi - Worker (or child, or server) side:

```python
#!/usr/bin/env python
from mpi4py import MPI
import numpy

comm = MPI.Comm.Get_parent()
size = comm.Get_size()
rank = comm.Get_rank()

N = numpy.array(0, dtype='i')
comm.Bcast([N, MPI.INT], root=0)
h = 1.0 / N; s = 0.0
for i in range(rank, N, size):
    x = h * (i + 0.5)
    s += 4.0 / (1.0 + x**2)
PI = numpy.array(s * h, dtype='d')
comm.Reduce([PI, MPI.DOUBLE], None,
            op=MPI.SUM, root=0)

comm.Disconnect()
```
3.6 GPU-aware MPI + Python GPU arrays

- Reduce-to-all CuPy arrays:

```python
from mpi4py import MPI
import cupy as cp

comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()

sendbuf = cp.arange(10, dtype='i')
recvbuf = cp.empty_like(sendbuf)
cp.cuda.get_current_stream().synchronize()
comm.Allreduce(sendbuf, recvbuf)

assert cp.allclose(recvbuf, sendbuf*size)
```

3.7 One-Sided Communication (RMA)

- Read from (write to) the entire RMA window:

```python
import numpy as np
from mpi4py import MPI
from mpi4py.util import dtlib

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

datatype = MPI.FLOAT
np_dtype = dtlib.to_numpy_dtype(datatype)
itemsize = datatype.Get_size()

N = 10
win_size = N * itemsize if rank == 0 else 0
win = MPI.Win.Allocate(win_size, comm=comm)

buf = np.empty(N, dtype=np_dtype)
if rank == 0:
    buf.fill(42)
    win.Lock(rank=0)
    win.Put(buf, target_rank=0)
    win.Unlock(rank=0)
    comm.Barrier()
else:
    comm.Barrier()
    win.Lock(rank=0)
    win.Get(buf, target_rank=0)
    win.Unlock(rank=0)

assert np.all(buf == 42)
```

- Accessing a part of the RMA window using the `target` argument, which is defined as `(offset, count, datatype)`: 

import numpy as np
from mpi4py import MPI
from mpi4py.util import dtlib

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

datatype = MPI.FLOAT
np_dtype = dtlib.to_numpy_dtype(datatype)
itemsize = datatype.Get_size()

N = comm.Get_size() + 1
win_size = N * itemsize
if rank == 0 else 0
win = MPI.Win.Allocate(size=win_size, disp_unit=itemsize, comm=comm,
    size=win_size,
    disp_unit=itemsize,
    comm=comm,
)
if rank == 0:
    mem = np.frombuffer(win, dtype=np_dtype)
    mem[:] = np.arange(len(mem), dtype=np_dtype)
comm.Barrier()

buf = np.zeros(3, dtype=np_dtype)
target = (rank, 2, datatype)
win.Lock(rank=0)
win.Get(buf, target_rank=0, target=target)
win.Unlock(rank=0)
assert np.all(buf == [rank, rank+1, 0])

3.8 Wrapping with SWIG

• C source:

```c
/* file: helloworld.c */
void sayhello(MPI_Comm comm)
{
    int size, rank;
    MPI_Comm_size(comm, &size);
    MPI_Comm_rank(comm, &rank);
    printf("Hello, World! 
           "I am process %d of %d.
", rank, size);
}
```

• SWIG interface file:

```c
// file: helloworld.i
%module helloworld
{%
#include <mpi.h>
#include "helloworld.c"
```
• Try it in the Python prompt:

```python
>>> from mpi4py import MPI
>>> import helloworld
>>> helloworld.sayhello(MPI.COMM_WORLD)
Hello, World! I am process 0 of 1.
```

3.9 Wrapping with F2Py

• Fortran 90 source:

```
! file: helloworld.f90
subroutine sayhello(comm)
    use mpi
    implicit none
    integer :: comm, rank, size, ierr
    call MPI_Comm_size(comm, size, ierr)
    call MPI_Comm_rank(comm, rank, ierr)
    print *, 'Hello, World! I am process ',rank,' of ',size,'.'
end subroutine sayhello
```

• Compiling example using f2py

```
$ f2py -c --f90exec=mpif90 helloworld.f90 -m helloworld
```

• Try it in the Python prompt:

```python
>>> from mpi4py import MPI
>>> import helloworld
>>> fcomm = MPI.COMM_WORLD.py2f()
>>> helloworld.sayhello(fcomm)
Hello, World! I am process 0 of 1.
```

4 mpi4py

The MPI for Python package.

The Message Passing Interface (MPI) is a standardized and portable message-passing system designed to function on a wide variety of parallel computers. The MPI standard defines the syntax and semantics of library routines and allows users to write portable programs in the main scientific programming languages (Fortran, C, or C++). Since its release, the MPI specification has become the leading standard for message-passing libraries for parallel computers.

MPI for Python provides MPI bindings for the Python programming language, allowing any Python program to exploit multiple processors. This package build on the MPI specification and provides an object oriented interface which closely follows MPI-2 C++ bindings.
4.1 Runtime configuration options

mpi4py.rc

This object has attributes exposing runtime configuration options that become effective at import time of the MPI module.

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize</td>
<td>Automatic MPI initialization at import</td>
</tr>
<tr>
<td>threads</td>
<td>Request initialization with thread support</td>
</tr>
<tr>
<td>thread_level</td>
<td>Level of thread support to request</td>
</tr>
<tr>
<td>finalize</td>
<td>Automatic MPI finalization at exit</td>
</tr>
<tr>
<td>fast_reduce</td>
<td>Use tree-based reductions for objects</td>
</tr>
<tr>
<td>recv_mprobe</td>
<td>Use matched probes to receive objects</td>
</tr>
<tr>
<td>errors</td>
<td>Error handling policy</td>
</tr>
</tbody>
</table>

Attributes Documentation

mpi4py.rc.initialize

Automatic MPI initialization at import.

Type

bool

Default

True

See also:

MPI4PY_RC_INITIALIZE

mpi4py.rc.threads

Request initialization with thread support.

Type

bool

Default

True

See also:

MPI4PY_RC_THREADS

mpi4py.rc.thread_level

Level of thread support to request.

Type

str

Default

"multiple"

Choices

"multiple", "serialized", "funneled", "single"
See also:

*MPI4PY_RC_THREAD_LEVEL*

**mpi4py.rc.finalize**

Automatic MPI finalization at exit.

**Type**

None or bool

**Default**

None

See also:

*MPI4PY_RC_FINALIZE*

**mpi4py.rc.fast_reduce**

Use tree-based reductions for objects.

**Type**

bool

**Default**

True

See also:

*MPI4PY_RC_FAST_REDUCE*

**mpi4py.rc.recv_mprobe**

Use matched probes to receive objects.

**Type**

bool

**Default**

True

See also:

*MPI4PY_RC_RECV_MPROBE*

**mpi4py.rc.errors**

Error handling policy.

**Type**

str

**Default**

"exception"

**Choices**

"exception", "default", "abort", "fatal"

See also:

*MPI4PY_RC_ERRORS*
Example

MPI for Python features automatic initialization and finalization of the MPI execution environment. By using the `mpi4py.rc` object, MPI initialization and finalization can be handled programmatically:

```python
import mpi4py
mpi4py.rc.initialize = False  # do not initialize MPI automatically
mpi4py.rc.finalize = False    # do not finalize MPI automatically

from mpi4py import MPI       # import the 'MPI' module

MPI.Init()                   # manual initialization of the MPI environment
...
# your finest code here ...

MPI.Finalize()                # manual finalization of the MPI environment
```

4.2 Environment variables

The following environment variables override the corresponding attributes of the `mpi4py.rc` and `MPI.pickle` objects at import time of the `MPI` module.

**Note:** For variables of boolean type, accepted values are 0 and 1 (interpreted as `False` and `True`, respectively), and strings specifying a YAML boolean value (case-insensitive).

**MPI4PY_RC_INITIALIZE**

<table>
<thead>
<tr>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
</tr>
</tbody>
</table>

**Default**

| True         |

Whether to automatically initialize MPI at import time of the `mpi4py.MPI` module.

**See also:**

`mpi4py.rc.initialize`

New in version 4.0.0.

**MPI4PY_RC_FINALIZE**

<table>
<thead>
<tr>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

**Default**

| None         |

**Choices**

| None, True, False |

Whether to automatically finalize MPI at exit time of the Python process.

**See also:**

`mpi4py.rc.finalize`

New in version 4.0.0.
**MPI4PY_RC_THREADS**

Type
   bool

Default
   True

Whether to initialize MPI with thread support.

See also:
   mpi4py.rc.threads

New in version 3.1.0.

**MPI4PY_RC_THREAD_LEVEL**

Default
   "multiple"

Choices
   "single", "funneled", "serialized", "multiple"

The level of required thread support.

See also:
   mpi4py.rc.thread_level

New in version 3.1.0.

**MPI4PY_RC_FAST_REDUCE**

Type
   bool

Default
   True

Whether to use tree-based reductions for objects.

See also:
   mpi4py.rc.fast_reduce

New in version 3.1.0.

**MPI4PY_RC_RECV_MPROBE**

Type
   bool

Default
   True

Whether to use matched probes to receive objects.

See also:
   mpi4py.rc.recv_mprobe

**MPI4PY_RC_ERRORS**

Default
   "exception"
Choices
"exception", "default", "abort", "fatal"

Controls default MPI error handling policy.

See also:

mpi4py.rc.errors

New in version 3.1.0.

MPI4PY_PICKLE_PROTOCOL

Type
int

Default
pickle.HIGHEST_PROTOCOL

Controls the default pickle protocol to use when communicating Python objects.

See also:

PROTOCOL attribute of the MPI.pickle object within the MPI module.

New in version 3.1.0.

MPI4PY_PICKLE_THRESHOLD

Type
int

Default
262144

Controls the default buffer size threshold for switching from in-band to out-of-band buffer handling when using pickle protocol version 5 or higher.

See also:

THRESHOLD attribute of the MPI.pickle object within the MPI module.

New in version 3.1.2.

4.3 Miscellaneous functions

mpi4py.profile(name, *, path=None)

Support for the MPI profiling interface.

Parameters

• name (str) – Name of the profiler library to load.
• path (sequence of str, optional) – Additional paths to search for the profiler.

Return type

None

mpi4py.get_config()

Return a dictionary with information about MPI.

Return type
dict[str, str]
mpi4py.get_include()

Return the directory in the package that contains header files.

Extension modules that need to compile against mpi4py should use this function to locate the appropriate include directory. Using Python distutils (or perhaps NumPy distutils):

```python
import mpi4py
Extension('extension_name', ...
           include_dirs=[..., mpi4py.get_include()])
```

Return type
str

5 mpi4py.MPI

5.1 Classes

Ancillary

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Datatype object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Status object</td>
</tr>
<tr>
<td>Request</td>
<td>Request handle</td>
</tr>
<tr>
<td>Prequest</td>
<td>Persistent request handle</td>
</tr>
<tr>
<td>Grequest</td>
<td>Generalized request handle</td>
</tr>
<tr>
<td>Op</td>
<td>Operation object</td>
</tr>
<tr>
<td>Group</td>
<td>Group of processes</td>
</tr>
<tr>
<td>Info</td>
<td>Info object</td>
</tr>
</tbody>
</table>

Communication

<table>
<thead>
<tr>
<th>Comm</th>
<th>Communicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intracomm</td>
<td>Intracomunicator</td>
</tr>
<tr>
<td>Topocomm</td>
<td>Topology intracomunicator</td>
</tr>
<tr>
<td>Cartcomm</td>
<td>Cartesian topology intracomunicator</td>
</tr>
<tr>
<td>Graphcomm</td>
<td>General graph topology intracomunicator</td>
</tr>
<tr>
<td>Distgraphcomm</td>
<td>Distributed graph topology intracomunicator</td>
</tr>
<tr>
<td>Intercomm</td>
<td>Intercommunicator</td>
</tr>
<tr>
<td>Message</td>
<td>Matched message handle</td>
</tr>
</tbody>
</table>
5.2 Functions

Version inquiry

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_version()</td>
<td>Obtain the version number of the MPI standard supported by the implementation as a tuple (version, subversion)</td>
</tr>
<tr>
<td>Get_library_version()</td>
<td>Obtain the version string of the MPI library</td>
</tr>
</tbody>
</table>

Initialization and finalization

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init()</td>
<td>Initialize the MPI execution environment</td>
</tr>
<tr>
<td>Init_thread([required])</td>
<td>Initialize the MPI execution environment</td>
</tr>
<tr>
<td>Finalize()</td>
<td>Terminate the MPI execution environment</td>
</tr>
<tr>
<td>Is_initialized()</td>
<td>Indicates whether Init has been called</td>
</tr>
<tr>
<td>Is_finalized()</td>
<td>Indicates whether Finalize has completed</td>
</tr>
<tr>
<td>Query_thread()</td>
<td>Return the level of thread support provided by the MPI library</td>
</tr>
<tr>
<td>Is_thread_main()</td>
<td>Indicate whether this thread called Init or Init_thread</td>
</tr>
</tbody>
</table>
**Memory allocation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Alloc_mem(size[, info])</code></td>
<td>Allocate memory for message passing and RMA</td>
</tr>
<tr>
<td><code>Free_mem(mem)</code></td>
<td>Free memory allocated with <code>Alloc_mem()</code></td>
</tr>
</tbody>
</table>

**Address manipulation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Get_address(location)</code></td>
<td>Get the address of a location in memory</td>
</tr>
<tr>
<td><code>Aint_add(base, disp)</code></td>
<td>Return the sum of base address and displacement</td>
</tr>
<tr>
<td><code>Aint_diff(addr1, addr2)</code></td>
<td>Return the difference between absolute addresses</td>
</tr>
</tbody>
</table>

**Timer**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Wtick()</code></td>
<td>Return the resolution of <code>Wtime</code></td>
</tr>
<tr>
<td><code>Wtime()</code></td>
<td>Return an elapsed time on the calling processor</td>
</tr>
</tbody>
</table>

**Error handling**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Get_error_class(errorcode)</code></td>
<td>Convert an <code>error code</code> into an <code>error class</code></td>
</tr>
<tr>
<td><code>Get_error_string(errorcode)</code></td>
<td>Return the <code>error string</code> for a given <code>error class</code> or <code>error code</code></td>
</tr>
<tr>
<td><code>Add_error_class()</code></td>
<td>Add an <code>error class</code> to the known error classes</td>
</tr>
<tr>
<td><code>Add_error_code(errorclass)</code></td>
<td>Add an <code>error code</code> to an <code>error class</code></td>
</tr>
<tr>
<td><code>Add_error_string(errorcode, string)</code></td>
<td>Associate an <code>error string</code> with an <code>error class</code> or <code>error code</code></td>
</tr>
</tbody>
</table>

**Dynamic process management**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Open_port([info])</code></td>
<td>Return an address that can be used to establish connections between groups of MPI processes</td>
</tr>
<tr>
<td><code>Close_port(port_name)</code></td>
<td>Close a port</td>
</tr>
<tr>
<td><code>Publish_name(service_name, port_name[, info])</code></td>
<td>Publish a service name</td>
</tr>
<tr>
<td><code>Unpublish_name(service_name, port_name[, info])</code></td>
<td>Unpublish a service name</td>
</tr>
<tr>
<td><code>Lookup_name(service_name[, info])</code></td>
<td>Lookup a port name given a service name</td>
</tr>
</tbody>
</table>
## Miscellanea

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attach_buffer(buf)</strong></td>
<td>Attach a user-provided buffer for sending in buffered mode</td>
</tr>
<tr>
<td><strong>Detach_buffer()</strong></td>
<td>Remove an existing attached buffer</td>
</tr>
<tr>
<td><strong>Compute_dims(nodes, dims)</strong></td>
<td>Return a balanced distribution of processes per coordinate direction</td>
</tr>
<tr>
<td><strong>Get_processor_name()</strong></td>
<td>Obtain the name of the calling processor</td>
</tr>
<tr>
<td><strong>Register_datarep(datarep, read_fn, write_fn, ...)</strong></td>
<td>Register user-defined data representations</td>
</tr>
<tr>
<td><strong>Pcontrol(level)</strong></td>
<td>Control profiling</td>
</tr>
</tbody>
</table>

## Utilities

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>get_vendor()</strong></td>
<td>Information about the underlying MPI implementation</td>
</tr>
</tbody>
</table>

## 5.3 Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDEFINED</td>
<td>Constant UNDEFINED of type int</td>
</tr>
<tr>
<td>ANY_SOURCE</td>
<td>Constant ANY_SOURCE of type int</td>
</tr>
<tr>
<td>ANY_TAG</td>
<td>Constant ANY_TAG of type int</td>
</tr>
<tr>
<td>PROC_NULL</td>
<td>Constant PROC_NULL of type int</td>
</tr>
<tr>
<td>ROOT</td>
<td>Constant ROOT of type int</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Constant BOTTOM of type BottomType</td>
</tr>
<tr>
<td>IN_PLACE</td>
<td>Constant IN_PLACE of type InPlaceType</td>
</tr>
<tr>
<td>KEYVAL_INVALID</td>
<td>Constant KEYVAL_INVALID of type int</td>
</tr>
<tr>
<td>TAG_UB</td>
<td>Constant TAG_UB of type int</td>
</tr>
<tr>
<td>HOST</td>
<td>Constant HOST of type int</td>
</tr>
<tr>
<td>IO</td>
<td>Constant IO of type int</td>
</tr>
<tr>
<td>WTIME_IS_GLOBAL</td>
<td>Constant WTIME_IS_GLOBAL of type int</td>
</tr>
<tr>
<td>UNIVERSE_SIZE</td>
<td>Constant UNIVERSE_SIZE of type int</td>
</tr>
<tr>
<td>APPNUM</td>
<td>Constant APPNUM of type int</td>
</tr>
<tr>
<td>LASTUSEDCODE</td>
<td>Constant LASTUSEDCODE of type int</td>
</tr>
<tr>
<td>WIN_BASE</td>
<td>Constant WIN_BASE of type int</td>
</tr>
<tr>
<td>WIN_SIZE</td>
<td>Constant WIN_SIZE of type int</td>
</tr>
<tr>
<td>WIN_DISP_UNIT</td>
<td>Constant WIN_DISP_UNIT of type int</td>
</tr>
<tr>
<td>WIN_CREATE_FLAVOR</td>
<td>Constant WIN_CREATE_FLAVOR of type int</td>
</tr>
<tr>
<td>WIN_FLAVOR</td>
<td>Constant WIN_FLAVOR of type int</td>
</tr>
<tr>
<td>WIN_MODEL</td>
<td>Constant WIN_MODEL of type int</td>
</tr>
<tr>
<td>SUCCESS</td>
<td>Constant SUCCESS of type int</td>
</tr>
<tr>
<td>ERR_LASTCODE</td>
<td>Constant ERR_LASTCODE of type int</td>
</tr>
<tr>
<td>ERR_COMM</td>
<td>Constant ERR_COMM of type int</td>
</tr>
<tr>
<td>ERR_GROUP</td>
<td>Constant ERR_GROUP of type int</td>
</tr>
<tr>
<td>ERR_TYPE</td>
<td>Constant ERR_TYPE of type int</td>
</tr>
<tr>
<td>ERR_REQUEST</td>
<td>Constant ERR_REQUEST of type int</td>
</tr>
<tr>
<td>ERR_OP</td>
<td>Constant ERR_OP of type int</td>
</tr>
<tr>
<td>ERR_BUFFER</td>
<td>Constant ERR_BUFFER of type int</td>
</tr>
<tr>
<td>ERR_COUNT</td>
<td>Constant ERR_COUNT of type int</td>
</tr>
<tr>
<td>Constant</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ERR_TAG</td>
<td>Constant ERR_TAG of type int</td>
</tr>
<tr>
<td>ERR_RANK</td>
<td>Constant ERR_RANK of type int</td>
</tr>
<tr>
<td>ERR_ROOT</td>
<td>Constant ERR_ROOT of type int</td>
</tr>
<tr>
<td>ERR_TRUNCATE</td>
<td>Constant ERR_TRUNCATE of type int</td>
</tr>
<tr>
<td>ERR_IN_STATUS</td>
<td>Constant ERR_IN_STATUS of type int</td>
</tr>
<tr>
<td>ERR_PENDING</td>
<td>Constant ERR_PENDING of type int</td>
</tr>
<tr>
<td>ERR_TOPOLOGY</td>
<td>Constant ERR_TOPOLOGY of type int</td>
</tr>
<tr>
<td>ERR_DIMS</td>
<td>Constant ERR_DIMS of type int</td>
</tr>
<tr>
<td>ERR_ARG</td>
<td>Constant ERR_ARG of type int</td>
</tr>
<tr>
<td>ERR_OTHER</td>
<td>Constant ERR_OTHER of type int</td>
</tr>
<tr>
<td>ERR_UNKNOWN</td>
<td>Constant ERR_UNKNOWN of type int</td>
</tr>
<tr>
<td>ERR_INTERN</td>
<td>Constant ERR_INTERN of type int</td>
</tr>
<tr>
<td>ERR_INFO</td>
<td>Constant ERR_INFO of type int</td>
</tr>
<tr>
<td>ERR_FILE</td>
<td>Constant ERR_FILE of type int</td>
</tr>
<tr>
<td>ERR_WIN</td>
<td>Constant ERR_WIN of type int</td>
</tr>
<tr>
<td>ERR_KEYVAL</td>
<td>Constant ERR_KEYVAL of type int</td>
</tr>
<tr>
<td>ERR_INFO_KEY</td>
<td>Constant ERR_INFO_KEY of type int</td>
</tr>
<tr>
<td>ERR_INFO_VALUE</td>
<td>Constant ERR_INFO_VALUE of type int</td>
</tr>
<tr>
<td>ERR_INFO_NOKEY</td>
<td>Constant ERR_INFO_NOKEY of type int</td>
</tr>
<tr>
<td>ERR_ACCESS</td>
<td>Constant ERR_ACCESS of type int</td>
</tr>
<tr>
<td>ERR_AMODE</td>
<td>Constant ERR_AMODE of type int</td>
</tr>
<tr>
<td>ERR_BAD_FILE</td>
<td>Constant ERR_BAD_FILE of type int</td>
</tr>
<tr>
<td>ERR_FILE_EXISTS</td>
<td>Constant ERR_FILE_EXISTS of type int</td>
</tr>
<tr>
<td>ERR_FILE_IN_USE</td>
<td>Constant ERR_FILE_IN_USE of type int</td>
</tr>
<tr>
<td>ERR_NO_SPACE</td>
<td>Constant ERR_NO_SPACE of type int</td>
</tr>
<tr>
<td>ERR_NO_SUCH_FILE</td>
<td>Constant ERR_NO_SUCH_FILE of type int</td>
</tr>
<tr>
<td>ERR_IO</td>
<td>Constant ERR_IO of type int</td>
</tr>
<tr>
<td>ERR_READ_ONLY</td>
<td>Constant ERR_READ_ONLY of type int</td>
</tr>
<tr>
<td>ERR_CONVERSION</td>
<td>Constant ERR_CONVERSION of type int</td>
</tr>
<tr>
<td>ERR_DUP_DATAREP</td>
<td>Constant ERR_DUP_DATAREP of type int</td>
</tr>
<tr>
<td>ERR_UNSUPPORTED_DATAREP</td>
<td>Constant ERR_UNSUPPORTED_DATAREP of type int</td>
</tr>
<tr>
<td>ERR_UNSUPPORTED_OPERATION</td>
<td>Constant ERR_UNSUPPORTED_OPERATION of type int</td>
</tr>
<tr>
<td>ERR_NAME</td>
<td>Constant ERR_NAME of type int</td>
</tr>
<tr>
<td>ERR_NO_MEM</td>
<td>Constant ERR_NO_MEM of type int</td>
</tr>
<tr>
<td>ERR_NOT_SAME</td>
<td>Constant ERR_NOT_SAME of type int</td>
</tr>
<tr>
<td>ERR_PORT</td>
<td>Constant ERR_PORT of type int</td>
</tr>
<tr>
<td>ERR_QUOTA</td>
<td>Constant ERR_QUOTA of type int</td>
</tr>
<tr>
<td>ERR_SERVICE</td>
<td>Constant ERR_SERVICE of type int</td>
</tr>
<tr>
<td>ERR_SPAWN</td>
<td>Constant ERR_SPAWN of type int</td>
</tr>
<tr>
<td>ERR_BASE</td>
<td>Constant ERR_BASE of type int</td>
</tr>
<tr>
<td>ERR_SIZE</td>
<td>Constant ERR_SIZE of type int</td>
</tr>
<tr>
<td>ERR_DISP</td>
<td>Constant ERR_DISP of type int</td>
</tr>
<tr>
<td>ERR_ASSERT</td>
<td>Constant ERR_ASSERT of type int</td>
</tr>
<tr>
<td>ERR_LOCKTYPE</td>
<td>Constant ERR_LOCKTYPE of type int</td>
</tr>
<tr>
<td>ERR_RMA_CONFLICT</td>
<td>Constant ERR_RMA_CONFLICT of type int</td>
</tr>
<tr>
<td>ERR_RMA_SYNC</td>
<td>Constant ERR_RMA_SYNC of type int</td>
</tr>
<tr>
<td>ERR_RMA_RANGE</td>
<td>Constant ERR_RMA_RANGE of type int</td>
</tr>
<tr>
<td>ERR_RMA_ATTACH</td>
<td>Constant ERR_RMA_ATTACH of type int</td>
</tr>
<tr>
<td>ERR_RMA_SHARED</td>
<td>Constant ERR_RMA_SHARED of type int</td>
</tr>
<tr>
<td>ERR_RMA_FLAVOR</td>
<td>Constant ERR_RMA_FLAVOR of type int</td>
</tr>
<tr>
<td>ORDER_C</td>
<td>Constant ORDER_C of type int</td>
</tr>
<tr>
<td>ORDER_F</td>
<td>Constant ORDER_F of type int</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER_FORTRAN</td>
<td>int</td>
</tr>
<tr>
<td>TYPECLASS_INTEGER</td>
<td>int</td>
</tr>
<tr>
<td>TYPECLASS_REAL</td>
<td>int</td>
</tr>
<tr>
<td>TYPECLASS_COMPLEX</td>
<td>int</td>
</tr>
<tr>
<td>DISTRIBUT_NONE</td>
<td>int</td>
</tr>
<tr>
<td>DISTRIBUT_BLOCK</td>
<td>int</td>
</tr>
<tr>
<td>DISTRIBUT_CYCLIC</td>
<td>int</td>
</tr>
<tr>
<td>DISTRIBUT_DFLT_DARG</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_NAMED</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_DUP</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_CONTIGUOUS</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_VECTOR</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_HVECTOR</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_INDEXED</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_HINDEXED</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_INDEXED_BLOCK</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_HINDEXED_BLOCK</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_STRUCT</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_SUBARRAY</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_DARRAY</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_RESIZED</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_F90_REAL</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_F90_COMPLEX</td>
<td>int</td>
</tr>
<tr>
<td>COMBINER_F90_INTEGER</td>
<td>int</td>
</tr>
<tr>
<td>IDENT</td>
<td>int</td>
</tr>
<tr>
<td>CONGRUENT</td>
<td>int</td>
</tr>
<tr>
<td>SIMILAR</td>
<td>int</td>
</tr>
<tr>
<td>UNEQUAL</td>
<td>int</td>
</tr>
<tr>
<td>CART</td>
<td>int</td>
</tr>
<tr>
<td>GRAPH</td>
<td>int</td>
</tr>
<tr>
<td>DIST_GRAPH</td>
<td>int</td>
</tr>
<tr>
<td>UNWEIGHTED</td>
<td>int</td>
</tr>
<tr>
<td>WEIGHTS_EMPTY</td>
<td>int</td>
</tr>
<tr>
<td>COMM_TYPE_SHARED</td>
<td>int</td>
</tr>
<tr>
<td>BSEND_OVERHEAD</td>
<td>int</td>
</tr>
<tr>
<td>WIN_FLAVOR_CREATE</td>
<td>int</td>
</tr>
<tr>
<td>WIN_FLAVOR_ALLOCATE</td>
<td>int</td>
</tr>
<tr>
<td>WIN_FLAVOR_DYNAMIC</td>
<td>int</td>
</tr>
<tr>
<td>WIN_FLAVOR_SHARED</td>
<td>int</td>
</tr>
<tr>
<td>WIN_SEPARATE</td>
<td>int</td>
</tr>
<tr>
<td>WIN_UNIFIED</td>
<td>int</td>
</tr>
<tr>
<td>MODE_NOCHECK</td>
<td>int</td>
</tr>
<tr>
<td>MODE_NOSTORE</td>
<td>int</td>
</tr>
<tr>
<td>MODE_NOPUT</td>
<td>int</td>
</tr>
<tr>
<td>MODE_NOPRECEDE</td>
<td>int</td>
</tr>
<tr>
<td>MODE_NOSUCCEED</td>
<td>int</td>
</tr>
<tr>
<td>LOCK_EXCLUSIVE</td>
<td>int</td>
</tr>
<tr>
<td>LOCK_SHARED</td>
<td>int</td>
</tr>
<tr>
<td>MODE_RDONLY</td>
<td>int</td>
</tr>
<tr>
<td>MODE_WRONLY</td>
<td>int</td>
</tr>
<tr>
<td>MODE_RDWR</td>
<td>int</td>
</tr>
<tr>
<td>MODE_CREATE</td>
<td>int</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_EXCL</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_DELETE_ON_CLOSE</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_UNIQUE_OPEN</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_SEQUENTIAL</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_APPEND</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>SEEK_SET</td>
<td>Constant SEEK_SET of type int</td>
</tr>
<tr>
<td>SEEK_CUR</td>
<td>Constant SEEK_CUR of type int</td>
</tr>
<tr>
<td>SEEK_END</td>
<td>Constant SEEK_END of type int</td>
</tr>
<tr>
<td>DISPLACEMENT_CURRENT</td>
<td>Constant DISPLACEMENT_CURRENT of type int</td>
</tr>
<tr>
<td>DISC_CUR</td>
<td>Constant DISC_CUR of type int</td>
</tr>
<tr>
<td>THREAD_SINGLE</td>
<td>Constant THREAD_SINGLE of type int</td>
</tr>
<tr>
<td>THREAD_FUNNELED</td>
<td>Constant THREAD_FUNNELED of type int</td>
</tr>
<tr>
<td>THREAD_SERIALIZED</td>
<td>Constant THREAD_SERIALIZED of type int</td>
</tr>
<tr>
<td>THREAD_MULTIPLE</td>
<td>Constant THREAD_MULTIPLE of type int</td>
</tr>
<tr>
<td>VERSION</td>
<td>Constant VERSION of type int</td>
</tr>
<tr>
<td>SUBVERSION</td>
<td>Constant SUBVERSION of type int</td>
</tr>
<tr>
<td>MAX_PROCESSOR_NAME</td>
<td>Constant MAX_PROCESSOR_NAME of type int</td>
</tr>
<tr>
<td>MAX_ERROR_STRING</td>
<td>Constant MAX_ERROR_STRING of type int</td>
</tr>
<tr>
<td>MAX_PORT_NAME</td>
<td>Constant MAX_PORT_NAME of type int</td>
</tr>
<tr>
<td>MAX_INFO_KEY</td>
<td>Constant MAX_INFO_KEY of type int</td>
</tr>
<tr>
<td>MAX_INFO_VAL</td>
<td>Constant MAX_INFO_VAL of type int</td>
</tr>
<tr>
<td>MAX_OBJECT_NAME</td>
<td>Constant MAX_OBJECT_NAME of type int</td>
</tr>
<tr>
<td>MAX_DATAREP_STRING</td>
<td>Constant MAX_DATAREP_STRING of type int</td>
</tr>
<tr>
<td>MAX_LIBRARY_VERSION_STRING</td>
<td>Constant MAX_LIBRARY_VERSION_STRING of type int</td>
</tr>
<tr>
<td>DATATYPE_NULL</td>
<td>Object DATATYPE_NULL of type Datatype</td>
</tr>
<tr>
<td>PACKED</td>
<td>Object PACKED of type Datatype</td>
</tr>
<tr>
<td>BYTE</td>
<td>Object BYTE of type Datatype</td>
</tr>
<tr>
<td>AINT</td>
<td>Object AINT of type Datatype</td>
</tr>
<tr>
<td>OFFSET</td>
<td>Object OFFSET of type Datatype</td>
</tr>
<tr>
<td>COUNT</td>
<td>Object COUNT of type Datatype</td>
</tr>
<tr>
<td>CHAR</td>
<td>Object CHAR of type Datatype</td>
</tr>
<tr>
<td>WCHAR</td>
<td>Object WCHAR of type Datatype</td>
</tr>
<tr>
<td>SIGNED_CHAR</td>
<td>Object SIGNED_CHAR of type Datatype</td>
</tr>
<tr>
<td>SHORT</td>
<td>Object SHORT of type Datatype</td>
</tr>
<tr>
<td>INT</td>
<td>Object INT of type Datatype</td>
</tr>
<tr>
<td>LONG</td>
<td>Object LONG of type Datatype</td>
</tr>
<tr>
<td>LONG_LONG</td>
<td>Object LONG_LONG of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_CHAR</td>
<td>Object UNSIGNED_CHAR of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_SHORT</td>
<td>Object UNSIGNED_SHORT of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED</td>
<td>Object UNSIGNED of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_LONG</td>
<td>Object UNSIGNED_LONG of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_LONG_LONG</td>
<td>Object UNSIGNED_LONG_LONG of type Datatype</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Object FLOAT of type Datatype</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>Object DOUBLE of type Datatype</td>
</tr>
<tr>
<td>LONG_DOUBLE</td>
<td>Object LONG_DOUBLE of type Datatype</td>
</tr>
<tr>
<td>C_BOOL</td>
<td>Object C_BOOL of type Datatype</td>
</tr>
<tr>
<td>INT8_T</td>
<td>Object INT8_T of type Datatype</td>
</tr>
<tr>
<td>INT16_T</td>
<td>Object INT16_T of type Datatype</td>
</tr>
<tr>
<td>INT32_T</td>
<td>Object INT32_T of type Datatype</td>
</tr>
<tr>
<td>INT64_T</td>
<td>Object INT64_T of type Datatype</td>
</tr>
<tr>
<td>UINT8_T</td>
<td>Object UINT8_T of type Datatype</td>
</tr>
<tr>
<td>UINT16_T</td>
<td>Object UINT16_T of type Datatype</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UINT32_T</td>
<td>Object UINT32_T of type Datatype</td>
</tr>
<tr>
<td>UINT64_T</td>
<td>Object UINT64_T of type Datatype</td>
</tr>
<tr>
<td>C_COMPLEX</td>
<td>Object C_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>C_FLOAT_COMPLEX</td>
<td>Object C_FLOAT_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>C_DOUBLE_COMPLEX</td>
<td>Object C_DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>C_LONG_DOUBLE_COMPLEX</td>
<td>Object C_LONG_DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>CXX_BOOL</td>
<td>Object CXX_BOOL of type Datatype</td>
</tr>
<tr>
<td>CXX_FLOAT_COMPLEX</td>
<td>Object CXX_FLOAT_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>CXX_DOUBLE_COMPLEX</td>
<td>Object CXX_DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>CXX_LONG_DOUBLE_COMPLEX</td>
<td>Object CXX_LONG_DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>SHORT_INT</td>
<td>Object SHORT_INT of type Datatype</td>
</tr>
<tr>
<td>INT_INT</td>
<td>Object INT_INT of type Datatype</td>
</tr>
<tr>
<td>TWOINT</td>
<td>Object TWOINT of type Datatype</td>
</tr>
<tr>
<td>LONG_INT</td>
<td>Object LONG_INT of type Datatype</td>
</tr>
<tr>
<td>FLOAT_INT</td>
<td>Object FLOAT_INT of type Datatype</td>
</tr>
<tr>
<td>DOUBLE_INT</td>
<td>Object DOUBLE_INT of type Datatype</td>
</tr>
<tr>
<td>LONG_DOUBLE_INT</td>
<td>Object LONG_DOUBLE_INT of type Datatype</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>Object CHARACTER of type Datatype</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>Object LOGICAL of type Datatype</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Object INTEGER of type Datatype</td>
</tr>
<tr>
<td>REAL</td>
<td>Object REAL of type Datatype</td>
</tr>
<tr>
<td>DOUBLE_PRECISION</td>
<td>Object DOUBLE_PRECISION of type Datatype</td>
</tr>
<tr>
<td>COMPLEX</td>
<td>Object COMPLEX of type Datatype</td>
</tr>
<tr>
<td>DOUBLE_COMPLEX</td>
<td>Object DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>LOGICAL1</td>
<td>Object LOGICAL1 of type Datatype</td>
</tr>
<tr>
<td>LOGICAL2</td>
<td>Object LOGICAL2 of type Datatype</td>
</tr>
<tr>
<td>LOGICAL4</td>
<td>Object LOGICAL4 of type Datatype</td>
</tr>
<tr>
<td>LOGICAL8</td>
<td>Object LOGICAL8 of type Datatype</td>
</tr>
<tr>
<td>INTEGER1</td>
<td>Object INTEGER1 of type Datatype</td>
</tr>
<tr>
<td>INTEGER2</td>
<td>Object INTEGER2 of type Datatype</td>
</tr>
<tr>
<td>INTEGER4</td>
<td>Object INTEGER4 of type Datatype</td>
</tr>
<tr>
<td>INTEGER8</td>
<td>Object INTEGER8 of type Datatype</td>
</tr>
<tr>
<td>INTEGER16</td>
<td>Object INTEGER16 of type Datatype</td>
</tr>
<tr>
<td>REAL2</td>
<td>Object REAL2 of type Datatype</td>
</tr>
<tr>
<td>REAL4</td>
<td>Object REAL4 of type Datatype</td>
</tr>
<tr>
<td>REAL8</td>
<td>Object REAL8 of type Datatype</td>
</tr>
<tr>
<td>REAL16</td>
<td>Object REAL16 of type Datatype</td>
</tr>
<tr>
<td>COMPLEX4</td>
<td>Object COMPLEX4 of type Datatype</td>
</tr>
<tr>
<td>COMPLEX8</td>
<td>Object COMPLEX8 of type Datatype</td>
</tr>
<tr>
<td>COMPLEX16</td>
<td>Object COMPLEX16 of type Datatype</td>
</tr>
<tr>
<td>COMPLEX32</td>
<td>Object COMPLEX32 of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_INT</td>
<td>Object UNSIGNED_INT of type Datatype</td>
</tr>
<tr>
<td>SIGNED_SHORT</td>
<td>Object SIGNED_SHORT of type Datatype</td>
</tr>
<tr>
<td>SIGNED_INT</td>
<td>Object SIGNED_INT of type Datatype</td>
</tr>
<tr>
<td>SIGNED_LONG</td>
<td>Object SIGNED_LONG of type Datatype</td>
</tr>
<tr>
<td>SIGNED_LONG_LONG</td>
<td>Object SIGNED_LONG_LONG of type Datatype</td>
</tr>
<tr>
<td>BOOL</td>
<td>Object BOOL of type Datatype</td>
</tr>
<tr>
<td>SINT8_T</td>
<td>Object SINT8_T of type Datatype</td>
</tr>
<tr>
<td>SINT16_T</td>
<td>Object SINT16_T of type Datatype</td>
</tr>
<tr>
<td>SINT32_T</td>
<td>Object SINT32_T of type Datatype</td>
</tr>
<tr>
<td>SINT64_T</td>
<td>Object SINT64_T of type Datatype</td>
</tr>
<tr>
<td>F_BOOL</td>
<td>Object F_BOOL of type Datatype</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_INT</td>
<td>Object F_INT of type Datatype</td>
</tr>
<tr>
<td>F_FLOAT</td>
<td>Object F_FLOAT of type Datatype</td>
</tr>
<tr>
<td>F_DOUBLE</td>
<td>Object F_DOUBLE of type Datatype</td>
</tr>
<tr>
<td>F_COMPLEX</td>
<td>Object F_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>F_FLOAT_COMPLEX</td>
<td>Object F_FLOAT_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>F_DOUBLE_COMPLEX</td>
<td>Object F_DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>REQUEST_NULL</td>
<td>Object REQUEST_NULL of type Request</td>
</tr>
<tr>
<td>MESSAGE_NULL</td>
<td>Object MESSAGE_NULL of type Message</td>
</tr>
<tr>
<td>MESSAGE_NO_PROC</td>
<td>Object MESSAGE_NO_PROC of type Message</td>
</tr>
<tr>
<td>OP_NULL</td>
<td>Object OP_NULL of type Op</td>
</tr>
<tr>
<td>MAX</td>
<td>Object MAX of type Op</td>
</tr>
<tr>
<td>MIN</td>
<td>Object MIN of type Op</td>
</tr>
<tr>
<td>SUM</td>
<td>Object SUM of type Op</td>
</tr>
<tr>
<td>PROD</td>
<td>Object PROD of type Op</td>
</tr>
<tr>
<td>LAND</td>
<td>Object LAND of type Op</td>
</tr>
<tr>
<td>BAND</td>
<td>Object BAND of type Op</td>
</tr>
<tr>
<td>LOR</td>
<td>Object LOR of type Op</td>
</tr>
<tr>
<td>BOR</td>
<td>Object BOR of type Op</td>
</tr>
<tr>
<td>LXOR</td>
<td>Object LXOR of type Op</td>
</tr>
<tr>
<td>BXOR</td>
<td>Object BXOR of type Op</td>
</tr>
<tr>
<td>MAXLOC</td>
<td>Object MAXLOC of type Op</td>
</tr>
<tr>
<td>MINLOC</td>
<td>Object MINLOC of type Op</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Object REPLACE of type Op</td>
</tr>
<tr>
<td>NO_OP</td>
<td>Object NO_OP of type Op</td>
</tr>
<tr>
<td>GROUP_NULL</td>
<td>Object GROUP_NULL of type Group</td>
</tr>
<tr>
<td>GROUP_EMPTY</td>
<td>Object GROUP_EMPTY of type Group</td>
</tr>
<tr>
<td>INFO_NULL</td>
<td>Object INFO_NULL of type Info</td>
</tr>
<tr>
<td>INFO_ENV</td>
<td>Object INFO_ENV of type Info</td>
</tr>
<tr>
<td>ERRHANDLER_NULL</td>
<td>Object ERRHANDLER_NULL of type Errhandler</td>
</tr>
<tr>
<td>ERRORS_RETURN</td>
<td>Object ERRORS_RETURN of type Errhandler</td>
</tr>
<tr>
<td>ERRORS_ARE_FATAL</td>
<td>Object ERRORS_ARE_FATAL of type Errhandler</td>
</tr>
<tr>
<td>COMM_NULL</td>
<td>Object COMM_NULL of type Comm</td>
</tr>
<tr>
<td>COMM_SELF</td>
<td>Object COMM_SELF of type Intracomm</td>
</tr>
<tr>
<td>COMM_WORLD</td>
<td>Object COMM_WORLD of type Intracomm</td>
</tr>
<tr>
<td>WIN_NULL</td>
<td>Object WIN_NULL of type Win</td>
</tr>
<tr>
<td>FILE_NULL</td>
<td>Object FILE_NULL of type File</td>
</tr>
<tr>
<td>pickle</td>
<td>Object pickle of type Pickle</td>
</tr>
</tbody>
</table>

6 mpi4py.typing

New in version 4.0.0.

This module provides type aliases used to add type hints to the various functions and methods within the MPI module.

See also:

Module typing
  Documentation of the typing standard module.
Types Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupportsBuffer</td>
<td>Python buffer protocol.</td>
</tr>
<tr>
<td>SupportsDLPack</td>
<td>DLPack data interchange protocol.</td>
</tr>
<tr>
<td>SupportsCAI</td>
<td>CUDA Array Interface (CAI) protocol.</td>
</tr>
<tr>
<td>Buffer</td>
<td>Buffer-like object.</td>
</tr>
<tr>
<td>Bottom</td>
<td>Start of the address range.</td>
</tr>
<tr>
<td>InPlace</td>
<td>In-place buffer argument.</td>
</tr>
<tr>
<td>Aint</td>
<td>Address-sized integral type.</td>
</tr>
<tr>
<td>Count</td>
<td>Integral type for counts.</td>
</tr>
<tr>
<td>Displ</td>
<td>Integral type for displacements.</td>
</tr>
<tr>
<td>Offset</td>
<td>Integral type for offsets.</td>
</tr>
<tr>
<td>TypeSpec</td>
<td>Datatype specification.</td>
</tr>
<tr>
<td>BufSpec</td>
<td>Buffer specification.</td>
</tr>
<tr>
<td>BufSpecB</td>
<td>Buffer specification (block).</td>
</tr>
<tr>
<td>BufSpecV</td>
<td>Buffer specification (vector).</td>
</tr>
<tr>
<td>BufSpecW</td>
<td>Buffer specification (generalized).</td>
</tr>
<tr>
<td>TargetSpec</td>
<td>Target specification.</td>
</tr>
</tbody>
</table>

Types Documentation

```python
mpi4py.typing.SupportsBuffer = <class 'mpi4py.typing.SupportsBuffer'>
Python buffer protocol.

See also:
Buffer Protocol
alias of mpi4py.typing.SupportsBuffer

mpi4py.typing.SupportsDLPack = <class 'mpi4py.typing.SupportsDLPack'>
DLPack data interchange protocol.

See also:
Python Specification for DLPack
alias of mpi4py.typing.SupportsDLPack

mpi4py.typing.SupportsCAI = <class 'mpi4py.typing.SupportsCAI'>
CUDA Array Interface (CAI) protocol.

See also:
CUDA Array Interface (Version 3)
alias of mpi4py.typing.SupportsCAI

mpi4py.typing.Buffer
Buffer-like object.
alias of Union[SupportsBuffer, SupportsDLPack, SupportsCAI]

mpi4py.typing.Bottom
Start of the address range.
alias of Optional[BottomType]
```
mpi4py.typing.InPlace
In-place buffer argument.
    alias of Optional[InPlaceType]

mpi4py.typing.Aint = <class 'numbers.Integral'>
Address-sized integral type.
    alias of numbers.Integral

mpi4py.typing.Count = <class 'numbers.Integral'>
Integral type for counts.
    alias of numbers.Integral

mpi4py.typing.Displ = <class 'numbers.Integral'>
Integral type for displacements.
    alias of numbers.Integral

mpi4py.typing.Offset = <class 'numbers.Integral'>
Integral type for offsets.
    alias of numbers.Integral

mpi4py.typing.TypeSpec
Datatype specification.
    alias of Union[Datatype, str]

mpi4py.typing.BufSpec
Buffer specification.
    • Buffer
    • Tuple[Buffer, Count]
    • Tuple[Buffer, TypeSpec]
    • Tuple[Buffer, Count, TypeSpec]
    • Tuple[Bottom, Count, Datatype]
    alias of  Union[SupportsBuffer, SupportsDLPack, SupportsCAI, Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Integral], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Union[Datatype, str]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Integral, Union[Datatype, str]], Tuple[Optional[BottomType], Integral, Datatype], List]

mpi4py.typing.BufSpecB
Buffer specification (block).
    • Buffer
    • Tuple[Buffer, Count]
    • Tuple[Buffer, TypeSpec]
    • Tuple[Buffer, Count, TypeSpec]
    alias of  Union[SupportsBuffer, SupportsDLPack, SupportsCAI, Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Integral], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Union[Datatype, str]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Integral, Union[Datatype, str]], List]
Buffer specification (vector).

- `Buffer`
- `Tuple[Buffer, Sequence[Count]]`
- `Tuple[Buffer, Tuple[Sequence[Count], Sequence[Displ]]]`
- `Tuple[Buffer, TypeSpec]`
- `Tuple[Buffer, Sequence[Count], TypeSpec]`
- `Tuple[Buffer, Tuple[Sequence[Count], Sequence[Displ]], TypeSpec]`
- `Tuple[Buffer, Sequence[Count], Sequence[Displ], TypeSpec]`
- `Tuple[Buffer, Sequence[Count], Sequence[Displ], Datatype]`
- `Tuple[Buffer, Sequence[Count], Sequence[Displ], Datatype]`

alias of `Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Sequence[Integral]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Tuple[Sequence[Integral], Sequence[Integral]]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Union[Datatype, str]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Sequence[Integral], Union[Datatype, str]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Tuple[Sequence[Integral], Sequence[Integral]], Union[Datatype, str]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Sequence[Integral], Sequence[Integral], Datatype], Tuple[Optional[BottomType], Sequence[Integral], Sequence[Integral], Datatype], List`

Buffer specification (generalized).

- `Tuple[Buffer, Sequence[Datatype]]`
- `Tuple[Buffer, Tuple[Sequence[Count], Sequence[Displ]], Sequence[Datatype]]`
- `Tuple[Buffer, Sequence[Count], Sequence[Displ], Sequence[Datatype]]`
- `Tuple[Buffer, Sequence[Count], Sequence[Displ], Sequence[Datatype]]`

alias of `Union[Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Sequence[Datatype]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Tuple[Sequence[Integral], Sequence[Integral]], Sequence[Datatype]], Tuple[Union[SupportsBuffer, SupportsDLPack, SupportsCAI], Tuple[Sequence[Integral], Sequence[Integral], Sequence[Datatype]], Tuple[Optional[BottomType], Sequence[Integral], Sequence[Datatype]], Tuple[Optional[BottomType], Sequence[Integral], Sequence[Datatype]], Tuple[Optional[BottomType], Sequence[Integral], Sequence[Datatype]], List]`

Target specification.

- `Displ`
- `Tuple[()]`
- `Tuple[Displ]`
- `Tuple[Displ, Count]`
7 mpi4py.futures

New in version 3.0.0.

This package provides a high-level interface for asynchronously executing callables on a pool of worker processes using MPI for inter-process communication.

The mpi4py.futures package is based on concurrent.futures from the Python standard library. More precisely, mpi4py.futures provides the MPIPoolExecutor class as a concrete implementation of the abstract class Executor. The submit() interface schedules a callable to be executed asynchronously and returns a Future object representing the execution of the callable. Future instances can be queried for the call result or exception. Sets of Future instances can be passed to the wait() and as_completed() functions.

See also:

Module concurrent.futures
Documentation of the concurrent.futures standard module.

7.1 MPIPoolExecutor

The MPIPoolExecutor class uses a pool of MPI processes to execute calls asynchronously. By performing computations in separate processes, it allows to side-step the global interpreter lock but also means that only picklable objects can be executed and returned. The __main__ module must be importable by worker processes, thus MPIPoolExecutor instances may not work in the interactive interpreter.

MPIPoolExecutor takes advantage of the dynamic process management features introduced in the MPI-2 standard. In particular, the MPI.Intracomm.Spawn method of MPI.COMM_SELF is used in the master (or parent) process to spawn new worker (or child) processes running a Python interpreter. The master process uses a separate thread (one for each MPIPoolExecutor instance) to communicate back and forth with the workers. The worker processes serve the execution of tasks in the main (and only) thread until they are signaled for completion.

Note: The worker processes must import the main script in order to unpickle any callable defined in the __main__ module and submitted from the master process. Furthermore, the callables may need access to other global variables. At the worker processes, mpi4py.futures executes the main script code (using the runpy module) under the __worker__ namespace to define the __main__ module. The __main__ and __worker__ modules are added to sys.modules (both at the master and worker processes) to ensure proper pickling and unpickling.

Warning: During the initial import phase at the workers, the main script cannot create and use new MPIPoolExecutor instances. Otherwise, each worker would attempt to spawn a new pool of workers, leading to infinite recursion. mpi4py.futures detects such recursive attempts to spawn new workers and aborts the MPI execution environment. As the main script code is run under the __worker__ namespace, the easiest way to avoid spawn recursion is using the idiom if __name__ == '__main__': ... in the main script.

class mpi4py.futures.MPIPoolExecutor(max_workers=None, initializer=None, initargs=(), **kwargs)

An Executor subclass that executes calls asynchronously using a pool of at most max_workers processes. If
max_workers is None or not given, its value is determined from the MPI4PY_FUTURES_MAX_WORKERS environment variable if set, or the MPI universe size if set, otherwise a single worker process is spawned. If max_workers is lower than or equal to 0, then a ValueError will be raised.

initializer is an optional callable that is called at the start of each worker process before executing any tasks; initargs is a tuple of arguments passed to the initializer. If initializer raises an exception, all pending tasks and any attempt to submit new tasks to the pool will raise a BrokenExecutor exception.

Other parameters:

* python_exe: Path to the Python interpreter executable used to spawn worker processes, otherwise sys.executable is used.

* python_args: list or iterable with additional command line flags to pass to the Python executable. Command line flags determined from inspection of sys.flags, sys.warnoptions and sys._xoptions in are passed unconditionally.

* mpi_info: dict or iterable yielding (key, value) pairs. These (key, value) pairs are passed (through an MPI_Info object) to the MPI.Intracomm.Spawn call used to spawn worker processes. This mechanism allows telling the MPI runtime system where and how to start the processes. Check the documentation of the backend MPI implementation about the set of keys it interprets and the corresponding format for values.

* globals: dict or iterable yielding (name, value) pairs to initialize the main module namespace in worker processes.

* main: If set to False, do not import the __main__ module in worker processes. Setting main to False prevents worker processes from accessing definitions in the parent __main__ namespace.

* path: list or iterable with paths to append to sys.path in worker processes to extend the module search path.

* wdir: Path to set the current working directory in worker processes using os.chdir(). The initial working directory is set by the MPI implementation. Quality MPI implementations should honor a wdir info key passed through mpi_info, although such feature is not mandatory.

* env: dict or iterable yielding (name, value) pairs with environment variables to update os.environ in worker processes. The initial environment is set by the MPI implementation. MPI implementations may allow setting the initial environment through mpi_info, however such feature is not required nor recommended by the MPI standard.

* use_pkl5: If set to True, use pickle5 with out-of-band buffers for interprocess communication. If use_pkl5 is set to None or not given, its value is determined from the MPI4PY_FUTURES_USE_PKL5 environment variable. Using pickle5 with out-of-band buffers may benefit applications dealing with large buffer-like objects like NumPy arrays. See mpi4py.util.pkl5 for additional information.

* backoff: float value specifying the maximum number of seconds a worker thread or process suspends execution with time.sleep() while idle-waiting. If not set, its value is determined from the MPI4PY_FUTURES_BACKOFF environment variable if set, otherwise the default value of 0.001 seconds is used. Lower values will reduce latency and increase execution throughput for very short-lived tasks, albeit at the expense of spinning CPU cores and increased energy consumption.

submit(func, *args, **kwargs)

Schedule the callable, func, to be executed as func(*args, **kwargs) and returns a Future object representing the execution of the callable.

executor = MPIPoolExecutor(max_workers=1)
future = executor.submit(pow, 321, 1234)
print(future.result())
map(func, *iterables, timeout=None, chunksize=1, **kwargs)

Equivalent to map(func, *iterables) except func is executed asynchronously and several calls to func may be made concurrently, out-of-order, in separate processes. The returned iterator raises a TimeoutError if __next__() is called and the result isn’t available after timeout seconds from the original call to map(). timeout can be an int or a float. If timeout is not specified or None, there is no limit to the wait time. If a call raises an exception, then that exception will be raised when its value is retrieved from the iterator. This method chops iterables into a number of chunks which it submits to the pool as separate tasks. The (approximate) size of these chunks can be specified by setting chunksize to a positive integer. For very long iterables, using a large value for chunksize can significantly improve performance compared to the default size of one. By default, the returned iterator yields results in-order, waiting for successive tasks to complete. This behavior can be changed by passing the keyword argument unordered as True, then the result iterator will yield a result as soon as any of the tasks complete.

```python
executor = MPIPoolExecutor(max_workers=3)
for result in executor.map(pow, [2]**32, range(32)):
    print(result)
```

starmap(func, iterable, timeout=None, chunksize=1, **kwargs)

Equivalent to itertools.starmap(func, iterable). Used instead of map() when argument parameters are already grouped in tuples from a single iterable (the data has been “pre-zipped”). map(func, *iterables) is equivalent to starmap(func, zip(*iterables)).

```python
executor = MPIPoolExecutor(max_workers=3)
iterable = ((2, n) for n in range(32))
for result in executor.starmap(pow, iterable):
    print(result)
```

shutdown(wait=True, cancel_futures=False)

Signal the executor that it should free any resources that it is using when the currently pending futures are done executing. Calls to submit() and map() made after shutdown() will raise RuntimeError.

If wait is True then this method will not return until all the pending futures are done executing and the resources associated with the executor have been freed. If wait is False then this method will return immediately and the resources associated with the executor will be freed when all pending futures are done executing. Regardless of the value of wait, the entire Python program will not exit until all pending futures are done executing.

If cancel_futures is True, this method will cancel all pending futures that the executor has not started running. Any futures that are completed or running won’t be cancelled, regardless of the value of cancel_futures.

You can avoid having to call this method explicitly if you use the with statement, which will shutdown the executor instance (waiting as if shutdown() were called with wait set to True).

```python
import time
with MPIPoolExecutor(max_workers=1) as executor:
    future = executor.submit(time.sleep, 2)
assert future.done()
```

bootup(wait=True)

Signal the executor that it should allocate eagerly any required resources (in particular, MPI worker processes). If wait is True, then bootup() will not return until the executor resources are ready to process submissions. Resources are automatically allocated in the first call to submit(), thus calling bootup() explicitly is seldom needed.
**num_workers**
Number or worker processes in the pool.

**MPI4PY_FUTURES_MAX_WORKERS**
If the `max_workers` parameter to `MPIPoolExecutor` is `None` or not given, the `MPI4PY_FUTURES_MAX_WORKERS` environment variable provides a fallback value for the maximum number of MPI worker processes to spawn.
New in version 3.1.0.

**MPI4PY_FUTURES_USE_PKL5**
If the `use_pkl5` keyword argument to `MPIPoolExecutor` is `None` or not given, the `MPI4PY_FUTURES_USE_PKL5` environment variable provides a fallback value for whether the executor should use pickle5 with out-of-band buffers for interprocess communication. Accepted values are 0 and 1 (interpreted as `False` and `True`, respectively), and strings specifying a YAML boolean value (case-insensitive). Using pickle5 with out-of-band buffers may benefit applications dealing with large buffer-like objects like NumPy arrays. See `mpi4py.util.pkl5` for additional information.
New in version 4.0.0.

**MPI4PY_FUTURES_BACKOFF**
If the `backoff` keyword argument to `MPIPoolExecutor` is not given, the `MPI4PY_FUTURES_BACKOFF` environment variable can be set to a `float` value specifying the maximum number of seconds a worker thread or process suspends execution with `time.sleep()` while idle-waiting. If not set, the default backoff value is 0.001 seconds. Lower values will reduce latency and increase execution throughput for very short-lived tasks, albeit at the expense of spinning CPU cores and increased energy consumption.
New in version 4.0.0.

**Note:** As the master process uses a separate thread to perform MPI communication with the workers, the backend MPI implementation should provide support for `MPI.THREAD_MULTIPLE`. However, some popular MPI implementations do not support yet concurrent MPI calls from multiple threads. Additionally, users may decide to initialize MPI with a lower level of thread support. If the level of thread support in the backend MPI is less than `MPI.THREAD_MULTIPLE`, `mpi4py.futures` will use a global lock to serialize MPI calls. If the level of thread support is less than `MPI.THREAD_SERIALIZED`, `mpi4py.futures` will emit a `RuntimeWarning`.

**Warning:** If the level of thread support in the backend MPI is less than `MPI.THREAD_SERIALIZED` (i.e, it is either `MPI.THREAD_SINGLE` or `MPI.THREAD_FUNNELED`), in theory `mpi4py.futures` cannot be used. Rather than raising an exception, `mpi4py.futures` emits a warning and takes a “cross-fingers” attitude to continue execution in the hope that serializing MPI calls with a global lock will actually work.

### 7.2 MPICommExecutor
Legacy MPI-1 implementations (as well as some vendor MPI-2 implementations) do not support the dynamic process management features introduced in the MPI-2 standard. Additionally, job schedulers and batch systems in supercomputing facilities may pose additional complications to applications using the `MPI_Comm_spawn()` routine.

With these issues in mind, `mpi4py.futures` supports an additional, more traditional, SPMD-like usage pattern requiring MPI-1 calls only. Python applications are started the usual way, e.g., using the `mpiexec` command. Python code should make a collective call to the `MPICommExecutor` context manager to partition the set of MPI processes within a MPI communicator in one master processes and many workers processes. The master process gets access to an `MPIPoolExecutor` instance to submit tasks. Meanwhile, the worker process follow a different execution path and team-up to execute the tasks submitted from the master.
Besides alleviating the lack of dynamic process management features in legacy MPI-1 or partial MPI-2 implementations, the `MPICommExecutor` context manager may be useful in classic MPI-based Python applications willing to take advantage of the simple, task-based, master/worker approach available in the `mpi4py.futures` package.

```python
class mpi4py.futures.MPICommExecutor(comm=None, root=0):
    Context manager for MPIPoolExecutor. This context manager splits a MPI (intra)communicator `comm` (defaults to `MPI.COMM_WORLD` if not provided or `None`) in two disjoint sets: a single master process (with rank `root` in `comm`) and the remaining worker processes. These sets are then connected through an intercommunicator. The target of the `with` statement is assigned either an `MPIPoolExecutor` instance (at the master) or `None` (at the workers).

    from mpi4py import MPI
    from mpi4py.futures import MPICommExecutor

    with MPICommExecutor(MPI.COMM_WORLD, root=0) as executor:
        if executor is not None:
            future = executor.submit(abs, -42)
            assert future.result() == 42
            answer = set(executor.map(abs, [-42, 42]))
            assert answer == {42}
```

**Warning:** If `MPICommExecutor` is passed a communicator of size one (e.g., `MPI.COMM_SELF`), then the executor instance assigned to the target of the `with` statement will execute all submitted tasks in a single worker thread, thus ensuring that task execution still progress asynchronously. However, the GIL will prevent the main and worker threads from running concurrently in multicore processors. Moreover, the thread context switching may harm noticeably the performance of CPU-bound tasks. In case of I/O-bound tasks, the GIL is not usually an issue, however, as a single worker thread is used, it progress one task at a time. We advice against using `MPICommExecutor` with communicators of size one and suggest refactoring your code to use instead a `ThreadPoolExecutor`.

### 7.3 Command line

Recalling the issues related to the lack of support for dynamic process management features in MPI implementations, `mpi4py.futures` supports an alternative usage pattern where Python code (either from scripts, modules, or zip files) is run under command line control of the `mpi4py.futures` package by passing `-m mpi4py.futures` to the `python` executable. The `mpi4py.futures` invocation should be passed a `pyfile` path to a script (or a zipfile/directory containing a `__main__.py` file). Additionally, `mpi4py.futures` accepts `-m mod` to execute a module named `mod`, `-c cmd` to execute a command string `cmd`, or even `-r` to read commands from standard input (`sys.stdin`). Summarizing, `mpi4py.futures` can be invoked in the following ways:

- `$ mpiexec -n numprocs python -m mpi4py.futures pyfile [arg] ...`
- `$ mpiexec -n numprocs python -m mpi4py.futures -m mod [arg] ...`
- `$ mpiexec -n numprocs python -m mpi4py.futures -c cmd [arg] ...`
- `$ mpiexec -n numprocs python -m mpi4py.futures - [arg] ...`

Before starting the main script execution, `mpi4py.futures` splits `MPI.COMM_WORLD` in one master (the process with rank 0 in `MPI.COMM_WORLD`) and `numprocs - 1` workers and connects them through an MPI intercommunicator. Afterwards, the master process proceeds with the execution of the user script code, which eventually creates `MPIPoolExecutor` instances to submit tasks. Meanwhile, the worker processes follow a different execution path to serve the master. Upon successful termination of the main script at the master, the entire MPI execution environment exists gracefully. In case of any unhandled exception in the main script, the master process calls `MPI.COMM_WORLD.Abort(1)` to prevent deadlocks and force termination of entire MPI execution environment.
Warning: Running scripts under command line control of *mpi4py.futures* is quite similar to executing a single-process application that spawn additional workers as required. However, there is a very important difference users should be aware of. All *MPIPoolExecutor* instances created at the master will share the pool of workers. Tasks submitted at the master from many different executors will be scheduled for execution in random order as soon as a worker is idle. Any executor can easily starve all the workers (e.g., by calling *MPIPoolExecutor.map()* with long iterables). If that ever happens, submissions from other executors will not be serviced until free workers are available.

See also:

Command line

- Documentation on Python command line interface.

### 7.4 Parallel tasks

The *mpi4py.futures* package favors an embarrassingly parallel execution model involving a series of sequential tasks independent of each other and executed asynchronously. Albeit unnatural, *MPIPoolExecutor* can still be used for handling workloads involving parallel tasks, where worker processes communicate and coordinate each other via MPI.

```python
mpi4py.futures.get_comm_workers()
```

Access an intracommunicator grouping MPI worker processes.

Executing parallel tasks with *mpi4py.futures* requires following some rules, cf. highlighted lines in example *cpi.py*:

- Use *MPIPoolExecutor.num_workers* to determine the number of worker processes in the executor and **submit exactly one callable per worker process** using the *MPIPoolExecutor.submit()* method.

- The submitted callable must use *get_comm_workers()* to access an intracommunicator grouping MPI worker processes. Afterwards, it is highly recommended calling the *Barrier()* method on the communicator. The barrier synchronization ensures that every worker process is executing the submitted callable exactly once. Afterwards, the parallel task can safely perform any kind of point-to-point or collective operation using the returned communicator.

- The *Future* instances returned by *MPIPoolExecutor.submit()* should be collected in a sequence. Use *wait()* with the sequence of *Future* instances to ensure logical completion of the parallel task.

### 7.5 Examples

#### Computing the Julia set

The following *julia.py* script computes the Julia set and dumps an image to disk in binary PGM format. The code starts by importing *MPIPoolExecutor* from the *mpi4py.futures* package. Next, some global constants and functions implement the computation of the Julia set. The computations are protected with the standard if __name__ == '__main__': ... idiom. The image is computed by whole scanlines submitting all these tasks at once using the *map* method. The result iterator yields scanlines in-order as the tasks complete. Finally, each scanline is dumped to disk.

```python
from mpi4py.futures import MPIPoolExecutor
```

(continues on next page)
The recommended way to execute the script is by using the `mpiexec` command specifying one MPI process (master) and (optional but recommended) the desired MPI universe size, which determines the number of additional dynamically spawned processes (workers). The MPI universe size is provided either by a batch system or set by the user via command-line arguments to `mpiexec` or environment variables. Below we provide examples for MPICH and Open MPI implementations. In all of these examples, the `mpiexec` command launches a single master process running the Python interpreter and executing the main script. When required, `mpi4py.futures` spawns the pool of 16 worker processes. The master submits tasks to the workers and waits for the results. The workers receive incoming tasks, execute them, and send back the results to the master.

When using MPICH implementation or its derivatives based on the Hydra process manager, users can set the MPI universe size via the `-usize` argument to `mpiexec`:

```
$ mpiexec -n 1 -usize 17 python julia.py
```

or, alternatively, by setting the `MPIEXEC_UNIVERSE_SIZE` environment variable:

```
$ env MPIEXEC_UNIVERSE_SIZE=17 mpiexec -n 1 python julia.py
```

In the Open MPI implementation, the MPI universe size can be set via the `-host` argument to `mpiexec`:

```
$ mpiexec -n 1 -host 17 python julia.py
```

1 When using an MPI implementation other than MPICH or Open MPI, please check the documentation of the implementation and/or batch system for the ways to specify the desired MPI universe size.
Another way to specify the number of workers is to use the `mpi4py.futures`-specific environment variable `MPI4PY_FUTURES_MAX_WORKERS`:

```
$ env MPI4PY_FUTURES_MAX_WORKERS=16 mpiexec -n 1 python julia.py
```

Note that in this case, the MPI universe size is ignored.

Alternatively, users may decide to execute the script in a more traditional way, that is, all the MPI processes are started at once. The user script is run under command-line control of `mpi4py.futures` passing the `-m` flag to the `python` executable:

```
$ mpiexec -n 17 python -m mpi4py.futures julia.py
```

As explained previously, the 17 processes are partitioned in one master and 16 workers. The master process executes the main script while the workers execute the tasks submitted by the master.

### Computing Pi (parallel task)

The number $\pi$ can be approximated via numerical integration with the simple midpoint rule, that is:

$$
\pi = \int_0^1 \frac{4}{1 + x^2} \, dx \approx \frac{1}{n} \sum_{i=1}^{n} \frac{4}{1 + \left[ \frac{1}{n} (i - \frac{1}{2}) \right]^2}.
$$

The following `cpi.py` script computes such approximations using `mpi4py.futures` with a parallel task involving a collective reduction operation. Highlighted lines correspond to the rules discussed in *Parallel tasks*.

Listing 2: cpi.py

```python
import math
import sys
from mpi4py.futures import MPIPoolExecutor, wait
from mpi4py.futures import get_comm_workers

def compute_pi(n):
    # Access intracommunicator and synchronize
    comm = get_comm_workers()
    comm.Barrier()

    rank = comm.Get_rank()
    size = comm.Get_size()

    # Local computation
    h = 1.0 / n
    s = 0.0
    for i in range(rank + 1, n + 1, size):
        x = h * (i - 0.5)
        s += 4.0 / (1.0 + x**2)
    pi_partial = s * h

    # Parallel reduce-to-all

```
pi = comm.allreduce(pi_partial)

# All workers return the same value
return pi

if __name__ == '__main__':
    n = int(sys.argv[1]) if len(sys.argv) > 1 else 256

    with MPIPoolExecutor() as executor:
        # Submit exactly one callable per worker
        P = executor.num_workers
        fs = [executor.submit(compute_pi, n) for _ in range(P)]

        # Wait for all workers to finish
        wait(fs)

        # Get result from the first future object.
        # In this particular example, due to using reduce-to-all,
        # all the other future objects hold the same result value.
        pi = fs[0].result()
        print(f'pi: {pi:.16f}, error: {abs(pi - math.pi):.3e}',
              f'({n:d} intervals, {P:d} workers)')
7.6 Citation

If `mpi4py.futures` been significant to a project that leads to an academic publication, please acknowledge our work by citing the following article [mpi4py-futures]:

8 mpi4py.util

New in version 3.1.0.

The `mpi4py.util` package collects miscellaneous utilities within the intersection of Python and MPI.

8.1 mpi4py.util.dtlib

New in version 3.1.0.

The `mpi4py.util.dtlib` module provides converter routines between NumPy and MPI datatypes.

`mpi4py.util.dtlib.from_numpy_dtype(dtype)`

Convert NumPy datatype to MPI datatype.

**Parameters**

dtype (*DTypeLike*) – NumPy dtype-like object.

**Return type**

Datatype

`mpi4py.util.dtlib.to_numpy_dtype(datatype)`

Convert MPI datatype to NumPy datatype.

**Parameters**

datatype (*Datatype*) – MPI datatype.

**Return type**

dtype[/*Any*]

8.2 mpi4py.util.pkl5

New in version 3.1.0.

`pickle` protocol 5 (see PEP 574) introduced support for out-of-band buffers, allowing for more efficient handling of certain object types with large memory footprints.

MPI for Python uses the traditional in-band handling of buffers. This approach is appropriate for communicating non-buffer Python objects, or buffer-like objects with small memory footprints. For point-to-point communication, in-band buffer handling allows for the communication of a pickled stream with a single MPI message, at the expense of additional CPU and memory overhead in the pickling and unpickling steps.

The `mpi4py.util.pkl5` module provides communicator wrapper classes reimplementing pickle-based point-to-point communication methods using pickle protocol 5. Handling out-of-band buffers necessarily involve multiple MPI messages, thus increasing latency and hurting performance in case of small size data. However, in case of large size data, the zero-copy savings of out-of-band buffer handling more than offset the extra latency costs. Additionally, these wrapper methods overcome the infamous 2 GiB message count limit (MPI-1 to MPI-3).

**Note:** Support for pickle protocol 5 is available in the `pickle` module within the Python standard library since Python 3.8. Previous Python 3 releases can use the `pickle5` backport, which is available on PyPI and can be installed with:
class mpi4py.util.pkl5.Request
    Request.
    Custom request class for nonblocking communications.

    **Note:** Request is not a subclass of mpi4py.MPI.Request

    Free()
    Free a communication request.

    **Return type**
    None

cancel()
    Cancel a communication request.

    **Return type**
    None

get_status(status=None)
    Non-destructive test for the completion of a request.

    **Parameters**
    status (Status | None) –

    **Return type**
    bool

test(status=None)
    Test for the completion of a request.

    **Parameters**
    status (Status | None) –

    **Return type**
    tuple[bool, Any | None]

wait(status=None)
    Wait for a request to complete.

    **Parameters**
    status (Status | None) –

    **Return type**
    Any

classmethod testall(requests, statuses=None)
    Test for the completion of all requests.

Classmethod

classmethod waitall(requests, statuses=None)
    Wait for all requests to complete.
class mpi4py.util.pkl5.Message
Message.
Custom message class for matching probes.

Note: Message is not a subclass of mpi4py.MPI.Message

recv(status=None)
Blocking receive of matched message.

Parameters
status (Status | None) –

Return type
Any

irecv()
Nonblocking receive of matched message.

Return type
Request

classmethod probe(comm, source=ANY_SOURCE, tag=ANY_TAG, status=None)
Blocking test for a matched message.

Classmethod
classmethod iprobe(comm, source=ANY_SOURCE, tag=ANY_TAG, status=None)
Nonblocking test for a matched message.

Classmethod

class mpi4py.util.pkl5.Comm
Communicator.
Base communicator wrapper class.

send(obj, dest, tag=0)
Blocking send in standard mode.

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –

Return type
None
bsend(obj, dest, tag=0)
Blocking send in buffered mode.

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –
Return type
None

ssend(obj, dest, tag=0)
Blocking send in synchronous mode.

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –

Return type
None

isend(obj, dest, tag=0)
Nonblocking send in standard mode.

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –

Return type
Request

ibsend(obj, dest, tag=0)
Nonblocking send in buffered mode.

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –

Return type
Request

issend(obj, dest, tag=0)
Nonblocking send in synchronous mode.

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –

Return type
Request

recv(buf=None, source=ANY_SOURCE, tag=ANY_TAG, status=None)
Blocking receive.

Parameters
• buf (Buffer | None) –
• source (int) –

return

• **tag** (int) –
• **status** (Status | None) –

**Return type**
Any

**irecv(buf=None, source=ANY_SOURCE, tag=ANY_TAG)**
Nonblocking receive.

**Warning:** This method cannot be supported reliably and raises **RuntimeError**.

**Parameters**
• **buf** (Buffer | None) –
• **source** (int) –
• **tag** (int) –

**Return type**
Request

**sendrecv(sendobj, dest, sendtag=0, recvbuf=None, source=ANY_SOURCE, recvtag=ANY_TAG, status=NONE)**
Send and receive.

**Parameters**
• **sendobj** (Any) –
• **dest** (int) –
• **sendtag** (int) –
• **recvbuf** (Buffer | None) –
• **source** (int) –
• **recvtag** (int) –
• **status** (Status | None) –

**Return type**
Any

**mprobe(source=ANY_SOURCE, tag=ANY_TAG, status=None)**
Blocking test for a matched message.

**Parameters**
• **source** (int) –
• **tag** (int) –
• **status** (Status | None) –

**Return type**
Message
**improve** *(source=ANY_SOURCE, tag=ANY_TAG, status=None)*

Nonblocking test for a matched message.

**Parameters**

- **source** *(int)* –
- **tag** *(int)* –
- **status** *(Status | None)* –

**Return type**

- Message | None

**bcast** *(obj, root=0)*

Broadcast.

New in version 3.1.0.

**Parameters**

- **obj** *(Any)* –
- **root** *(int)* –

**Return type**

- Any

**gather** *(sendobj, root=0)*

Gather.

New in version 4.0.0.

**Parameters**

- **sendobj** *(Any)* –
- **root** *(int)* –

**Return type**

- list[Any] | None

**scatter** *(sendobj, root=0)*

Scatter.

New in version 4.0.0.

**Parameters**

- **sendobj** *(Sequence[Any] | None)* –
- **root** *(int)* –

**Return type**

- Any

**allgather** *(sendobj)*

Gather to All.

New in version 4.0.0.

**Parameters**

- **sendobj** *(Any)* –

**Return type**

- list[Any]
alltoall(sendobj)
    All to All Scatter/Gather.
    New in version 4.0.0.

Parameters
    sendobj (Sequence[Any]) –

Return type
    list[Any]

class mpi4py.util.pkl5.Intracomm
    Intracommunicator.
    Intracommunicator wrapper class.

class mpi4py.util.pkl5.Intercomm
    Intercommunicator.
    Intercommunicator wrapper class.

Examples

Listing 3: test-pkl5-1.py

```python
import numpy as np
from mpi4py import MPI
from mpi4py.util import pkl5

comm = pkl5.Intracomm(MPI.COMM_WORLD)  # comm wrapper
size = comm.Get_size()
rank = comm.Get_rank()
dst = (rank + 1) % size
src = (rank - 1) % size

sobj = np.full(1024**3, rank, dtype='i4')  # > 4 GiB
sreq = comm.isend(sobj, dst, tag=42)
robj = comm.recv(None, src, tag=42)
sreq.Free()

assert np.min(robj) == src
assert np.max(robj) == src
```

Listing 4: test-pkl5-2.py

```python
import numpy as np
from mpi4py import MPI
from mpi4py.util import pkl5

comm = pkl5.Intracomm(MPI.COMM_WORLD)  # comm wrapper
size = comm.Get_size()
rank = comm.Get_rank()
dst = (rank + 1) % size
src = (rank - 1) % size
```
sobj = np.full(1024**3, rank, dtype='i4')  # > 4 GiB
sreq = comm.isend(sobj, dst, tag=42)

status = MPI.Status()
rmmsg = comm.mprobe(status=status)
assert status.Get_source() == src
assert status.Get_tag() == 42
rreq = rmmsg.irecv()
robj = rreq.wait()

sreq.Free()
assert np.max(robj) == src
assert np.min(robj) == src

8.3 mpi4py.util.pool

New in version 4.0.0.

See also:
This module intends to be a drop-in replacement for the multiprocessing.pool interface from the Python standard library. The Pool class exposed here is implemented as a thin wrapper around MPIPoolExecutor.

Note: The mpi4py.futures package offers a higher level interface for asynchronously pushing tasks to MPI worker process, allowing for a clear separation between submitting tasks and waiting for the results.

class mpi4py.util.pool.Pool

Pool using MPI processes as workers.

__init__(processes=None, initializer=None, initargs=(), **kwargs)

Initialize a new Pool instance.

Parameters

• processes – Number of worker processes.

• initializer – An callable used to initialize workers processes.

• initargs – A tuple of arguments to pass to the initializer.

Note: Additional keyword arguments are passed down to the MPIPoolExecutor constructor.

Warning: The maxtasksperchild and context arguments of multiprocessing.pool.Pool are not supported. Specifying maxtasksperchild or context with a value other than None will issue a warning of category UserWarning.

apply(func, args=(), kwds={})

Call func with arguments args and keyword arguments kwds.
Equivalent to func(*args, **kwds).
apply_async(func, args=(), kwds={}, callback=None, error_callback=None)
Asynchronous version of apply() returning ApplyResult.

map(func, iterable, chunksize=None)
Apply func to each element in iterable.
Equivalent to list(map(func, iterable)).
Block until all results are ready and return them in a list.
The iterable is choped into a number of chunks which are submitted as separate tasks. The (approximate) size of these chunks can be specified by setting chunksize to a positive integer.
Consider using imap() or imap_unordered() with explicit chunksize for better efficiency.

map_async(func, iterable, chunksize=None, callback=None, error_callback=None)
Asynchronous version of map() returning MapResult.

imap(func, iterable, chunksize=1)
Like map() but return an iterator.
Equivalent to map(func, iterable).

imap_unordered(func, iterable, chunksize=1)
Like imap() but ordering of results is arbitrary.

starmap(func, iterable, chunksize=None)
Apply func to each argument tuple in iterable.
Equivalent to list(itertools.starmap(func, iterable)).
Block until all results are ready and return them in a list.
The iterable is choped into a number of chunks which are submitted as separate tasks. The (approximate) size of these chunks can be specified by setting chunksize to a positive integer.
Consider using istarmap() or istarmap_unordered() with explicit chunksize for better efficiency.

starmap_async(func, iterable, chunksize=None, callback=None, error_callback=None)
Asynchronous version of starmap() returning MapResult.

istarmap(func, iterable, chunksize=1)
Like starmap() but return an iterator.
Equivalent to itertools.starmap(func, iterable).

istarmap_unordered(func, iterable, chunksize=1)
Like istarmap() but ordering of results is arbitrary.

close()
Prevent any more tasks from being submitted to the pool.

terminate()
Stop the worker processes without completing pending tasks.

join()
Wait for the worker processes to exit.

class mpi4py.util.pool.ThreadPool
Bases: Pool
Pool using threads as workers.
class mpi4py.util.poolAsyncResult

Asynchronous result.

get(timeout=None)

Return the result when it arrives.

If timeout is not None and the result does not arrive within timeout seconds then raise TimeoutError.

If the remote call raised an exception then that exception will be reraised.

wait(timeout=None)

Wait until the result is available or timeout seconds pass.

ready()

Return whether the call has completed.

successful()

Return whether the call completed without raising an exception.

If the result is not ready then raise ValueError.

class mpi4py.util.pool.ApplyResult

Bases:AsyncResult

Result type of apply_async().

class mpi4py.util.pool.MapResult

Bases:AsyncResult

Result type of map_async() and starmap_async().

9 mpi4py.run

New in version 3.0.0.

At import time, mpi4py initializes the MPI execution environment calling MPI_Init_thread() and installs an exit hook to automatically call MPI_Finalize() just before the Python process terminates. Additionally, mpi4py overrides the default ERRORS_ARE_FATAL error handler in favor of ERRORS_RETURN, which allows translating MPI errors in Python exceptions. These departures from standard MPI behavior may be controversial, but are quite convenient within the highly dynamic Python programming environment. Third-party code using mpi4py can just from mpi4py import MPI and perform MPI calls without the tedious initialization/finalization handling. MPI errors, once translated automatically to Python exceptions, can be dealt with the common try...except...finally clauses; unhandled MPI exceptions will print a traceback which helps in locating problems in source code.

Unfortunately, the interplay of automatic MPI finalization and unhandled exceptions may lead to deadlocks. In unattended runs, these deadlocks will drain the battery of your laptop, or burn precious allocation hours in your supercomputing facility.
9.1 Exceptions and deadlocks

Consider the following snippet of Python code. Assume this code is stored in a standard Python script file and run with mpiexec in two or more processes.

```
from mpi4py import MPI
assert MPI.COMM_WORLD.Get_size() > 1
rank = MPI.COMM_WORLD.Get_rank()
if rank == 0:
    1/0
MPI.COMM_WORLD.send(None, dest=1, tag=42)
elif rank == 1:
    MPI.COMM_WORLD.recv(source=0, tag=42)
```

Process 0 raises ZeroDivisionError exception before performing a send call to process 1. As the exception is not handled, the Python interpreter running in process 0 will proceed to exit with non-zero status. However, as mpi4py installed a finalizer hook to call MPI_Finalize() before exit, process 0 will block waiting for other processes to also enter the MPI_Finalize() call. Meanwhile, process 1 will block waiting for a message to arrive from process 0, thus never reaching to MPI_Finalize(). The whole MPI execution environment is irremediably in a deadlock state.

To alleviate this issue, mpi4py offers a simple, alternative command line execution mechanism based on using the -m flag and implemented with the runpy module. To use this features, Python code should be run passing -m mpi4py in the command line invoking the Python interpreter. In case of unhandled exceptions, the finalizer hook will call MPI_Abort() on the MPI_COMM_WORLD communicator, thus effectively aborting the MPI execution environment.

### Warning:
When a process is forced to abort, resources (e.g. open files) are not cleaned-up and any registered finalizers (either with the atexit module, the Python C/API function Py_AtExit(), or even the C standard library function atexit()) will not be executed. Thus, aborting execution is an extremely impolite way of ensuring process termination. However, MPI provides no other mechanism to recover from a deadlock state.

9.2 Command line

The use of -m mpi4py to execute Python code on the command line resembles that of the Python interpreter.

- `mpiexec -n numprocs python -m mpi4py pyfile [arg] ...`
- `mpiexec -n numprocs python -m mpi4py -m mod [arg] ...`
- `mpiexec -n numprocs python -m mpi4py -c cmd [arg] ...`
- `mpiexec -n numprocs python -m mpi4py - [arg] ...

<pyfile>
Execute the Python code contained in `pyfile`, which must be a filesystem path referring to either a Python file, a directory containing a __main__.py file, or a zipfile containing a __main__.py file.

-m <mod>
Search `sys.path` for the named module `mod` and execute its contents.

-c <cmd>
Execute the Python code in the `cmd` string command.
Read commands from standard input (`sys.stdin`).

See also:
Command line
Documentation on Python command line interface.

10 mpi4py.bench

New in version 3.0.0.

11 Reference

11.1 mpi4py.MPI

Message Passing Interface.

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomType</td>
<td>Type of <code>BOTTOM</code></td>
</tr>
<tr>
<td>Cartcomm</td>
<td>Cartesian topology intracommunicator</td>
</tr>
<tr>
<td>Comm</td>
<td>Communicator</td>
</tr>
<tr>
<td>Datatype</td>
<td>Datatype object</td>
</tr>
<tr>
<td>Distgraphcomm</td>
<td>Distributed graph topology intracommunicator</td>
</tr>
<tr>
<td>Errhandler</td>
<td>Error handler</td>
</tr>
<tr>
<td>File</td>
<td>File handle</td>
</tr>
<tr>
<td>Graphcomm</td>
<td>General graph topology intracommunicator</td>
</tr>
<tr>
<td>Grequest</td>
<td>Generalized request handle</td>
</tr>
<tr>
<td>Group</td>
<td>Group of processes</td>
</tr>
<tr>
<td>InPlaceType</td>
<td>Type of <code>IN_PLACE</code></td>
</tr>
<tr>
<td>Info</td>
<td>Info object</td>
</tr>
<tr>
<td>Intercomm</td>
<td>Intercommunicator</td>
</tr>
<tr>
<td>Intracomm</td>
<td>Intracommitor</td>
</tr>
<tr>
<td>Message</td>
<td>Matched message handle</td>
</tr>
<tr>
<td>Op</td>
<td>Operation object</td>
</tr>
<tr>
<td>Pickle</td>
<td>Pickle/unpickle Python objects</td>
</tr>
<tr>
<td>Prequest</td>
<td>Persistent request handle</td>
</tr>
<tr>
<td>Request</td>
<td>Request handle</td>
</tr>
<tr>
<td>Session</td>
<td>Session</td>
</tr>
<tr>
<td>Status</td>
<td>Status object</td>
</tr>
<tr>
<td>Topocomm</td>
<td>Topology intracommunicator</td>
</tr>
<tr>
<td>Win</td>
<td>Window handle</td>
</tr>
<tr>
<td>memory</td>
<td>Memory buffer</td>
</tr>
</tbody>
</table>
**mpi4py.MPI.BottomType**

**class mpi4py.MPI.BottomType**

Bases: int

Type of BOTTOM

**static __new__(cls)**

Return type BottomType

**mpi4py.MPI.Cartcomm**

**class mpi4py.MPI.Cartcomm**

Bases: Topocomm

Cartesian topology intracommunicator

**static __new__(cls, comm=None)**

Parameters

comm (Cartcomm | None) –

Return type Cartcomm

**Methods Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_cart_rank</td>
<td>Translate logical coordinates to ranks</td>
</tr>
<tr>
<td>Get_coords</td>
<td>Translate ranks to logical coordinates</td>
</tr>
<tr>
<td>Get_dim</td>
<td>Return number of dimensions</td>
</tr>
<tr>
<td>Get_topo</td>
<td>Return information on the cartesian topology</td>
</tr>
<tr>
<td>Shift</td>
<td>Return a tuple (source, dest) of process ranks for data</td>
</tr>
<tr>
<td></td>
<td>shifting with Comm.Sendrecv()</td>
</tr>
<tr>
<td>Sub</td>
<td>Return cartesian communicators that form lower-dimensional</td>
</tr>
<tr>
<td></td>
<td>subgrids</td>
</tr>
</tbody>
</table>

**Attributes Summary**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coords</td>
<td>coordinates</td>
</tr>
<tr>
<td>dim</td>
<td>number of dimensions</td>
</tr>
<tr>
<td>dims</td>
<td>dimensions</td>
</tr>
<tr>
<td>ndim</td>
<td>number of dimensions</td>
</tr>
<tr>
<td>periods</td>
<td>periodicity</td>
</tr>
<tr>
<td>topo</td>
<td>topology information</td>
</tr>
</tbody>
</table>
Methods Documentation

**Get_cart_rank** *(coords)*

Translate logical coordinates to ranks

- **Parameters**
  - coords *(Sequence[int]*)

- **Return type**
  - int

**Get_coords** *(rank)*

Translate ranks to logical coordinates

- **Parameters**
  - rank *(int)*

- **Return type**
  - list[int]

**Get_dim** *

Return number of dimensions

- **Return type**
  - int

**Get_topo** *

Return information on the cartesian topology

- **Return type**
  - tuple[list[int], list[int], list[int]]

**Shift** *(direction, disp)*

Return a tuple (source, dest) of process ranks for data shifting with Comm.Sendrecv()

- **Parameters**
  - direction *(int)*
  - disp *(int)*

- **Return type**
  - tuple[int, int]

**Sub** *(remain_dims)*

Return cartesian communicators that form lower-dimensional subgrids

- **Parameters**
  - remain_dims *(Sequence[bool]*)

- **Return type**
  - Cartcomm
Attributes Documentation

cords
coordinates
dim
number of dimensions
dims
dimensions
ndim
number of dimensions
periods
periodicity
topo
topology information

mpi4py.MPI.Comm

class mpi4py.MPI.Comm
    Bases: object
    Communicator
    static __new__(cls, comm=None)
        Parameters
        comm (Comm | None) –
        Return type
        Comm

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort(errorcode)</td>
<td>Terminate MPI execution environment</td>
</tr>
<tr>
<td>Ack_failed(num_to_ack)</td>
<td>Acknowledge failures on a communicator</td>
</tr>
<tr>
<td>Agree(flag)</td>
<td>Blocking agreement</td>
</tr>
<tr>
<td>Allgather(sendbuf, recvbuf)</td>
<td>Gather to All, gather data from all processes and distribute it to all other processes in a group</td>
</tr>
<tr>
<td>Allgatherv(sendbuf, recvbuf)</td>
<td>Gather to All Vector, gather data from all processes and distribute it to all other processes in a group</td>
</tr>
<tr>
<td>Allgatherv_init(sendbuf, recvbuf[, info])</td>
<td>Persistent Gather to All Vector</td>
</tr>
<tr>
<td>Allzgatherv_init(sendbuf, recvbuf[, info])</td>
<td>Persistent Gather to All Vector</td>
</tr>
<tr>
<td>Allreduce(sendbuf, recvbuf[, op])</td>
<td>Reduce to All</td>
</tr>
<tr>
<td>Allreduce_init(sendbuf, recvbuf[, op, info])</td>
<td>Persistent All Reduce</td>
</tr>
<tr>
<td>Alltoall(sendbuf, recvbuf)</td>
<td>All to All Scatter/Gather, send data from all to all processes in a group</td>
</tr>
<tr>
<td>Alltoall_init(sendbuf, recvbuf[, info])</td>
<td>Persistent All to All Scatter/Gather</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alltoallv</td>
<td>All to All Scatter/Gather Vector, send data from all to all processes in a group providing different amount of data and displacements</td>
</tr>
<tr>
<td>Alltoallv_init</td>
<td>Persistent All to All Scatter/Gather Vector</td>
</tr>
<tr>
<td>Alltoallw</td>
<td>Generalized All-to-All communication allowing different counts, displacements and datatypes for each partner</td>
</tr>
<tr>
<td>Alltoallw_init</td>
<td>Persistent Generalized All-to-All</td>
</tr>
<tr>
<td>Barrier</td>
<td>Barrier synchronization</td>
</tr>
<tr>
<td>Barrier_init</td>
<td>Persistent Barrier</td>
</tr>
<tr>
<td>Bcast</td>
<td>Broadcast a message from one process to all other processes in a group</td>
</tr>
<tr>
<td>Bcast_init</td>
<td>Persistent Broadcast</td>
</tr>
<tr>
<td>Bsend</td>
<td>Blocking send in buffered mode</td>
</tr>
<tr>
<td>Bsend_init</td>
<td>Persistent request for a send in buffered mode</td>
</tr>
<tr>
<td>Call_errhandler</td>
<td>Call the error handler installed on a communicator</td>
</tr>
<tr>
<td>Clone</td>
<td>Clone an existing communicator</td>
</tr>
<tr>
<td>Compare</td>
<td>Compare two communicators</td>
</tr>
<tr>
<td>Create</td>
<td>Create communicator from group</td>
</tr>
<tr>
<td>Create_errhandler</td>
<td>Create a new error handler for communicators</td>
</tr>
<tr>
<td>Create_keyval</td>
<td>Create a new attribute key for communicators</td>
</tr>
<tr>
<td>Delete_attr</td>
<td>Delete attribute value associated with a key</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Disconnect from a communicator</td>
</tr>
<tr>
<td>Dup</td>
<td>Duplicate an existing communicator</td>
</tr>
<tr>
<td>Dup_with_info</td>
<td>Duplicate an existing communicator</td>
</tr>
<tr>
<td>Free</td>
<td>Free a communicator</td>
</tr>
<tr>
<td>Free_keyval</td>
<td>Free an attribute key for communicators</td>
</tr>
<tr>
<td>Gather</td>
<td>Gather together values from a group of processes</td>
</tr>
<tr>
<td>Gather_init</td>
<td>Persistent Gather</td>
</tr>
<tr>
<td>Gatherv</td>
<td>Gather Vector, gather data to one process from all other processes in a group providing different amount of data and displacements at the receiving sides</td>
</tr>
<tr>
<td>Gatherv_init</td>
<td>Persistent Gather Vector</td>
</tr>
<tr>
<td>Get_attr</td>
<td>Retrieve attribute value by key</td>
</tr>
<tr>
<td>Get_errhandler</td>
<td>Get the error handler for a communicator</td>
</tr>
<tr>
<td>Get_failed</td>
<td>Extract the group of failed processes</td>
</tr>
<tr>
<td>Get_group</td>
<td>Access the group associated with a communicator</td>
</tr>
<tr>
<td>Get_info</td>
<td>Return the hints for a communicator that are currently in use</td>
</tr>
<tr>
<td>Get_name</td>
<td>Get the print name for this communicator</td>
</tr>
<tr>
<td>Get_parent</td>
<td>Return the parent intercommunicator for this process</td>
</tr>
<tr>
<td>Get_rank</td>
<td>Return the rank of this process in a communicator</td>
</tr>
<tr>
<td>Get_size</td>
<td>Return the number of processes in a communicator</td>
</tr>
<tr>
<td>Get_topology</td>
<td>Determine the type of topology (if any) associated with a communicator</td>
</tr>
<tr>
<td>Iagree</td>
<td>Non blocking agreement</td>
</tr>
<tr>
<td>Iallgather</td>
<td>Nonblocking Gather to All</td>
</tr>
<tr>
<td>Iallgatherv</td>
<td>Nonblocking Gather to All Vector</td>
</tr>
<tr>
<td>Iallreduce</td>
<td>Nonblocking Reduce to All</td>
</tr>
<tr>
<td>Ialltoall</td>
<td>Nonblocking All to All Scatter/Gather</td>
</tr>
<tr>
<td>Ialltoallv</td>
<td>Nonblocking All to All Scatter/Gather Vector</td>
</tr>
<tr>
<td>Ialltoallw</td>
<td>Nonblocking Generalized All-to-All</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Ibarrier()</code></td>
<td>Nonblocking Barrier</td>
</tr>
<tr>
<td><code>Ibcast(buf, root)</code></td>
<td>Nonblocking Broadcast</td>
</tr>
<tr>
<td><code>Ibsend(buf, dest[, tag])</code></td>
<td>Nonblocking send in buffered mode</td>
</tr>
<tr>
<td><code>Idup([info])</code></td>
<td>Nonblocking duplicate an existing communicator</td>
</tr>
<tr>
<td><code>Idup_with_info(info)</code></td>
<td>Duplicate an existing communicator</td>
</tr>
<tr>
<td><code>Igather(sendbuf, recvbuf[, root])</code></td>
<td>Nonblocking Gather</td>
</tr>
<tr>
<td><code>Igatherv(sendbuf, recvbuf[, root])</code></td>
<td>Nonblocking Gather Vector</td>
</tr>
<tr>
<td><code>Improve([source, tag, status])</code></td>
<td>Nonblocking test for a matched message</td>
</tr>
<tr>
<td><code>Iprobe([source, tag, status])</code></td>
<td>Nonblocking test for a message</td>
</tr>
<tr>
<td><code>Irecv(buf[, source, tag])</code></td>
<td>Nonblocking receive</td>
</tr>
<tr>
<td><code>Ireduce(sendbuf, recvbuf[, op, root])</code></td>
<td>Nonblocking Reduce to Root</td>
</tr>
<tr>
<td><code>Ireduce_scatter(sendbuf, recvbuf[, ...])</code></td>
<td>Nonblocking Reduce-Scatter (vector version)</td>
</tr>
<tr>
<td><code>Ireduce_scatter_block(sendbuf, recvbuf[, op])</code></td>
<td>Nonblocking Reduce-Scatter Block (regular, non-vector version)</td>
</tr>
<tr>
<td><code>Irsend(buf, dest[, tag])</code></td>
<td>Nonblocking send in ready mode</td>
</tr>
<tr>
<td><code>Is_inter()</code></td>
<td>Test to see if a comm is an intercommunicator</td>
</tr>
<tr>
<td><code>Is_intra()</code></td>
<td>Test to see if a comm is an intracommunicator</td>
</tr>
<tr>
<td><code>Is_revoked()</code></td>
<td>Indicate whether the communicator has been revoked</td>
</tr>
<tr>
<td><code>Iscatter(sendbuf, recvbuf[, root])</code></td>
<td>Nonblocking Scatter</td>
</tr>
<tr>
<td><code>Iscatterv(sendbuf, recvbuf[, root])</code></td>
<td>Nonblocking Scatter Vector</td>
</tr>
<tr>
<td><code>Isend(buf[, dest, tag])</code></td>
<td>Nonblocking send</td>
</tr>
<tr>
<td><code>Irecv_init(buf[, source, tag])</code></td>
<td>Create a persistent request for a receive</td>
</tr>
<tr>
<td><code>Isendrecv(sendbuf, dest[, sendtag, recvbuf, ...])</code></td>
<td>Send and receive a message</td>
</tr>
<tr>
<td><code>Isendrecv_replace(buf, dest[, sendtag, ...])</code></td>
<td>Send and receive a message</td>
</tr>
<tr>
<td><code>Ishrink()</code></td>
<td>Nonblocking shrink a communicator to remove all failed processes</td>
</tr>
<tr>
<td><code>Issend(buf, dest[, tag])</code></td>
<td>Nonblocking send in synchronous mode</td>
</tr>
<tr>
<td><code>Join(fd)</code></td>
<td>Create an intercommunicator by joining two processes connected by a socket</td>
</tr>
<tr>
<td><code>Mprobe([source, tag, status])</code></td>
<td>Blocking test for a matched message</td>
</tr>
<tr>
<td><code>Precv_init(buf partitions[, source, tag, info])</code></td>
<td>Create request for a partitioned recv operation</td>
</tr>
<tr>
<td><code>Probe([source, tag, status])</code></td>
<td>Blocking test for a message</td>
</tr>
<tr>
<td><code>Psend_init(buf partitions, dest[, tag, info])</code></td>
<td>Create request for a partitioned send operation</td>
</tr>
<tr>
<td><code>Recv(buf[, source, tag, status])</code></td>
<td>Blocking receive</td>
</tr>
<tr>
<td><code>Recv_init(buf[, source, tag])</code></td>
<td>Create a persistent request for a receive</td>
</tr>
<tr>
<td><code>Reduce(sendbuf, recvbuf[, op, root])</code></td>
<td>Reduce to Root</td>
</tr>
<tr>
<td><code>Reduce_init(sendbuf, recvbuf[, op, root, info])</code></td>
<td>Persistent Reduce</td>
</tr>
<tr>
<td><code>Reduce_scatter(sendbuf, recvbuf[, ...])</code></td>
<td>Reduce-Scatter (vector version)</td>
</tr>
<tr>
<td><code>Reduce_scatter_block(sendbuf, recvbuf[, op])</code></td>
<td>Reduce-Scatter Block (regular, non-vector version)</td>
</tr>
<tr>
<td><code>Reduce_scatter_block_init(sendbuf, recvbuf)</code></td>
<td>Persistent Reduce-Scatter Block (regular, non-vector version)</td>
</tr>
<tr>
<td><code>Reduce_scatter_init(sendbuf, recvbuf[, ...])</code></td>
<td>Persistent Reduce-Scatter (vector version)</td>
</tr>
<tr>
<td><code>Revoke()</code></td>
<td>Revoke a communicator</td>
</tr>
<tr>
<td><code>Rsend(buf, dest[, tag])</code></td>
<td>Blocking send in ready mode</td>
</tr>
<tr>
<td><code>Rsend_init(buf, dest[, tag])</code></td>
<td>Persistent request for a send in ready mode</td>
</tr>
<tr>
<td><code>Scatter(sendbuf, recvbuf[, root])</code></td>
<td>Scatter data from one process to all other processes in a group</td>
</tr>
<tr>
<td><code>Scatter_init(sendbuf, recvbuf[, root, info])</code></td>
<td>Persistent Scatter</td>
</tr>
<tr>
<td><code>Scatterv(sendbuf, recvbuf[, root])</code></td>
<td>Scatter Vector, scatter data from one process to all other processes in a group providing different amount of data and displacements at the sending side</td>
</tr>
<tr>
<td><code>Scatterv_init(sendbuf, recvbuf[, root, info])</code></td>
<td>Persistent Scatter Vector</td>
</tr>
<tr>
<td><code>Send(buf, dest[, tag])</code></td>
<td>Blocking send</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Send_init(buf, dest[, tag])</td>
<td>Create a persistent request for a standard send</td>
</tr>
<tr>
<td>Sendrecv(sendbuf, dest[, sendtag, recvbuf, ...])</td>
<td>Send and receive a message</td>
</tr>
<tr>
<td>Sendrecv_replace(buf, dest[, sendtag, ...])</td>
<td>Send and receive a message</td>
</tr>
<tr>
<td>Set_attr(keyval, attrval)</td>
<td>Store attribute value associated with a key</td>
</tr>
<tr>
<td>Set_errhandler(errhandler)</td>
<td>Set the error handler for a communicator</td>
</tr>
<tr>
<td>Set_info(info)</td>
<td>Set new values for the hints associated with a communicator</td>
</tr>
<tr>
<td>Set_name(name)</td>
<td>Set the print name for this communicator</td>
</tr>
<tr>
<td>Shrink()</td>
<td>Shrink a communicator to remove all failed processes</td>
</tr>
<tr>
<td>Split([color, key])</td>
<td>Split communicator by color and key</td>
</tr>
<tr>
<td>Split_type(split_type[, key, info])</td>
<td>Split communicator by split type</td>
</tr>
<tr>
<td>Ssend(buf, dest[, tag])</td>
<td>Blocking send in synchronous mode</td>
</tr>
<tr>
<td>Ssend_init(buf, dest[, tag])</td>
<td>Persistent request for a send in synchronous mode</td>
</tr>
<tr>
<td>allgather(sendobj)</td>
<td>Gather to All</td>
</tr>
<tr>
<td>allreduce(sendobj[, op])</td>
<td>Reduce to All</td>
</tr>
<tr>
<td>alltoall(sendobj)</td>
<td>All to All Scatter/Gather</td>
</tr>
<tr>
<td>barrier()</td>
<td>Barrier synchronization</td>
</tr>
<tr>
<td>bcast(obj[, root])</td>
<td>Broadcast</td>
</tr>
<tr>
<td>bsend(obj, dest[, tag])</td>
<td>Send in buffered mode</td>
</tr>
<tr>
<td>f2py(arg)</td>
<td></td>
</tr>
<tr>
<td>gather(sendobj[, root])</td>
<td>Gather</td>
</tr>
<tr>
<td>ibsend(obj, dest[, tag])</td>
<td>Nonblocking send in buffered mode</td>
</tr>
<tr>
<td>improbe([source, tag, status])</td>
<td>Nonblocking test for a matched message</td>
</tr>
<tr>
<td>iprobe([source, tag, status])</td>
<td>Nonblocking test for a message</td>
</tr>
<tr>
<td>irecv(buf, source, tag)</td>
<td>Nonblocking receive</td>
</tr>
<tr>
<td>issend(obj, dest[, tag])</td>
<td>Nonblocking send in synchronous mode</td>
</tr>
<tr>
<td>mprobe([source, tag, status])</td>
<td>Blocking test for a matched message</td>
</tr>
<tr>
<td>probe([source, tag, status])</td>
<td>Blocking test for a message</td>
</tr>
<tr>
<td>py2f()</td>
<td></td>
</tr>
<tr>
<td>recv([buf, source, tag, status])</td>
<td>Receive</td>
</tr>
<tr>
<td>reduce(sendobj[, op, root])</td>
<td>Reduce to Root</td>
</tr>
<tr>
<td>scatter(sendobj[, root])</td>
<td>Scatter</td>
</tr>
<tr>
<td>send(obj, dest[, tag])</td>
<td>Send</td>
</tr>
<tr>
<td>sendrecv(sendobj, dest[, sendtag, recvbuf, ...])</td>
<td>Send and Receive</td>
</tr>
<tr>
<td>ssend(obj, dest[, tag])</td>
<td>Send in synchronous mode</td>
</tr>
</tbody>
</table>

### Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>communicator group</td>
</tr>
<tr>
<td>info</td>
<td>communicator info</td>
</tr>
<tr>
<td>is_inter</td>
<td>is intercommunicator</td>
</tr>
<tr>
<td>is_intra</td>
<td>is intracommunicator</td>
</tr>
<tr>
<td>is_topo</td>
<td>is a topology communicator</td>
</tr>
<tr>
<td>name</td>
<td>communicator name</td>
</tr>
<tr>
<td>rank</td>
<td>rank of this process in communicator</td>
</tr>
<tr>
<td>size</td>
<td>number of processes in communicator</td>
</tr>
<tr>
<td>topology</td>
<td>communicator topology type</td>
</tr>
</tbody>
</table>
Methods Documentation

Abort (errorcode=0)
 Terminate MPI execution environment

Warning:  This is a direct call, use it with care!!!.

Parameters
  errorcode (int) –

Return type
  NoReturn

Ack_failed (num_to_ack=None)
 Acknowledge failures on a communicator

Parameters
  num_to_ack (int / None) –

Return type
  int

Agree (flag)
 Blocking agreement

Parameters
  flag (int) –

Return type
  int

Allgather (sendbuf, recvbuf)
 Gather to All, gather data from all processes and distribute it to all other processes in a group

Parameters
  • sendbuf (BufSpec / InPlace) –
  • recvbuf (BufSpecB) –

Return type
  None

Allgather_init (sendbuf, recvbuf, info=INFO_NULL)
 Persistent Gather to All

Parameters
  • sendbuf (BufSpec / InPlace) –
  • recvbuf (BufSpecB) –
  • info (Info) –

Return type
  Prequest
**Allgatherv**\((sendbuf, recvbuf)\)

Gather to All Vector, gather data from all processes and distribute it to all other processes in a group providing different amount of data and displacements

**Parameters**

- **sendbuf** (BufSpec | InPlace) –
- **recvbuf** (BufSpecV) –

**Return type**

None

**Allgatherv_init**\((sendbuf, recvbuf, info=INFO_NULL)\)

Persistent Gather to All Vector

**Parameters**

- **sendbuf** (BufSpec | InPlace) –
- **recvbuf** (BufSpecV) –
- **info** (Info) –

**Return type**

Prequest

**Allreduce**\((sendbuf, recvbuf, op=SUM)\)

Reduce to All

**Parameters**

- **sendbuf** (BufSpec | InPlace) –
- **recvbuf** (BufSpec) –
- **op** (Op) –

**Return type**

None

**Allreduce_init**\((sendbuf, recvbuf, op=SUM, info=INFO_NULL)\)

Persistent All Reduce

**Parameters**

- **sendbuf** (BufSpec | InPlace) –
- **recvbuf** (BufSpec) –
- **op** (Op) –
- **info** (Info) –

**Return type**

Prequest

**Alltoall**\((sendbuf, recvbuf)\)

All to All Scatter/Gather, send data from all to all processes in a group

**Parameters**

- **sendbuf** (BufSpecB | InPlace) –
- **recvbuf** (BufSpecB) –
**Return type**
None

**Alltoall_init** *(sendbuf, recvbuf, info=INFO_NULL)*
Persistent All to All Scatter/Gather

**Parameters**
- *sendbuf* (BufSpecB | InPlace) –
- *recvbuf* (BufSpecB) –
- *info* (Info) –

**Return type**
Prequest

**Alltoallv** *(sendbuf, recvbuf)*
All to All Scatter/Gather Vector, send data from all to all processes in a group providing different amount of data and displacements

**Parameters**
- *sendbuf* (BufSpecV | InPlace) –
- *recvbuf* (BufSpecV) –

**Return type**
None

**Alltoallv_init** *(sendbuf, recvbuf, info=INFO_NULL)*
Persistent All to All Scatter/Gather Vector

**Parameters**
- *sendbuf* (BufSpecV | InPlace) –
- *recvbuf* (BufSpecV) –
- *info* (Info) –

**Return type**
Prequest

**Alltoallw** *(sendbuf, recvbuf)*
Generalized All-to-All communication allowing different counts, displacements and datatypes for each partner

**Parameters**
- *sendbuf* (BufSpecW | InPlace) –
- *recvbuf* (BufSpecW) –

**Return type**
None

**Alltoallw_init** *(sendbuf, recvbuf, info=INFO_NULL)*
Persistent Generalized All-to-All

**Parameters**
- *sendbuf* (BufSpecW | InPlace) –
- *recvbuf* (BufSpecW) –
• **info** *(Info)* –

**Return type**

*Prequest*

**Barrier()**

Barrier synchronization

**Return type**

*None*

**Barrier_init**(info=INFO_NULL)

Persistent Barrier

**Parameters**

• **info** *(Info)* –

**Return type**

*Prequest*

**Bcast**(buf, root=0)

Broadcast a message from one process to all other processes in a group

**Parameters**

• **buf** *(BufSpec)* –

• **root** *(int)* –

**Return type**

*None*

**Bcast_init**(buf, root=0, info=INFO_NULL)

Persistent Broadcast

**Parameters**

• **buf** *(BufSpec)* –

• **root** *(int)* –

• **info** *(Info)* –

**Return type**

*Prequest*

**Bsend**(buf, dest, tag=0)

Blocking send in buffered mode

**Parameters**

• **buf** *(BufSpec)* –

• **dest** *(int)* –

• **tag** *(int)* –

**Return type**

*None*

**Bsend_init**(buf, dest, tag=0)

Persistent request for a send in buffered mode

**Parameters**

• **buf** *(BufSpec)* –
- `dest (int)` –
- `tag (int)` –

Return type
Request

**Call Errhandler** (*errcode*)

Call the error handler installed on a communicator

Parameters

- `errcode (int)` –

Return type
None

**Clone**

Clone an existing communicator

Return type
Self

**Compare** (*comm*)

Compare two communicators

Parameters

- `comm (Comm)` –

Return type
int

**Create** (*group*)

Create communicator from group

Parameters

- `group (Group)` –

Return type
Comm

**classmethod Create Errhandler** (*errhandler_fn*)

Create a new error handler for communicators

Parameters

- `errhandler_fn (Callable[[Comm, int], None])` –

Return type
Errhandler

**classmethod Create Keyval** (*copy_fn=*, *delete_fn=*, *nopython=*)

Create a new attribute key for communicators

Parameters

- `copy_fn (Callable[[Comm, int, Any], Any] | None)` –
- `delete_fn (Callable[[Comm, int, Any], None] | None)` –
- `nopython (bool)` –

Return type
int
Delete_attr(keyval)
    Delete attribute value associated with a key

    Parameters
    keyval (int) –

    Return type
    None

Disconnect()
    Disconnect from a communicator

    Return type
    None

Dup(info=None)
    Duplicate an existing communicator

    Parameters
    info (Info | None) –

    Return type
    Self

Dup_with_info(info)
    Duplicate an existing communicator

    Parameters
    info (Info) –

    Return type
    Self

Free()
    Free a communicator

    Return type
    None

classmethod Free_keyval(keyval)
    Free an attribute key for communicators

    Parameters
    keyval (int) –

    Return type
    int

Gather(sendbuf, recvbuf, root=0)
    Gather together values from a group of processes

    Parameters
    • sendbuf (BufSpec | InPlace) –
    • recvbuf (BufSpecB | None) –
    • root (int) –

    Return type
    None
Gather_init(sendbuf, recvbuf, root=0, info=INFO_NULL)
Persistent Gather

Parameters

• sendbuf (BufSpec | InPlace) –
• recvbuf (BufSpecB | None) –
• root (int) –
• info (Info) –

Return type

Prequest

Gatherv(sendbuf, recvbuf, root=0)
Gather Vector, gather data to one process from all other processes in a group providing different amount of data and displacements at the receiving sides

Parameters

• sendbuf (BufSpec | InPlace) –
• recvbuf (BufSpecV | None) –
• root (int) –

Return type

None

Gatherv_init(sendbuf, recvbuf, root=0, info=INFO_NULL)
Persistent Gather Vector

Parameters

• sendbuf (BufSpec | InPlace) –
• recvbuf (BufSpecV | None) –
• root (int) –
• info (Info) –

Return type

Prequest

Get_attr(keyval)
Retrieve attribute value by key

Parameters

keyval (int) –

Return type

int | Any | None

Get_errhandler()
Get the error handler for a communicator

Return type

Errhandler
Get_failed()
   Extract the group of failed processes
   
   Return type
   Group

Get_group()
   Access the group associated with a communicator
   
   Return type
   Group

Get_info()
   Return the hints for a communicator that are currently in use
   
   Return type
   Info

Get_name()
   Get the print name for this communicator
   
   Return type
   str

classmethod Get_parent()
   Return the parent intercommunicator for this process
   
   Return type
   Intercomm

Get_rank()
   Return the rank of this process in a communicator
   
   Return type
   int

Get_size()
   Return the number of processes in a communicator
   
   Return type
   int

Get_topology()
   Determine the type of topology (if any) associated with a communicator
   
   Return type
   int

Iagree(flag)
   Non blocking agreement
   
   Parameters
   flag (Buffer) –
   
   Return type
   Request

Iallgather(sendbuf, recvbuf)
   Nonblocking Gather to All
   
   Parameters
• `sendbuf` ([BufSpec / InPlace]) –
• `recvbuf` ([BufSpec]) –

**Return type**
`Request`

**Iallgatherv**(`sendbuf`, `recvbuf`)
Nonblocking Gather to All Vector

**Parameters**
• `sendbuf` ([BufSpec / InPlace]) –
• `recvbuf` ([BufSpecV]) –

**Return type**
`Request`

**Iallreduce**(`sendbuf`, `recvbuf`, `op=SUM`)
Nonblocking Reduce to All

**Parameters**
• `sendbuf` ([BufSpec / InPlace]) –
• `recvbuf` ([BufSpec]) –
• `op` ([Op]) –

**Return type**
`Request`

**Ialltoall**(`sendbuf`, `recvbuf`)
Nonblocking All to All Scatter/Gather

**Parameters**
• `sendbuf` ([BufSpecB / InPlace]) –
• `recvbuf` ([BufSpecB]) –

**Return type**
`Request`

**Ialltoallv**(`sendbuf`, `recvbuf`)
Nonblocking All to All Scatter/Gather Vector

**Parameters**
• `sendbuf` ([BufSpecV / InPlace]) –
• `recvbuf` ([BufSpecV]) –

**Return type**
`Request`

**Ialltoallw**(`sendbuf`, `recvbuf`)
Nonblocking Generalized All-to-All

**Parameters**
• `sendbuf` ([BufSpecW / InPlace]) –
• `recvbuf` ([BufSpecW]) –
**Return type**

*Request*

**Ibarrier()**

Nonblocking Barrier

**Return type**

*Request*

**Ibcast(buf, root=0)**

Nonblocking Broadcast

**Parameters**

- **buf** (*BufSpec*)
- **root** (*int*)

**Return type**

*Request*

**Ibsend(buf, dest, tag=0)**

Nonblocking send in buffered mode

**Parameters**

- **buf** (*BufSpec*)
- **dest** (*int*)
- **tag** (*int*)

**Return type**

*Request*

**Idup(info=None)**

Nonblocking duplicate an existing communicator

**Parameters**

- **info** (*Info | None*)

**Return type**

*tuple*[Self, *Request]*

**Idup_with_info(info)**

Duplicate an existing communicator

**Parameters**

- **info** (*Info*)

**Return type**

*tuple*[Self, *Request]*

**Igather(sendbuf, recvbuf, root=0)**

Nonblocking Gather

**Parameters**

- **sendbuf** (*BufSpec / InPlace*)
- **recvbuf** (*BufSpecB / None*)
- **root** (*int*).
**Return type**
*Request*

**Igatherv**(*sendbuf, recvbuf, root=0*)
Nonblocking Gather Vector

**Parameters**
- *sendbuf* (*BufSpec | InPlace*)
- *recvbuf* (*BufSpecV | None*)
- *root* (*int*)

**Return type**
*Request*

**Improbe**(*source=ANY_SOURCE, tag=ANY_TAG, status=None*)
Nonblocking test for a matched message

**Parameters**
- *source* (*int*)
- *tag* (*int*)
- *status* (*Status | None*)

**Return type**
*Message | None*

**Iprobe**(*source=ANY_SOURCE, tag=ANY_TAG, status=None*)
Nonblocking test for a message

**Parameters**
- *source* (*int*)
- *tag* (*int*)
- *status* (*Status | None*)

**Return type**
*bool*

**Irecv**(*buf, source=ANY_SOURCE, tag=ANY_TAG*)
Nonblocking receive

**Parameters**
- *buf* (*BufSpec*)
- *source* (*int*)
- *tag* (*int*)

**Return type**
*Request*

**Ireduce**(*sendbuf, recvbuf, op=SUM, root=0*)
Nonblocking Reduce to Root

**Parameters**
- *sendbuf* (*BufSpec | InPlace*)
- *recvbuf* (*BufSpec | None*)
• \texttt{op (Op)} –
• \texttt{root (int)} –

\textbf{Return type}\n\textit{Request}

\textbf{Ireduce\_scatter} (\texttt{sendbuf, recvbuf, recvcounts=None, op=SUM})

Nonblocking Reduce-Scatter (vector version)

\textbf{Parameters}

• \texttt{sendbuf (BufSpec | InPlace)} –
• \texttt{recvbuf (BufSpec)} –
• \texttt{recvcounts (Sequence[int] | None)} –
• \texttt{op (Op)} –

\textbf{Return type}\n\textit{Request}

\textbf{Ireduce\_scatter\_block}(\texttt{sendbuf, recvbuf, op=SUM})

Nonblocking Reduce-Scatter Block (regular, non-vector version)

\textbf{Parameters}

• \texttt{sendbuf (BufSpecB | InPlace)} –
• \texttt{recvbuf (BufSpec | BufSpecB)} –
• \texttt{op (Op)} –

\textbf{Return type}\n\textit{Request}

\textbf{Irsend} (\texttt{buf, dest, tag=0})

Nonblocking send in ready mode

\textbf{Parameters}

• \texttt{buf (BufSpec)} –
• \texttt{dest (int)} –
• \texttt{tag (int)} –

\textbf{Return type}\n\textit{Request}

\textbf{Is\_inter}()

Test to see if a comm is an intercommunicator

\textbf{Return type}\n\textit{bool}

\textbf{Is\_intra}()

Test to see if a comm is an intracommunicator

\textbf{Return type}\n\textit{bool}
**Is_revoked()**

Indicate whether the communicator has been revoked

**Return type**

`bool`

**Iscatter**(`sendbuf, recvbuf, root=0`)

Nonblocking Scatter

**Parameters**

- `sendbuf` (BufSpecB | None)
- `recvbuf` (BufSpec | InPlace)
- `root` (int)

**Return type**

`Request`

**Iscatterv**(`sendbuf, recvbuf, root=0`)

Nonblocking Scatter Vector

**Parameters**

- `sendbuf` (BufSpecV | None)
- `recvbuf` (BufSpec | InPlace)
- `root` (int)

**Return type**

`Request`

**Isend**(`buf, dest, tag=0`)

Nonblocking send

**Parameters**

- `buf` (BufSpec)
- `dest` (int)
- `tag` (int)

**Return type**

`Request`

**Isendrecv**(`sendbuf, dest, sendtag=0, recvbuf=None, source=ANY_SOURCE, recvtag=ANY_TAG`)

Nonblocking send and receive

**Parameters**

- `sendbuf` (BufSpec)
- `dest` (int)
- `sendtag` (int)
- `recvbuf` (BufSpec | None)
- `source` (int)
- `recvtag` (int)

**Return type**

`Request`
**Isendrecv_replace***(buf, dest, sendtag=0, source=ANY_SOURCE, recvtag=ANY_TAG)***

Send and receive a message

**Note:** This function is guaranteed not to deadlock in situations where pairs of blocking sends and receives may deadlock.

**Caution:** A common mistake when using this function is to mismatch the tags with the source and destination ranks, which can result in deadlock.

**Parameters**
- **buf** (BufSpec)
- **dest** (int)
- **sendtag** (int)
- **source** (int)
- **recvtag** (int)

**Return type**
Request

**Ishrink()**
Nonblocking shrink a communicator to remove all failed processes

**Return type**
tuple[Comm, Request]

**Issend(buf, dest, tag=0)**
Nonblocking send in synchronous mode

**Parameters**
- **buf** (BufSpec)
- **dest** (int)
- **tag** (int)

**Return type**
Request

**classmethod Join(fd)**
Create a intercommunicator by joining two processes connected by a socket

**Parameters**
- **fd** (int)

**Return type**
Intercomm

**Mprobe(source=ANY_SOURCE, tag=ANY_TAG, status=None)**
Blocking test for a matched message

**Parameters**
- **source** (int)
Precv_init(buf, partitions, source=ANY_SOURCE, tag=ANY_TAG, info=INFO_NULL)
Create request for a partitioned recv operation

Parameters
- buf (BufSpec) –
- partitions (int) –
- source (int) –
- tag (int) –
- info (Info) –

Return type
Message

Probe(source=ANY_SOURCE, tag=ANY_TAG, status=None)
Blocking test for a message

Note: This function blocks until the message arrives.

Parameters
- source (int) –
- tag (int) –
- status (Status | None) –

Return type
Literal[True]

Psend_init(buf, partitions, dest, tag=0, info=INFO_NULL)
Create request for a partitioned send operation

Parameters
- buf (BufSpec) –
- partitions (int) –
- dest (int) –
- tag (int) –
- info (Info) –

Return type
Prequest
**Recv**(*buf, source=ANY_SOURCE, tag=ANY_TAG, status=None*)

Blocking receive

---

**Note:** This function blocks until the message is received

**Parameters**

- **buf** (*BufSpec*) –
- **source** (*int*) –
- **tag** (*int*) –
- **status** (*Status / None*) –

**Return type**

None

**Recv_init**(*buf, source=ANY_SOURCE, tag=ANY_TAG*)

Create a persistent request for a receive

**Parameters**

- **buf** (*BufSpec*) –
- **source** (*int*) –
- **tag** (*int*) –

**Return type**

Prequest

**Reduce**(*sendbuf, recvbuf, op=SUM, root=0*)

Reduce to Root

**Parameters**

- **sendbuf** (*BufSpec / InPlace*) –
- **recvbuf** (*BufSpec / None*) –
- **op** (*Op*) –
- **root** (*int*) –

**Return type**

None

**Reduce_init**(*sendbuf, recvbuf, op=SUM, root=0, info=INFO_NULL*)

Persistent Reduce

**Parameters**

- **sendbuf** (*BufSpec / InPlace*) –
- **recvbuf** (*BufSpec / None*) –
- **op** (*Op*) –
- **root** (*int*) –
- **info** (*Info*) –
Return type

Prequest

Reduce_scatter\((sendbuf, recvbuf, recvcounts=None, op=SUM)\)
Reduce-Scatter (vector version)

Parameters

- **sendbuf** (BufSpec | InPlace) –
- **recvbuf** (BufSpec) –
- **recvcounts** (Sequence[int] | None) –
- **op** (Op) –

Return type

None

Reduce_scatter_block\((sendbuf, recvbuf, op=SUM)\)
Reduce-Scatter Block (regular, non-vector version)

Parameters

- **sendbuf** (BufSpecB | InPlace) –
- **recvbuf** (BufSpec | BufSpecB) –
- **op** (Op) –

Return type

None

Reduce_scatter_block_init\((sendbuf, recvbuf, op=SUM, info=INFO_NULL)\)
Persistent Reduce-Scatter Block (regular, non-vector version)

Parameters

- **sendbuf** (BufSpecB | InPlace) –
- **recvbuf** (BufSpec | BufSpecB) –
- **op** (Op) –
- **info** (Info) –

Return type

Prequest

Reduce_scatter_init\((sendbuf, recvbuf, recvcounts=None, op=SUM, info=INFO_NULL)\)
Persistent Reduce-Scatter (vector version)

Parameters

- **sendbuf** (BufSpec | InPlace) –
- **recvbuf** (BufSpec) –
- **recvcounts** (Sequence[int] | None) –
- **op** (Op) –
- **info** (Info) –

Return type

Prequest
Revoke()
Revoke a communicator

Return type
None

Rsend(buf, dest, tag=0)
Blocking send in ready mode

Parameters
• buf (BufSpec) –
• dest (int) –
• tag (int) –

Return type
None

Rsend_init(buf, dest, tag=0)
Persistent request for a send in ready mode

Parameters
• buf (BufSpec) –
• dest (int) –
• tag (int) –

Return type
Request

Scatter(sendbuf, recvbuf, root=0)
Scatter data from one process to all other processes in a group

Parameters
• sendbuf (BufSpecB | None) –
• recvbuf (BufSpec / InPlace) –
• root (int) –

Return type
None

Scatter_init(sendbuf, recvbuf, root=0, info=INFO_NULL)
Persistent Scatter

Parameters
• sendbuf (BufSpecB | None) –
• recvbuf (BufSpec / InPlace) –
• root (int) –
• info (Info) –

Return type
Prequest
**Scatterv**(*sendbuf, recvbuf, root=0*)

Scatter Vector, scatter data from one process to all other processes in a group providing different amount of data and displacements at the sending side

**Parameters**

- **sendbuf** (*BufSpecV | None*) –
- **recvbuf** (*BufSpec | InPlace*) –
- **root** (*int*) –

**Return type**

None

**Scatterv_init**(*sendbuf, recvbuf, root=0, info=INFO_NULL*)

Persistent Scatter Vector

**Parameters**

- **sendbuf** (*BufSpecV | None*) –
- **recvbuf** (*BufSpec | InPlace*) –
- **root** (*int*) –
- **info** (*Info*) –

**Return type**

Prequest

**Send**(*buf, dest, tag=0*)

Blocking send

**Note:** This function may block until the message is received. Whether or not *Send* blocks depends on several factors and is implementation dependent

**Parameters**

- **buf** (*BufSpec*) –
- **dest** (*int*) –
- **tag** (*int*) –

**Return type**

None

**Send_init**(*buf, dest, tag=0*)

Create a persistent request for a standard send

**Parameters**

- **buf** (*BufSpec*) –
- **dest** (*int*) –
- **tag** (*int*) –

**Return type**

Prequest
**Sendrecv**

\[
\text{Sendrecv}(sendbuf, dest, sendtag=0, recvbuf=None, source=ANY\_SOURCE, recvtag=ANY\_TAG, status=None)
\]

Send and receive a message

**Note:** This function is guaranteed not to deadlock in situations where pairs of blocking sends and receives may deadlock.

**Caution:** A common mistake when using this function is to mismatch the tags with the source and destination ranks, which can result in deadlock.

**Parameters**

- **sendbuf (BufSpec)** –
- **dest (int)** –
- **sendtag (int)** –
- **recvbuf (BufSpec | None)** –
- **source (int)** –
- **recvtag (int)** –
- **status (Status | None)** –

**Return type**

None

**Sendrecv_replace**

\[
\text{Sendrecv\_replace}(buf, dest, sendtag=0, source=ANY\_SOURCE, recvtag=ANY\_TAG, status=None)
\]

Send and receive a message

**Note:** This function is guaranteed not to deadlock in situations where pairs of blocking sends and receives may deadlock.

**Caution:** A common mistake when using this function is to mismatch the tags with the source and destination ranks, which can result in deadlock.

**Parameters**

- **buf (BufSpec)** –
- **dest (int)** –
- **sendtag (int)** –
- **source (int)** –
- **recvtag (int)** –
- **status (Status | None)** –

**Return type**

None
**Set_attr**(keyval, attrval)

Store attribute value associated with a key

Parameters

- **keyval** (int) –
- **attrval** (Any) –

Return type

None

**Set_errhandler**(errhandler)

Set the error handler for a communicator

Parameters

- errhandler (Errhandler) –

Return type

None

**Set_info**(info)

Set new values for the hints associated with a communicator

Parameters

- info (Info) –

Return type

None

**Set_name**(name)

Set the print name for this communicator

Parameters

- name (str) –

Return type

None

**Shrink**( )

Shrink a communicator to remove all failed processes

Return type

Comm

**Split**(color=0, key=0)

Split communicator by color and key

Parameters

- color (int) –
- key (int) –

Return type

Comm

**Split_type**(split_type, key=0, info=INFO_NULL)

Split communicator by split type

Parameters

- split_type (int) –
key (int) –
info (Info) –

Return type
Comm

Ssend(buf, dest, tag=0)
Blocking send in synchronous mode

Parameters
• buf (BufSpec) –
• dest (int) –
• tag (int) –

Return type
None

Ssend_init(buf, dest, tag=0)
Persistent request for a send in synchronous mode

Parameters
• buf (BufSpec) –
• dest (int) –
• tag (int) –

Return type
Request

allgather(sendobj)
Gather to All

Parameters
sendobj (Any) –

Return type
list[Any]

allreduce(sendobj, op=SUM)
Reduce to All

Parameters
• sendobj (Any) –
• op (Op | Callable[[Any, Any], Any]) –

Return type
Any

alltoall(sendobj)
All to All Scatter/Gather

Parameters
sendobj (Sequence[Any]) –

Return type
list[Any]
barrier()
Barrier synchronization

Note: This method is equivalent to `Comm.Barrier()`

Return type
None

bcast(obj, root=0)
Broadcast

Parameters
• obj (Any) –
• root (int) –

Return type
Any

bsend(obj, dest, tag=0)
Send in buffered mode

Parameters
• obj (Any) –
• dest (int) –
• tag (int) –

Return type
None

classmethod f2py(arg)

Parameters
arg (int) –

Return type
Comm

gather(sendobj, root=0)
Gather

Parameters
• sendobj (Any) –
• root (int) –

Return type
list[Any] | None

ibsend(obj, dest, tag=0)
Nonblocking send in buffered mode

Parameters
• obj (Any) –
• dest (int) –
• **tag (int)** –

**Return type**  
Request

**improve**(*source=ANY_SOURCE, tag=ANY_TAG, status=None*)
Nonblocking test for a matched message

**Parameters**
• **source (int)** –
• **tag (int)** –
• **status (Status | None)** –

**Return type**  
Message | None

**iprobe** (*source=ANY_SOURCE, tag=ANY_TAG, status=None*)
Nonblocking test for a message

**Parameters**
• **source (int)** –
• **tag (int)** –
• **status (Status | None)** –

**Return type**  
bool

**irecv**(*buf=None, source=ANY_SOURCE, tag=ANY_TAG*)
Nonblocking receive

**Parameters**
• **buf (Buffer | None)** –
• **source (int)** –
• **tag (int)** –

**Return type**  
Request

**isend**(*obj, dest, tag=0*)
Nonblocking send

**Parameters**
• **obj (Any)** –
• **dest (int)** –
• **tag (int)** –

**Return type**  
Request

**issend**(*obj, dest, tag=0*)
Nonblocking send in synchronous mode

**Parameters**
• **obj (Any)** –
• dest (int) –
• tag (int) –

Return type
Request

mprobe(source=ANY_SOURCE, tag=ANY_TAG, status=None)
Blocking test for a matched message

Parameters
• source (int) –
• tag (int) –
• status (Status | None) –

Return type
Message

probe(source=ANY_SOURCE, tag=ANY_TAG, status=None)
Blocking test for a message

Parameters
• source (int) –
• tag (int) –
• status (Status | None) –

Return type
Literal[True]

py2f()

Return type
int

recv(buf=None, source=ANY_SOURCE, tag=ANY_TAG, status=None)
Receive

Parameters
• buf (Buffer | None) –
• source (int) –
• tag (int) –
• status (Status | None) –

Return type
Any

reduce(sendobj, op=SUM, root=0)
Reduce to Root

Parameters
• sendobj (Any) –
• op (Op | Callable[[Any, Any], Any]) –
• root (int) –
Return type
   Any | None

scatter(sendobj, root=0)
Scatter

Parameters
   • sendobj (Sequence[Any] | None) –
   • root (int) –

Return type
   Any

send(obj, dest, tag=0)
Send

Parameters
   • obj (Any) –
   • dest (int) –
   • tag (int) –

Return type
   None

sendrecv(sendobj, dest, sendtag=0, recvbuf=None, source=ANY_SOURCE, recvtag=ANY_TAG, status=None)
Send and Receive

Parameters
   • sendobj (Any) –
   • dest (int) –
   • sendtag (int) –
   • recvbuf (Buffer | None) –
   • source (int) –
   • recvtag (int) –
   • status (Status | None) –

Return type
   Any

ssend(obj, dest, tag=0)
Send in synchronous mode

Parameters
   • obj (Any) –
   • dest (int) –
   • tag (int) –

Return type
   None
Attributes Documentation

group
communicator group

info
communicator info

is_inter
is intercommunicator

is_intra
is intracommmunicator

is_topo
is a topology communicator

name
communicator name

rank
rank of this process in communicator

size
number of processes in communicator

topology
communicator topology type

mpi4py.MPI.Datatype

class mpi4py.MPI.Datatype
Bases: object
Datatype object

static __new__(cls, datatype=None)

Parameters

datatype (Datatype | None) –

Return type
Datatype

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit()</td>
<td>Commit the datatype</td>
</tr>
<tr>
<td>Create_contiguous(count)</td>
<td>Create a contiguous datatype</td>
</tr>
<tr>
<td>Create_darray(size, rank, gsizes, distris, ...)</td>
<td>Create a datatype representing an HPF-like distributed array on Cartesian process grids</td>
</tr>
<tr>
<td>Create_f90_complex(p, r)</td>
<td>Return a bounded complex datatype</td>
</tr>
<tr>
<td>Create_f90_integer(r)</td>
<td>Return a bounded integer datatype</td>
</tr>
<tr>
<td>Create_f90_real(p, r)</td>
<td>Return a bounded real datatype</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Create_hindexed</code></td>
<td>(blocklengths, displacements) Create an indexed datatype with displacements in bytes</td>
</tr>
<tr>
<td><code>Create_hindexed_block</code></td>
<td>(blocklength, displacements) Create an indexed datatype with constant-sized blocks and displacements in bytes</td>
</tr>
<tr>
<td><code>Create_hvector</code></td>
<td>(count, blocklength, stride) Create a vector (strided) datatype</td>
</tr>
<tr>
<td><code>Create_indexed</code></td>
<td>(blocklengths, displacements) Create an indexed datatype</td>
</tr>
<tr>
<td><code>Create_indexed_block</code></td>
<td>(blocklength, displacements) Create an indexed datatype with constant-sized blocks</td>
</tr>
<tr>
<td><code>Create_keyval</code></td>
<td>([copy_fn, delete_fn, nopython]) Create a new attribute key for datatypes</td>
</tr>
<tr>
<td><code>Create_resized</code></td>
<td>(lb, extent) Create a datatype with a new lower bound and extent</td>
</tr>
<tr>
<td><code>Create_struct</code></td>
<td>(blocklengths, displacements, ...) Create a datatype from a general set of block sizes, displacements and datatypes</td>
</tr>
<tr>
<td><code>Create_subarray</code></td>
<td>(sizes, subsizes, starts[, order]) Create a datatype for a subarray of a regular, multi-dimensional array</td>
</tr>
<tr>
<td><code>Create_vector</code></td>
<td>(count, blocklength, stride) Create a vector (strided) datatype</td>
</tr>
<tr>
<td><code>Delete_attr</code></td>
<td>(keyval) Delete attribute value associated with a key</td>
</tr>
<tr>
<td><code>Dup</code></td>
<td>() Duplicate a datatype</td>
</tr>
<tr>
<td><code>Free</code></td>
<td>() Free the datatype</td>
</tr>
<tr>
<td><code>Free_keyval</code></td>
<td>(keyval) Free an attribute key for datatypes</td>
</tr>
<tr>
<td><code>Get_attr</code></td>
<td>(keyval) Retrieve attribute value by key</td>
</tr>
<tr>
<td><code>Get_contents</code></td>
<td>() Retrieve the actual arguments used in the call that created a datatype</td>
</tr>
<tr>
<td><code>Get_envelope</code></td>
<td>() Return information on the number and type of input arguments used in the call that created a datatype</td>
</tr>
<tr>
<td><code>Get_extent</code></td>
<td>() Return lower bound and extent of datatype</td>
</tr>
<tr>
<td><code>Get_name</code></td>
<td>() Get the print name for this datatype</td>
</tr>
<tr>
<td><code>Get_size</code></td>
<td>() Return the number of bytes occupied by entries in the datatype</td>
</tr>
<tr>
<td><code>Get_true_extent</code></td>
<td>() Return the true lower bound and extent of a datatype</td>
</tr>
<tr>
<td><code>Match_size</code></td>
<td>(typeclass, size) Find a datatype matching a specified size in bytes</td>
</tr>
<tr>
<td><code>Pack</code></td>
<td>(inbuf, outbuf, position, comm) Pack into contiguous memory according to datatype.</td>
</tr>
<tr>
<td><code>Pack_external</code></td>
<td>(datarep, inbuf, outbuf, position) Pack into contiguous memory according to datatype, using a portable data representation (external32).</td>
</tr>
<tr>
<td><code>Pack_external_size</code></td>
<td>(datarep, count) Return the upper bound on the amount of space (in bytes) needed to pack a message according to datatype, using a portable data representation (external32).</td>
</tr>
<tr>
<td><code>Pack_size</code></td>
<td>(count, comm) Return the upper bound on the amount of space (in bytes) needed to pack a message according to datatype.</td>
</tr>
<tr>
<td><code>Set_attr</code></td>
<td>(keyval, attrval) Store attribute value associated with a key</td>
</tr>
<tr>
<td><code>Set_name</code></td>
<td>(name) Set the print name for this datatype</td>
</tr>
<tr>
<td><code>Unpack</code></td>
<td>(inbuf, position, outbuf, comm) Unpack from contiguous memory according to datatype.</td>
</tr>
<tr>
<td><code>Unpack_external</code></td>
<td>(datarep, inbuf, position, outbuf) Unpack from contiguous memory according to datatype, using a portable data representation (external32).</td>
</tr>
<tr>
<td><code>decode</code></td>
<td>() Convenience method for decoding a datatype</td>
</tr>
<tr>
<td><code>f2py</code></td>
<td>(arg) Get predefined MPI datatype from character code or type string</td>
</tr>
</tbody>
</table>
Table 3 – continued from previous page

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>py2f()</td>
<td>Get character code or type string from predefined MPI datatype</td>
</tr>
<tr>
<td>tocode()</td>
<td>Get character code or type string from predefined MPI datatype</td>
</tr>
</tbody>
</table>

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>combiner</td>
<td>datatype combiner</td>
</tr>
<tr>
<td>contents</td>
<td>datatype contents</td>
</tr>
<tr>
<td>envelope</td>
<td>datatype envelope</td>
</tr>
<tr>
<td>extent</td>
<td></td>
</tr>
<tr>
<td>is_named</td>
<td>is a named datatype</td>
</tr>
<tr>
<td>is_predefined</td>
<td>is a predefined datatype</td>
</tr>
<tr>
<td>lb</td>
<td>lower bound</td>
</tr>
<tr>
<td>name</td>
<td>datatype name</td>
</tr>
<tr>
<td>size</td>
<td></td>
</tr>
<tr>
<td>true_extent</td>
<td>true extent</td>
</tr>
<tr>
<td>true_lb</td>
<td>true lower bound</td>
</tr>
<tr>
<td>true_ub</td>
<td>true upper bound</td>
</tr>
<tr>
<td>typechar</td>
<td>character code</td>
</tr>
<tr>
<td>typestr</td>
<td>type string</td>
</tr>
<tr>
<td>ub</td>
<td>upper bound</td>
</tr>
</tbody>
</table>

Methods Documentation

Commit()
Commit the datatype

Return type
`Self`

Create_contiguous(count)
Create a contiguous datatype

Parameters

- `count (int)`

Return type
`Self`

Create_darray(size, rank, gsizes, distribs, dargs, psizes, order=ORDER_C)
Create a datatype representing an HPF-like distributed array on Cartesian process grids

Parameters

- `size (int)`
- `rank (int)`
- `gsizes (Sequence[int])`
- `distrib (Sequence[int])`
• **dargs** (Sequence[int]) –
• **psizes** (Sequence[int]) –
• **order** (int) –

Return type

`Self`

classmethod **Create_f90_complex**(p, r)

Return a bounded complex datatype

Parameters

• **p** (int) –
• **r** (int) –

Return type

`Self`

classmethod **Create_f90_integer**(r)

Return a bounded integer datatype

Parameters

• **r** (int) –

Return type

`Self`

classmethod **Create_f90_real**(p, r)

Return a bounded real datatype

Parameters

• **p** (int) –
• **r** (int) –

Return type

`Self`

**Create_hindexed**(blocklengths, displacements)

Create an indexed datatype with displacements in bytes

Parameters

• **blocklengths** (Sequence[int]) –
• **displacements** (Sequence[int]) –

Return type

`Self`

**Create_hindexed_block**(blocklength, displacements)

Create an indexed datatype with constant-sized blocks and displacements in bytes

Parameters

• **blocklength** (int) –
• **displacements** (Sequence[int]) –

Return type

`Self`
**Create_hvector**(*count, blocklength, stride*)
Create a vector (strided) datatype

**Parameters**
- count *(int)* –
- blocklength *(int)* –
- stride *(int)* –

**Return type**
*Self*

**Create_indexed**(*blocklengths, displacements*)
Create an indexed datatype

**Parameters**
- blocklengths *(Sequence[int]*) –
- displacements *(Sequence[int]*) –

**Return type**
*Self*

**Create_indexed_block**(*blocklength, displacements*)
Create an indexed datatype with constant-sized blocks

**Parameters**
- blocklength *(int)* –
- displacements *(Sequence[int]*) –

**Return type**
*Self*

**classmethod Create_keyval**(*copy_fn=None, delete_fn=None, nopython=False*)
Create a new attribute key for datatypes

**Parameters**
- copy_fn *(Callable[[Datatype, int, Any], Any] | None)* –
- delete_fn *(Callable[[Datatype, int, Any], None] | None)* –
- nopython *(bool)* –

**Return type**
*int*

**Create_resized**(*lb, extent*)
Create a datatype with a new lower bound and extent

**Parameters**
- lb *(int)* –
- extent *(int)* –

**Return type**
*Self*
classmethod Create_struct(blocklengths, displacements, datatypes)
    Create an datatype from a general set of block sizes, displacements and datatypes
    
    Parameters
    • blocklengths (Sequence[int]) –
    • displacements (Sequence[int]) –
    • datatypes (Sequence[Datatype]) –
    
    Return type
    Self

Create_subarray(sizes, subsizes, starts, order=ORDER_C)
    Create a datatype for a subarray of a regular, multidimensional array
    
    Parameters
    • sizes (Sequence[int]) –
    • subsizes (Sequence[int]) –
    • starts (Sequence[int]) –
    • order (int) –
    
    Return type
    Self

Create_vector(count, blocklength, stride)
    Create a vector (strided) datatype
    
    Parameters
    • count (int) –
    • blocklength (int) –
    • stride (int) –
    
    Return type
    Self

Delete_attr(keyval)
    Delete attribute value associated with a key
    
    Parameters
    keyval (int) –
    
    Return type
    None

Dup()
    Duplicate a datatype
    
    Return type
    Self

Free()
    Free the datatype
    
    Return type
    None
classmethod Free_keyval(keyval)
Free an attribute key for datatypes

Parameters
keyval (int) –

Return type
int

Get_attr(keyval)
Retrieve attribute value by key

Parameters
keyval (int) –

Return type
int | Any | None

Get_contents()
Retrieve the actual arguments used in the call that created a datatype

Return type
tuple[list[int], list[int], list[int], list[Datatype]]

Get_envelope()
Return information on the number and type of input arguments used in the call that created a datatype

Return type
tuple[int, int, int, int, int]

Get_extent()
Return lower bound and extent of datatype

Return type
tuple[int, int]

Get_name()
Get the print name for this datatype

Return type
str

Get_size()
Return the number of bytes occupied by entries in the datatype

Return type
int

Get_true_extent()
Return the true lower bound and extent of a datatype

Return type
tuple[int, int]

classmethod Match_size(typeclass, size)
Find a datatype matching a specified size in bytes

Parameters
• typeclass (int) –
• size (int) –
**Return type**

*Self*

**Pack** *(inbuf, outbuf, position, comm)*

Pack into contiguous memory according to datatype.

**Parameters**

- `inbuf` *(BufSpec)*
- `outbuf` *(BufSpec)*
- `position` *(int)*
- `comm` *(Comm)*

**Return type**

*int*

**Pack_external** *(datarep, inbuf, outbuf, position)*

Pack into contiguous memory according to datatype, using a portable data representation *(external32)*.

**Parameters**

- `datarep` *(str)*
- `inbuf` *(BufSpec)*
- `outbuf` *(BufSpec)*
- `position` *(int)*

**Return type**

*int*

**Pack_external_size** *(datarep, count)*

Return the upper bound on the amount of space (in bytes) needed to pack a message according to datatype, using a portable data representation *(external32)*.

**Parameters**

- `datarep` *(str)*
- `count` *(int)*

**Return type**

*int*

**Pack_size** *(count, comm)*

Return the upper bound on the amount of space (in bytes) needed to pack a message according to datatype.

**Parameters**

- `count` *(int)*
- `comm` *(Comm)*

**Return type**

*int*

**Set_attr** *(keyval, attrval)*

Store attribute value associated with a key.

**Parameters**

- `keyval` *(int)*
• `attrval (Any)` –

Return type
None

Set_name(name)

Set the print name for this datatype

Parameters
  name (str) –

Return type
None

Unpack(inbuf, position, outbuf, comm)

Unpack from contiguous memory according to datatype.

Parameters
  • inbuf (BufSpec) –
  • position (int) –
  • outbuf (BufSpec) –
  • comm (Comm) –

Return type
int

Unpack_external(datarep, inbuf, position, outbuf)

Unpack from contiguous memory according to datatype, using a portable data representation (external32).

Parameters
  • datarep (str) –
  • inbuf (BufSpec) –
  • position (int) –
  • outbuf (BufSpec) –

Return type
int

decode()

Convenience method for decoding a datatype

Return type
  tuple[Datatype, str, dict[str, Any]]

classmethod f2py(arg)

Parameters
  arg (int) –

Return type
  Datatype

classmethod fromcode(code)

Get predefined MPI datatype from character code or type string

Parameters
  code (str) –
**Return type**
```
Datatype
```

**py2f()**
```
Return type
int
```

**tocode()**
```
Get character code or type string from predefined MPI datatype

Return type
```
str
```

**Attributes Documentation**

**combiner**
```datatype combiner```

**contents**
```datatype contents```

**envelope**
```datatype envelope```

**extent**

**is_named**
```is a named datatype```

**is_predefined**
```is a predefined datatype```

**lb**
```lower bound```

**name**
```datatype name```

**size**

**true_extent**
```true extent```

**true_lb**
```true lower bound```

**true_ub**
```true upper bound```

**typechar**
```character code```

**typestr**
```type string```

**ub**
```upper bound```
class mpi4py.MPI.Distgraphcomm

Bases: Topocomm

Distributed graph topology intracommunicator

static __new__(cls, comm=None)

Parameters

comm (Distgraphcomm | None) –

Return type

Distgraphcomm

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_dist_neighbors()</td>
<td>Return adjacency information for a distributed graph topology</td>
</tr>
<tr>
<td>Get_dist_neighbors_count()</td>
<td>Return adjacency information for a distributed graph topology</td>
</tr>
</tbody>
</table>

Methods Documentation

Get_dist_neighbors()

Return adjacency information for a distributed graph topology

Return type
tuple[list[int], list[int], tuple[list[int], list[int]] | None]

Get_dist_neighbors_count()

Return adjacency information for a distributed graph topology

Return type

int

class mpi4py.MPI.Errhandler

Bases: object

Error handler

static __new__(cls, errhandler=None)

Parameters

errhandler (Errhandler | None) –

Return type

Errhandler
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free()</td>
<td>Free an error handler</td>
</tr>
<tr>
<td>f2py(arg)</td>
<td></td>
</tr>
<tr>
<td>py2f()</td>
<td></td>
</tr>
</tbody>
</table>

Methods Documentation

Free()

Free an error handler

Return type
None

classmethod f2py(arg)

Parameters
arg (int) –

Return type
Errhandler

py2f()

Return type
int

mpi4py.MPI.File

class mpi4py.MPI.File

Bases: object

File handle

static __new__(cls, file=None)

Parameters
file (File | None) –

Return type
File

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call_errhandler(errorcode)</td>
<td>Call the error handler installed on a file</td>
</tr>
<tr>
<td>Close()</td>
<td>Close a file</td>
</tr>
<tr>
<td>Create_errhandler(errhandler_fn)</td>
<td>Create a new error handler for files</td>
</tr>
<tr>
<td>Delete(filename[, info])</td>
<td>Delete a file</td>
</tr>
<tr>
<td>Get_amode()</td>
<td>Return the file access mode</td>
</tr>
<tr>
<td>Get_atomicity()</td>
<td>Return the atomicity mode</td>
</tr>
</tbody>
</table>

continues on next page
Table 4 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Get_byte_offset(offset)</code></td>
<td>Return the absolute byte position in the file corresponding to 'offset' etypes relative to the current view</td>
</tr>
<tr>
<td><code>Get_errhandler()</code></td>
<td>Get the error handler for a file</td>
</tr>
<tr>
<td><code>Get_group()</code></td>
<td>Return the group of processes that opened the file</td>
</tr>
<tr>
<td><code>Get_info()</code></td>
<td>Return the hints for a file that are currently in use</td>
</tr>
<tr>
<td><code>Get_position()</code></td>
<td>Return the current position of the individual file pointer in etype units relative to the current view</td>
</tr>
<tr>
<td><code>Get_position_shared()</code></td>
<td>Return the current position of the shared file pointer in etype units relative to the current view</td>
</tr>
<tr>
<td><code>Get_size()</code></td>
<td>Return the file size</td>
</tr>
<tr>
<td><code>Get_type_extent(datatype)</code></td>
<td>Return the extent of datatype in the file</td>
</tr>
<tr>
<td><code>Get_view()</code></td>
<td>Return the file view</td>
</tr>
<tr>
<td><code>Iread(buf)</code></td>
<td>Nonblocking read using individual file pointer</td>
</tr>
<tr>
<td><code>Iread_all(buf)</code></td>
<td>Nonblocking collective read using individual file pointer</td>
</tr>
<tr>
<td><code>Iread_at(offset, buf)</code></td>
<td>Nonblocking read using explicit offset</td>
</tr>
<tr>
<td><code>Iread_at_all(offset, buf)</code></td>
<td>Nonblocking collective read using explicit offset</td>
</tr>
<tr>
<td><code>Iwrite(buf)</code></td>
<td>Nonblocking write using individual file pointer</td>
</tr>
<tr>
<td><code>Iwrite_all(buf)</code></td>
<td>Nonblocking collective write using individual file pointer</td>
</tr>
<tr>
<td><code>Iwrite_at(offset, buf)</code></td>
<td>Nonblocking write using explicit offset</td>
</tr>
<tr>
<td><code>Iwrite_at_all(offset, buf)</code></td>
<td>Nonblocking collective write using explicit offset</td>
</tr>
<tr>
<td><code>Iwrite_shared(buf)</code></td>
<td>Nonblocking write using shared file pointer</td>
</tr>
<tr>
<td><code>Open(comm, filename[, amode, info])</code></td>
<td>Open a file</td>
</tr>
<tr>
<td><code>Preallocate(size)</code></td>
<td>Preallocate storage space for a file</td>
</tr>
<tr>
<td><code>Read(buf[, status])</code></td>
<td>Read using individual file pointer</td>
</tr>
<tr>
<td><code>Read_all(buf[, status])</code></td>
<td>Collective read using individual file pointer</td>
</tr>
<tr>
<td><code>Read_all_begin(buf)</code></td>
<td>Start a split collective read using individual file pointer</td>
</tr>
<tr>
<td><code>Read_all_end(buf[, status])</code></td>
<td>Complete a split collective read using individual file pointer</td>
</tr>
<tr>
<td><code>Read_at(offset, buf[, status])</code></td>
<td>Read using explicit offset</td>
</tr>
<tr>
<td><code>Read_at_all(offset, buf[, status])</code></td>
<td>Collective read using explicit offset</td>
</tr>
<tr>
<td><code>Read_at_all_begin(offset, buf)</code></td>
<td>Start a split collective read using explicit offset</td>
</tr>
<tr>
<td><code>Read_at_all_end(buf[, status])</code></td>
<td>Complete a split collective read using explicit offset</td>
</tr>
<tr>
<td><code>Read_ordered(buf[, status])</code></td>
<td>Collective read using shared file pointer</td>
</tr>
<tr>
<td><code>Read_ordered_begin(buf)</code></td>
<td>Start a split collective read using shared file pointer</td>
</tr>
<tr>
<td><code>Read_ordered_end(buf[, status])</code></td>
<td>Complete a split collective read using shared file pointer</td>
</tr>
<tr>
<td><code>Read_shared(buf[, status])</code></td>
<td>Read using shared file pointer</td>
</tr>
<tr>
<td><code>Seek(whence)</code></td>
<td>Update the individual file pointer</td>
</tr>
<tr>
<td><code>Seek_shared(whence)</code></td>
<td>Update the shared file pointer</td>
</tr>
<tr>
<td><code>Set_atomicity(flag)</code></td>
<td>Set the atomicity mode</td>
</tr>
<tr>
<td><code>Set_errhandler(errhandler)</code></td>
<td>Set the error handler for a file</td>
</tr>
<tr>
<td><code>Set_info(info)</code></td>
<td>Set new values for the hints associated with a file</td>
</tr>
<tr>
<td><code>Set_size(size)</code></td>
<td>Set the file size</td>
</tr>
<tr>
<td><code>Set_view([disp, etype, filetype, datarep, info])</code></td>
<td>Set the file view</td>
</tr>
<tr>
<td><code>Sync()</code></td>
<td>Causes all previous writes to be transferred to the storage device</td>
</tr>
<tr>
<td><code>Write(buf[, status])</code></td>
<td>Write using individual file pointer</td>
</tr>
<tr>
<td><code>Write_all(buf[, status])</code></td>
<td>Collective write using individual file pointer</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Write_all_begin(buf)</code></td>
<td>Start a split collective write using individual file pointer</td>
</tr>
<tr>
<td><code>Write_all_end(buf[, status])</code></td>
<td>Complete a split collective write using individual file pointer</td>
</tr>
<tr>
<td><code>Write_at(offset, buf[, status])</code></td>
<td>Write using explicit offset</td>
</tr>
<tr>
<td><code>Write_at_all(offset, buf[, status])</code></td>
<td>Collective write using explicit offset</td>
</tr>
<tr>
<td><code>Write_at_all_begin(offset, buf)</code></td>
<td>Start a split collective write using explicit offset</td>
</tr>
<tr>
<td><code>Write_at_all_end(buf[, status])</code></td>
<td>Complete a split collective write using explicit offset</td>
</tr>
<tr>
<td><code>Write_ordered(buf[, status])</code></td>
<td>Collective write using shared file pointer</td>
</tr>
<tr>
<td><code>Write_ordered_begin(buf)</code></td>
<td>Start a split collective write using shared file pointer</td>
</tr>
<tr>
<td><code>Write_ordered_end(buf[, status])</code></td>
<td>Complete a split collective write using shared file pointer</td>
</tr>
<tr>
<td><code>Write_shared(buf[, status])</code></td>
<td>Write using shared file pointer</td>
</tr>
</tbody>
</table>

### Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>amode</code></td>
<td>file access mode</td>
</tr>
<tr>
<td><code>atomicity</code></td>
<td></td>
</tr>
<tr>
<td><code>group</code></td>
<td>file group</td>
</tr>
<tr>
<td><code>info</code></td>
<td>file info</td>
</tr>
<tr>
<td><code>size</code></td>
<td>file size</td>
</tr>
</tbody>
</table>

### Methods Documentation

**`Call_errhandler(errorcode)`**
Call the error handler installed on a file

**Parameters**

- `errorcode (int)`

**Return type**

- `int`

**`Close()`**
Close a file

**Return type**

- `None`

**classmethod `Create_errhandler(errhandler_fn)`**
Create a new error handler for files

**Parameters**

- `errhandler_fn (Callable[[File, int], None])`

**Return type**

- `Errhandler`
classmethod Delete(filename, info=INFO_NULL)
    Delete a file

    Parameters
    • filename (PathLike | str | bytes) –
    • info (Info) –

    Return type
    None

Get_amode()
    Return the file access mode

    Return type
    int

Get_atomicity()
    Return the atomicity mode

    Return type
    bool

Get_byte_offset(offset)
    Return the absolute byte position in the file corresponding to 'offset' etypes relative to the current view

    Parameters
    • offset (int) –

    Return type
    int

Get_errhandler()
    Get the error handler for a file

    Return type
    Errhandler

Get_group()
    Return the group of processes that opened the file

    Return type
    Group

Get_info()
    Return the hints for a file that are currently in use

    Return type
    Info

Get_position()
    Return the current position of the individual file pointer in etype units relative to the current view

    Return type
    int

Get_position_shared()
    Return the current position of the shared file pointer in etype units relative to the current view

    Return type
    int
Get_size()
Return the file size

Return type
int

Get_type_extent(datatype)
Return the extent of datatype in the file

Parameters
datatype (Datatype) –

Return type
int

Get_view()
Return the file view

Return type
tuple[int, Datatype, Datatype, str]

Iread(buf)
Nonblocking read using individual file pointer

Parameters
buf (BufSpec) –

Return type
Request

Iread_all(buf)
Nonblocking collective read using individual file pointer

Parameters
buf (BufSpec) –

Return type
Request

Iread_at(offset, buf)
Nonblocking read using explicit offset

Parameters
• offset (int) –
• buf (BufSpec) –

Return type
Request

Iread_at_all(offset, buf)
Nonblocking collective read using explicit offset

Parameters
• offset (int) –
• buf (BufSpec) –

Return type
Request
**Iread_shared(buf)**
Nonblocking read using shared file pointer

**Parameters**
buf (BufSpec) –

**Return type**
Request

**Iwrite(buf)**
Nonblocking write using individual file pointer

**Parameters**
buf (BufSpec) –

**Return type**
Request

**Iwrite_all(buf)**
Nonblocking collective write using individual file pointer

**Parameters**
buf (BufSpec) –

**Return type**
Request

**Iwrite_at(offset, buf)**
Nonblocking write using explicit offset

**Parameters**
- offset (int) –
- buf (BufSpec) –

**Return type**
Request

**Iwrite_at_all(offset, buf)**
Nonblocking collective write using explicit offset

**Parameters**
- offset (int) –
- buf (BufSpec) –

**Return type**
Request

**Iwrite_shared(buf)**
Nonblocking write using shared file pointer

**Parameters**
buf (BufSpec) –

**Return type**
Request

**classmethod Open(comm, filename, amode=MODE_RDONLY, info=INFO_NULL)**
Open a file

**Parameters**
• comm (Intracomm) –
• filename (PathLike | str | bytes) –
• amode (int) –
• info (Info) –

Return type

Self

Preallocate (size)
Preallocate storage space for a file

Parameters

size (int) –

Return type

None

Read (buf, status=None)
Read using individual file pointer

Parameters

• buf (BufSpec) –
• status (Status | None) –

Return type

None

Read_all (buf, status=None)
Collective read using individual file pointer

Parameters

• buf (BufSpec) –
• status (Status | None) –

Return type

None

Read_all_begin (buf)
Start a split collective read using individual file pointer

Parameters

buf (BufSpec) –

Return type

None

Read_all_end (buf, status=None)
Complete a split collective read using individual file pointer

Parameters

• buf (BufSpec) –
• status (Status | None) –

Return type

None
**Read_at** *(offset, buf, status=None)*
Read using explicit offset

**Parameters**
- **offset** *(int)* –
- **buf** *(BufSpec)* –
- **status** *(Status | None)* –

**Return type**
None

**Read_at_all** *(offset, buf, status=None)*
Collective read using explicit offset

**Parameters**
- **offset** *(int)* –
- **buf** *(BufSpec)* –
- **status** *(Status | None)* –

**Return type**
None

**Read_at_all_begin** *(offset, buf)*
Start a split collective read using explicit offset

**Parameters**
- **offset** *(int)* –
- **buf** *(BufSpec)* –

**Return type**
None

**Read_at_all_end** *(buf, status=None)*
Complete a split collective read using explicit offset

**Parameters**
- **buf** *(BufSpec)* –
- **status** *(Status | None)* –

**Return type**
None

**Read_ordered** *(buf, status=None)*
Collective read using shared file pointer

**Parameters**
- **buf** *(BufSpec)* –
- **status** *(Status | None)* –

**Return type**
None
Read_ordered_begin(buf)
Start a split collective read using shared file pointer

Parameters
buf (BufSpec) –

Return type
None

Read_ordered_end(buf, status=None)
Complete a split collective read using shared file pointer

Parameters
• buf (BufSpec) –
• status (Status | None) –

Return type
None

Read_shared(buf, status=None)
Read using shared file pointer

Parameters
• buf (BufSpec) –
• status (Status | None) –

Return type
None

Seek(offset, whence=SEEK_SET)
Update the individual file pointer

Parameters
• offset (int) –
• whence (int) –

Return type
None

Seek_shared(offset, whence=SEEK_SET)
Update the shared file pointer

Parameters
• offset (int) –
• whence (int) –

Return type
None

Set_atomicity(flag)
Set the atomicity mode

Parameters
flag (bool) –

Return type
None
Set_errhandler(\textit{errhandler})

Set the error handler for a file

\textbf{Parameters}

\textit{errhandler} (Errhandler) –

\textbf{Return type}

None

Set_info\textit{(info)}

Set new values for the hints associated with a file

\textbf{Parameters}

\textit{info} (Info) –

\textbf{Return type}

None

Set_size\textit{(size)}

Set the file size

\textbf{Parameters}

\textit{size} (int) –

\textbf{Return type}

None

Set_view\textit{(disp=0, etype=BYTE, filetype=None, datarep='native', info=INFO_NULL)}

Set the file view

\textbf{Parameters}

\textbf{•} \textit{disp} (int) –
\textbf{•} \textit{etype} (Datatype) –
\textbf{•} \textit{filetype} (Datatype | None) –
\textbf{•} \textit{datarep} (str) –
\textbf{•} \textit{info} (Info) –

\textbf{Return type}

None

Sync()

Causes all previous writes to be transferred to the storage device

\textbf{Return type}

None

Write\textit{(buf, status=None)}

Write using individual file pointer

\textbf{Parameters}

\textbf{•} \textit{buf} (BufSpec) –
\textbf{•} \textit{status} (Status | None) –

\textbf{Return type}

None
Write_all(buf, status=None)
Collective write using individual file pointer

Parameters

• buf (BufSpec) –
• status (Status | None) –

Return type
None

Write_all_begin(buf)
Start a split collective write using individual file pointer

Parameters

buf (BufSpec) –

Return type
None

Write_all_end(buf, status=None)
Complete a split collective write using individual file pointer

Parameters

• buf (BufSpec) –
• status (Status | None) –

Return type
None

Write_at(offset, buf, status=None)
Write using explicit offset

Parameters

• offset (int) –
• buf (BufSpec) –
• status (Status | None) –

Return type
None

Write_at_all(offset, buf, status=None)
Collective write using explicit offset

Parameters

• offset (int) –
• buf (BufSpec) –
• status (Status | None) –

Return type
None

Write_at_all_begin(offset, buf)
Start a split collective write using explicit offset

Parameters
• offset (int) –

• buf (BufSpec) –

Return type
None

Write_at_all_end(buf, status=None)
Complete a split collective write using explicit offset

Parameters

• buf (BufSpec) –

• status (Status | None) –

Return type
None

Write_ordered(buf, status=None)
Collective write using shared file pointer

Parameters

• buf (BufSpec) –

• status (Status | None) –

Return type
None

Write_ordered_begin(buf)
Start a split collective write using shared file pointer

Parameters

buf (BufSpec) –

Return type
None

Write_ordered_end(buf, status=None)
Complete a split collective write using shared file pointer

Parameters

• buf (BufSpec) –

• status (Status | None) –

Return type
None

Write_shared(buf, status=None)
Write using shared file pointer

Parameters

• buf (BufSpec) –

• status (Status | None) –

Return type
None
classmethod f2py(arg)

Parameters
arg (int) –

Return type
File

py2f()

Return type
int

Attributes Documentation

amode
    file access mode
atomicity
group
    file group
info
    file info
size
    file size

mpi4py.MPI.Graphcomm

class mpi4py.MPI.Graphcomm
    Bases: Topocomm
    General graph topology intracommunicator

static __new__(cls, comm=None)

Parameters
comm (Graphcomm | None) –

Return type
Graphcomm

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_dims()</td>
<td>Return the number of nodes and edges</td>
</tr>
<tr>
<td>Get_neighbors(rank)</td>
<td>Return list of neighbors of a process</td>
</tr>
<tr>
<td>Get_neighbors_count(rank)</td>
<td>Return number of neighbors of a process</td>
</tr>
<tr>
<td>Get_topo()</td>
<td>Return index and edges</td>
</tr>
</tbody>
</table>
Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dims</td>
<td>number of nodes and edges</td>
</tr>
<tr>
<td>edges</td>
<td></td>
</tr>
<tr>
<td>index</td>
<td></td>
</tr>
<tr>
<td>nedges</td>
<td>number of edges</td>
</tr>
<tr>
<td>neighbors</td>
<td></td>
</tr>
<tr>
<td>nneighbors</td>
<td>number of neighbors</td>
</tr>
<tr>
<td>nnodes</td>
<td>number of nodes</td>
</tr>
<tr>
<td>topo</td>
<td>topology information</td>
</tr>
</tbody>
</table>

Methods Documentation

**Get_dims()**

Return the number of nodes and edges

**Return type**

tuple[int, int]

**Get_neighbors(rank)**

Return list of neighbors of a process

**Parameters**

- rank (int)

**Return type**

list[int]

**Get_neighbors_count(rank)**

Return number of neighbors of a process

**Parameters**

- rank (int)

**Return type**

int

**Get_topo()**

Return index and edges

**Return type**

tuple[list[int], list[int]]
Attributes Documentation

dims
  number of nodes and edges
edges
index
nedges
  number of edges
neighbors
nneighbors
  number of neighbors
nnodes
  number of nodes
topo
  topology information

mpi4py.MPI.Grequest

class mpi4py.MPI.Grequest
  Bases: Request
  Generalized request handle
  static __new__(cls, request=None)
    Parameters
      request (Grequest | None) –
    Return type
      Grequest

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete()</td>
<td>Notify that a user-defined request is complete</td>
</tr>
<tr>
<td>Start()</td>
<td>Create and return a user-defined request</td>
</tr>
</tbody>
</table>

Methods Documentation

Complete()
  Notify that a user-defined request is complete

  Return type
    None
classmethod Start(query_fn, free_fn, cancel_fn, args=None, kwargs=None)

Create and return a user-defined request

Parameters
  • query_fn (Callable[..., None]) –
  • free_fn (Callable[..., None]) –
  • cancel_fn (Callable[..., None]) –
  • args (tuple[Any] | None) –
  • kwargs (dict[str, Any] | None) –

Return type
  Grequest

mpi4py.MPI.Group

class mpi4py.MPI.Group
  Bases: object

Group of processes

static __new__(cls, group=None)

Parameters
  group (Group | None) –

Return type
  Group

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare(group)</td>
<td>Compare two groups</td>
</tr>
<tr>
<td>Create_from_session_pset(session, pset_name)</td>
<td>Create a new group from session and process set</td>
</tr>
<tr>
<td>Difference(group1, group2)</td>
<td>Create a new group from the difference of two existing groups</td>
</tr>
<tr>
<td>Dup()</td>
<td>Duplicate a group</td>
</tr>
<tr>
<td>Excl(ranks)</td>
<td>Create a new group by excluding listed members</td>
</tr>
<tr>
<td>Free()</td>
<td>Free a group</td>
</tr>
<tr>
<td>Get_rank()</td>
<td>Return the rank of this process in a group</td>
</tr>
<tr>
<td>Get_size()</td>
<td>Return the size of a group</td>
</tr>
<tr>
<td>Incl(ranks)</td>
<td>Create a new group by including listed members</td>
</tr>
<tr>
<td>Intersection(group1, group2)</td>
<td>Create a new group from the intersection of two existing groups</td>
</tr>
<tr>
<td>Range_excl(ranks)</td>
<td>Create a new group by excluding ranges of members</td>
</tr>
<tr>
<td>Range_incl(ranks)</td>
<td>Create a new group by including ranges of members</td>
</tr>
<tr>
<td>Translate_ranks([ranks, group])</td>
<td>Translate ranks of processes in one group to those in another group</td>
</tr>
<tr>
<td>Union(group1, group2)</td>
<td>Create a new group from the union of two existing groups</td>
</tr>
<tr>
<td>f2py(arg)</td>
<td></td>
</tr>
<tr>
<td>py2f()</td>
<td></td>
</tr>
</tbody>
</table>
Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
<td>rank of this process in group</td>
</tr>
<tr>
<td>size</td>
<td>number of processes in group</td>
</tr>
</tbody>
</table>

Methods Documentation

**Compare**(*group*)

Compare two groups

**Parameters**

- **group** (*Group*) –

**Return type**

`int`

**classmethod Create_from_session_pset**(*session, pset_name*)

Create a new group from session and process set

**Parameters**

- **session** (*Session*) –
- **pset_name** (*str*) –

**Return type**

`Self`

**classmethod Difference**(*group1, group2*)

Create a new group from the difference of two existing groups

**Parameters**

- **group1** (*Group*) –
- **group2** (*Group*) –

**Return type**

`Self`

**Dup()**

Duplicate a group

**Return type**

`Self`

**Excl(ranks)**

Create a new group by excluding listed members

**Parameters**

- **ranks** (*Sequence[int]*) –

**Return type**

`Self`

**Free()**

Free a group

**Return type**

`None`
Get_rank()
    Return the rank of this process in a group
    
    Return type
    int

Get_size()
    Return the size of a group
    
    Return type
    int

Incl(ranks)
    Create a new group by including listed members
    
    Parameters
    ranks (Sequence[int]) –
    
    Return type
    Self

class method Intersection(group1, group2)
    Create a new group from the intersection of two existing groups
    
    Parameters
    • group1 (Group) –
    • group2 (Group) –
    
    Return type
    Self

Range_excl(ranks)
    Create a new group by excluding ranges of members
    
    Parameters
    ranks (Sequence[tuple[int, int, int]]) –
    
    Return type
    Self

Range_incl(ranks)
    Create a new group by including ranges of members
    
    Parameters
    ranks (Sequence[tuple[int, int, int]]) –
    
    Return type
    Self

Translate_ranks(ranks=None, group=None)
    Translate ranks of processes in one group to those in another group
    
    Parameters
    • ranks (Sequence[int] | None) –
    • group (Group | None) –
    
    Return type
    list[int]
classmethod Union(group1, group2)

Create a new group from the union of two existing groups

Parameters

• group1 (Group) –
• group2 (Group) –

Return type

Self

classmethod f2py(arg)

Parameters

arg (int) –

Return type

Group

py2f()

Return type

int

Attributes Documentation

rank

rank of this process in group

size

number of processes in group

mpi4py.MPI.InPlaceType

class mpi4py.MPI.InPlaceType

Bases: int

Type of IN_PLACE

static __new__(cls)

Return type

InPlaceType

mpi4py.MPI.Info

class mpi4py.MPI.Info

Bases: object

Info object

static __new__(cls, info=None)

Parameters

info (Info | None) –

Return type

Info
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create</strong>([items])</td>
<td>Create a new info object</td>
</tr>
<tr>
<td><strong>Create_env</strong>([args])</td>
<td>Create a new environment info object</td>
</tr>
<tr>
<td><strong>Delete</strong>(key)</td>
<td>Remove a (key, value) pair from info</td>
</tr>
<tr>
<td><strong>Dup</strong>()</td>
<td>Duplicate an existing info object, creating a new object, with the same (key, value) pairs and the same ordering of keys</td>
</tr>
<tr>
<td><strong>Free</strong>()</td>
<td>Free an info object</td>
</tr>
<tr>
<td><strong>Get</strong>(key)</td>
<td>Retrieve the value associated with a key</td>
</tr>
<tr>
<td><strong>Get_nkeys</strong>()</td>
<td>Return the number of currently defined keys in info</td>
</tr>
<tr>
<td><strong>Get_nthkey</strong>(n)</td>
<td>Return the nth defined key in info.</td>
</tr>
<tr>
<td><strong>Set</strong>(key, value)</td>
<td>Add the (key, value) pair to info, and overrides the value if a value for the same key was previously set</td>
</tr>
<tr>
<td><strong>clear</strong>()</td>
<td>info clear</td>
</tr>
<tr>
<td><strong>copy</strong>()</td>
<td>info copy</td>
</tr>
<tr>
<td><strong>f2py</strong>(arg)</td>
<td></td>
</tr>
<tr>
<td><strong>get</strong>(key[, default])</td>
<td>info get</td>
</tr>
<tr>
<td><strong>items</strong>()</td>
<td>info items</td>
</tr>
<tr>
<td><strong>keys</strong>()</td>
<td>info keys</td>
</tr>
<tr>
<td><strong>pop</strong>(key, *default)</td>
<td>info pop</td>
</tr>
<tr>
<td><strong>popitem</strong>()</td>
<td>info popitem</td>
</tr>
<tr>
<td><strong>py2f</strong>()</td>
<td></td>
</tr>
<tr>
<td><strong>update</strong>([items])</td>
<td>info update</td>
</tr>
<tr>
<td><strong>values</strong>()</td>
<td>info values</td>
</tr>
</tbody>
</table>

Methods Documentation

**classmethod** **Create**(items=None)
Create a new info object

Parameters

- **items** (*Info | Mapping[*str*, *str*] | Iterable[*tuple*[str, str]] | None*) –

Return type

- **Self**

**classmethod** **Create_env**(args=None)
Create a new environment info object

Parameters

- **args** (*Sequence*[str] | None*) –

Return type

- **Self**

**Delete**(key)
Remove a (key, value) pair from info

Parameters

- **key** (*str*) –
Return type
None

Dup()
Duplicate an existing info object, creating a new object, with the same (key, value) pairs and the same ordering of keys

Return type
Self

Free()
Free an info object

Return type
None

Get(key)
Retrieve the value associated with a key

Parameters
key (str) –

Return type
str | None

Get_nkeys()
Return the number of currently defined keys in info

Return type
int

Get_nthkey(n)
Return the nth defined key in info. Keys are numbered in the range [0, N) where N is the value returned by Info.Get_nkeys()

Parameters
n (int) –

Return type
str

Set(key, value)
Add the (key, value) pair to info, and overrides the value if a value for the same key was previously set

Parameters
• key (str) –
• value (str) –

Return type
None

clear()

info clear

Return type
None

copy()

info copy
Return type

    Self

classmethod f2py(arg)

Parameters

    arg (int) –

Return type

    Info

get(key, default=None)

Parameters

    • key (str) –
    • default (str | None) –

Return type

    str | None

items()

Return type

    list[tuple[str, str]]

delkeys()

Return type

    list[str]

pop(key, *default)

Parameters

    • key (str) –
    • default (str) –

Return type

    str

delitem()

Return type

    tuple[str, str]

py2f()

Return type

    int

update(items=(), **kwds)

Parameters

    • items (Info | Mapping[str, str] | Iterable[tuple[str, str]]) –
• **kwds** *(str)*

`Return type`

None

**values()**

info values

`Return type`

list[str]

---

**mpi4py.MPI.Intercomm**

**class mpi4py.MPI.Intercomm**

Bases: *Comm*

Intercommunicator

**static **`__new__`**(cls, comm=None)**

**Parameters**

- **comm** *(Intercomm | None)*

**Return type**

Intercomm

---

**Methods Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Create_from_groups(local_group, ...[, ...])</code></td>
<td>Create communicator from group</td>
</tr>
<tr>
<td><code>Get_remote_group()</code></td>
<td>Access the remote group associated with the inter-communicator</td>
</tr>
<tr>
<td><code>Get_remote_size()</code></td>
<td>Intercommunicator remote size</td>
</tr>
<tr>
<td><code>Merge([high])</code></td>
<td>Merge intercommunicator</td>
</tr>
</tbody>
</table>

---

**Attributes Summary**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>remote_group</code></td>
<td>remote group</td>
</tr>
<tr>
<td><code>remote_size</code></td>
<td>number of remote processes</td>
</tr>
</tbody>
</table>

---

**Methods Documentation**

**classmethod Create_from_groups**(local_group, local_leader, remote_group, remote_leader, stringtag='org.mpi4py', info=INFO_NULL, errhandler=None)

Create communicator from group

**Parameters**

- **local_group** *(Group)*
- **local_leader** *(int)*
- **remote_group** *(Group)*
• **remote_leader** (*int*) –
• **stringtag** (*str*) –
• **info** (*Info*) –
• **errhandler** (*Errhandler | None*) –

**Return type**
*Intracomm*

**Get_remote_group()**
Access the remote group associated with the inter-communicator

**Return type**
*Group*

**Get_remote_size()**
Intercommunicator remote size

**Return type**
*int*

**Merge** (*high=False*)
Merge intercommunicator

**Parameters**
• **high** (*bool*) –

**Return type**
*Intracomm*

**Attributes Documentation**

**remote_group**
remote group

**remote_size**
number of remote processes

**mpi4py.MPI.Intracomm**

**class** mpi4py.MPI.Intracomm

Bases: *Comm*

Intrcommunicator

**static **new**(cls, comm=None)**

**Parameters**
• **comm** (*Intracomm | None*) –

**Return type**
*Intracomm*
## Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accept</strong> <em>(port_name[, info, root])</em></td>
<td>Accept a request to form a new intercommunicator</td>
</tr>
<tr>
<td><strong>Cart_map</strong> <em>(dims[, periods])</em></td>
<td>Return an optimal placement for the calling process on the physical machine</td>
</tr>
<tr>
<td><strong>Connect</strong> <em>(port_name[, info, root])</em></td>
<td>Make a request to form a new intercommunicator</td>
</tr>
<tr>
<td><strong>Create_cart</strong> <em>(dims[, periods, reorder])</em></td>
<td>Create cartesian communicator</td>
</tr>
<tr>
<td><strong>Create_dist_graph</strong> <em>(sources, degrees, destinations)</em></td>
<td>Create distributed graph communicator</td>
</tr>
<tr>
<td><strong>Create_dist_graph_adjacent</strong> <em>(sources, destinations)</em></td>
<td>Create distributed graph communicator</td>
</tr>
<tr>
<td><strong>Create_from_group</strong> <em>(group[, stringtag, info, ...])</em></td>
<td>Create communicator from group</td>
</tr>
<tr>
<td><strong>Create_graph</strong> <em>(index, edges[, reorder])</em></td>
<td>Create graph communicator</td>
</tr>
<tr>
<td><strong>Create_intercomm</strong> <em>(local_leader, peer_comm, ...)</em></td>
<td>Create intercommunicator</td>
</tr>
<tr>
<td><strong>Exscan</strong> <em>(sendbuf, recvbuf[, op])</em></td>
<td>Exclusive Scan</td>
</tr>
<tr>
<td><strong>Exscan_init</strong> <em>(sendbuf, recvbuf[, op, info])</em></td>
<td>Inclusive Scan</td>
</tr>
<tr>
<td><strong>Graph_map</strong> <em>(index, edges)</em></td>
<td>Return an optimal placement for the calling process on the physical machine</td>
</tr>
<tr>
<td><strong>Iexscan</strong> <em>(sendbuf, recvbuf[, op])</em></td>
<td>Inclusive Scan</td>
</tr>
<tr>
<td><strong>Iscan</strong> <em>(sendbuf, recvbuf[, op])</em></td>
<td>Inclusive Scan</td>
</tr>
<tr>
<td><strong>Scan</strong> <em>(sendbuf, recvbuf[, op])</em></td>
<td>Inclusive Scan</td>
</tr>
<tr>
<td><strong>Scan_init</strong> <em>(sendbuf, recvbuf[, op, info])</em></td>
<td>Inclusive Scan</td>
</tr>
<tr>
<td><strong>Spawn</strong> <em>(command[, args, maxprocs, info, root, ...])</em></td>
<td>Spawn instances of a single MPI application</td>
</tr>
<tr>
<td><strong>Spawn_multiple</strong> <em>(command[, args, maxprocs, ...])</em></td>
<td>Spawn instances of multiple MPI applications</td>
</tr>
<tr>
<td><strong>exscan</strong> <em>(sendobj[, op])</em></td>
<td>Exclusive Scan</td>
</tr>
<tr>
<td><strong>scan</strong> <em>(sendobj[, op])</em></td>
<td>Inclusive Scan</td>
</tr>
</tbody>
</table>

## Methods Documentation

**Accept** *(port_name, info=INFO_NULL, root=0)*

Accept a request to form a new intercommunicator

**Parameters**

- **port_name** *(str)* –
- **info** *(Info)* –
- **root** *(int)* –

**Return type**

Intercomm

**Cart_map** *(dims, periods=None)*

Return an optimal placement for the calling process on the physical machine

**Parameters**

- **dims** *(Sequence[int])* –
- **periods** *(Sequence[bool] | None)* –

**Return type**

int
**Connect** *(port_name, info=INFO_NULL, root=0)*

Make a request to form a new intercommunicator

**Parameters**
- *port_name (str)* –
- *info (Info)* –
- *root (int)* –

**Return type**
Intercomm

**Create_cart** *(dims, periods=None, reorder=False)*

Create cartesian communicator

**Parameters**
- *dims (Sequence[int])* –
- *periods (Sequence[bool] | None)* –
- *reorder (bool)* –

**Return type**
Cartcomm

**Create_dist_graph** *(sources, degrees, destinations, weights=None, info=INFO_NULL, reorder=False)*

Create distributed graph communicator

**Parameters**
- *sources (Sequence[int])* –
- *degrees (Sequence[int])* –
- *destinations (Sequence[int])* –
- *weights (Sequence[int] | None)* –
- *info (Info)* –
- *reorder (bool)* –

**Return type**
Distgraphcomm

**Create_dist_graph_adjacent** *(sources, destinations, sourceweights=None, destweights=None, info=INFO_NULL, reorder=False)*

Create distributed graph communicator

**Parameters**
- *sources (Sequence[int])* –
- *destinations (Sequence[int])* –
- *sourceweights (Sequence[int] | None)* –
- *destweights (Sequence[int] | None)* –
- *info (Info)* –
- *reorder (bool)* –
Return type
Distgraphcomm
classmethod Create_from_group(group, stringtag='org.mpi4py', info=INFO_NULL, errhandler=None)
Create communicator from group

Parameters
• group (Group) –
• stringtag (str) –
• info (Info) –
• errhandler (Errhandler | None) –

Return type
Intracomm
Create_graph(index, edges, reorder=False)
Create graph communicator

Parameters
• index (Sequence[int]) –
• edges (Sequence[int]) –
• reorder (bool) –

Return type
Graphcomm
Create_group(group, tag=0)
Create communicator from group

Parameters
• group (Group) –
• tag (int) –

Return type
Intracomm
Create_intercomm(local_leader, peer_comm, remote_leader, tag=0)
Create intercommunicator

Parameters
• local_leader (int) –
• peer_comm (Intracomm) –
• remote_leader (int) –
• tag (int) –

Return type
Intercomm
Exscan(sendbuf, recvbuf, op=SUM)
Exclusive Scan

Parameters
• sendbuf (BufSpec | InPlace) –
recvbuf (BufSpec) –
• op (Op) –

Return type
None

Exscan_init (sendbuf, recvbuf, op=SUM, info=INFO_NULL)
Inclusive Scan

Parameters
• sendbuf (BufSpec | InPlace) –
• recvbuf (BufSpec) –
• op (Op) –
• info (Info) –

Return type
Prequest

Graph_map (index, edges)
Return an optimal placement for the calling process on the physical machine

Parameters
• index (Sequence[int]) –
• edges (Sequence[int]) –

Return type
int

Iexscan (sendbuf, recvbuf, op=SUM)
Inclusive Scan

Parameters
• sendbuf (BufSpec | InPlace) –
• recvbuf (BufSpec) –
• op (Op) –

Return type
Request

Iscan (sendbuf, recvbuf, op=SUM)
Inclusive Scan

Parameters
• sendbuf (BufSpec | InPlace) –
• recvbuf (BufSpec) –
• op (Op) –

Return type
Request
**Scan**(sendbuf, recvbuf, op=SUM)

Inclusive Scan

**Parameters**

- sendbuf (BufSpec | InPlace)
- recvbuf (BufSpec)
- op (Op)

**Return type**

None

**Scan_init**(sendbuf, recvbuf, op=SUM, info=INFO_NULL)

Inclusive Scan

**Parameters**

- sendbuf (BufSpec | InPlace)
- recvbuf (BufSpec)
- op (Op)
- info (Info)

**Return type**

Prequest

**Spawn**(command, args=None, maxprocs=1, info=INFO_NULL, root=0, errcodes=None)

Spawn instances of a single MPI application

**Parameters**

- command (str)
- args (Sequence[str] | None)
- maxprocs (int)
- info (Info)
- root (int)
- errcodes (list[int] | None)

**Return type**

Intercomm

**Spawn_multiple**(command, args=None, maxprocs=None, info=INFO_NULL, root=0, errcodes=None)

Spawn instances of multiple MPI applications

**Parameters**

- command (Sequence[str])
- args (Sequence[Sequence[str]] | None)
- maxprocs (Sequence[int] | None)
- info (Sequence[Info] | Info)
- root (int)
- errcodes (list[list[int]] | None)
Return type
Intercomm

**exscan**(*sendobj, op=SUM*)
Exclusive Scan

Parameters
- *sendobj* (*Any*)
- *op* (*Op | Callable[[*Any*, *Any], *Any]*)

Return type
*Any*

**scan**(*sendobj, op=SUM*)
Inclusive Scan

Parameters
- *sendobj* (*Any*)
- *op* (*Op | Callable[[*Any*, *Any], *Any]*)

Return type
*Any*

**mpi4py.MPI.Message**

**class** *mpi4py.MPI.Message*
Bases: *object*
Matched message handle

**static __new__(cls, message=None)**

Parameters
- *message* (*Message | None*)

Return type
*Message*

**Methods Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iprobe</td>
<td>Nonblocking test for a matched message</td>
</tr>
<tr>
<td>Irecev</td>
<td>Nonblocking receive of matched message</td>
</tr>
<tr>
<td>Probe</td>
<td>Blocking test for a matched message</td>
</tr>
<tr>
<td>Recv</td>
<td>Blocking receive of matched message</td>
</tr>
<tr>
<td>f2py</td>
<td></td>
</tr>
<tr>
<td>iprobe</td>
<td>Nonblocking test for a matched message</td>
</tr>
<tr>
<td>irecv</td>
<td>Nonblocking receive of matched message</td>
</tr>
<tr>
<td>probe</td>
<td>Blocking test for a matched message</td>
</tr>
<tr>
<td>py2f</td>
<td></td>
</tr>
<tr>
<td>recv</td>
<td>Blocking receive of matched message</td>
</tr>
</tbody>
</table>
Methods Documentation

classmethod Iprobe(comm, source=ANY_SOURCE, tag=ANY_TAG, status=None)
    Nonblocking test for a matched message

    Parameters
    • comm (Comm) –
    • source (int) –
    • tag (int) –
    • status (Status | None) –

    Return type
    Self | None

Irecv(buf)
    Nonblocking receive of matched message

    Parameters
    • buf (BufSpec) –

    Return type
    Request

classmethod Probe(comm, source=ANY_SOURCE, tag=ANY_TAG, status=None)
    Blocking test for a matched message

    Parameters
    • comm (Comm) –
    • source (int) –
    • tag (int) –
    • status (Status | None) –

    Return type
    Self

Recv(buf, status=None)
    Blocking receive of matched message

    Parameters
    • buf (BufSpec) –
    • status (Status | None) –

    Return type
    None

classmethod f2py(arg)

    Parameters
    • arg (int) –

    Return type
    Message
classmethod iprobe(comm, source=ANY_SOURCE, tag=ANY_TAG, status=None)
    Nonblocking test for a matched message
    Parameters
    • comm (Comm) –
    • source (int) –
    • tag (int) –
    • status (Status | None) –
    Return type
    Self | None

irecv()  
    Nonblocking receive of matched message
    Return type
    Request

classmethod probe(comm, source=ANY_SOURCE, tag=ANY_TAG, status=None)
    Blocking test for a matched message
    Parameters
    • comm (Comm) –
    • source (int) –
    • tag (int) –
    • status (Status | None) –
    Return type
    Self

py2f()
    Return type
    int

recv(status=None)
    Blocking receive of matched message
    Parameters
    • status (Status | None) –
    Return type
    Any

mpi4py.MPI.Op
class mpi4py.MPI.Op  
    Bases: object
    Operation object
    static __new__(cls, op=op)
    Parameters
    • op (Op | None) –
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create(function[, commute])</td>
<td>Create a user-defined operation</td>
</tr>
<tr>
<td>Free()</td>
<td>Free the operation</td>
</tr>
<tr>
<td>Is_commutative()</td>
<td>Query reduction operations for their commutativity</td>
</tr>
<tr>
<td>Reduce_local(inbuf, inoutbuf)</td>
<td>Apply a reduction operator to local data</td>
</tr>
<tr>
<td>f2py(arg)</td>
<td></td>
</tr>
<tr>
<td>py2f()</td>
<td></td>
</tr>
</tbody>
</table>

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_commutative</td>
<td>is commutative</td>
</tr>
<tr>
<td>is_predefined</td>
<td>is a predefined operation</td>
</tr>
</tbody>
</table>

Methods Documentation

classmethod Create(function, commute=False)

Create a user-defined operation

Parameters

- function (Callable[[Buffer, Buffer, Datatype], None]) –
- commute (bool) –

Return type

Self

Free()

Free the operation

Return type

None

Is_commutative()

Query reduction operations for their commutativity

Return type

bool

Reduce_local(inbuf, inoutbuf)

Apply a reduction operator to local data

Parameters

- inbuf (BufSpec) –
- inoutbuf (BufSpec) –
Return type
None
classmethod f2py(arg)

Parameters
arg (int) –

Return type
Op

py2f()

Return type
int

Attributes Documentation

is_commutative
is commutative

is_predefined
is a predefined operation

mpi4py.MPI.Pickle

class mpi4py.MPI.Pickle
Bases: object
Pickle/unpickle Python objects

static __new__(cls, pickle=None)

Parameters
pickle (Pickle | None) –

Return type
Pickle

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dumps(obj)</td>
<td>Serialize object to pickle data stream.</td>
</tr>
<tr>
<td>dumps_oob(obj)</td>
<td>Serialize object to pickle data stream and out-of-band buffers.</td>
</tr>
<tr>
<td>loads(data)</td>
<td>Deserialize object from pickle data stream.</td>
</tr>
<tr>
<td>loads_oob(data, buffers)</td>
<td>Deserialize object from pickle data stream and out-of-band buffers.</td>
</tr>
</tbody>
</table>
Attributes Summary

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTOCOL</td>
<td>protocol version</td>
</tr>
<tr>
<td>THRESHOLD</td>
<td>out-of-band threshold</td>
</tr>
</tbody>
</table>

Methods Documentation

dumps(obj)

Serialize object to pickle data stream.

Parameters

- obj (Any) –

Return type

bytes

dumps_oob(obj)

Serialize object to pickle data stream and out-of-band buffers.

Parameters

- obj (Any) –

Return type

tuple[bytes, list[memory]]

loads(data)

Deserialize object from pickle data stream.

Parameters

- data (Buffer) –

Return type

Any

loads_oob(data, buffers)

Deserialize object from pickle data stream and out-of-band buffers.

Parameters

- data (Buffer) –
- buffers (Iterable[Buffer]) –

Return type

Any

Attributes Documentation

PROTOCOL

protocol version

THRESHOLD

out-of-band threshold
mpi4py.MPI.Prequest

class mpi4py.MPI.Prequest

Bases: Request

Persistent request handle

static __new__(cls, request=None)

Parameters
request (Prequest | None) –

Return type
Prequest

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parrived(partition)</td>
<td>Test partial completion of a partitioned receive operation</td>
</tr>
<tr>
<td>Pready(partition)</td>
<td>Mark a given partition as ready</td>
</tr>
<tr>
<td>Pready_list(partitions)</td>
<td>Mark a sequence of partitions as ready</td>
</tr>
<tr>
<td>Pready_range(partition_low, partition_high)</td>
<td>Mark a range of partitions as ready</td>
</tr>
<tr>
<td>Start()</td>
<td>Initiate a communication with a persistent request</td>
</tr>
<tr>
<td>Startall(requests)</td>
<td>Start a collection of persistent requests</td>
</tr>
</tbody>
</table>

Methods Documentation

Parrived(partition)

Test partial completion of a partitioned receive operation

Parameters
partition (int) –

Return type
bool

Pready(partition)

Mark a given partition as ready

Parameters
partition (int) –

Return type
None

Pready_list(partitions)

Mark a sequence of partitions as ready

Parameters
partitions (Sequence[int]) –

Return type
None
**Pready_range**(*partition_low*, *partition_high*)

Mark a range of partitions as ready

**Parameters**

- *partition_low* (*int*)
- *partition_high* (*int*)

**Return type**

None

**Start()**

Initiate a communication with a persistent request

**Return type**

None

**classmethod Startall(***requests***)**

Start a collection of persistent requests

**Parameters**

- *requests* (*list[Prequest]*)

**Return type**

None

```python
mpi4py.MPI.Request
class mpi4py.MPI.Request
    Bases: object
    Request handle

    static __new__(cls, request=None)
    Parameters
    - request (*Request | None*)

    Return type
    - Request
```
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel()</td>
<td>Cancel a communication request</td>
</tr>
<tr>
<td>Free()</td>
<td>Free a communication request</td>
</tr>
<tr>
<td>Get_status(status)</td>
<td>Non-destructive test for the completion of a request</td>
</tr>
<tr>
<td>Test(status)</td>
<td>Test for the completion of a send or receive</td>
</tr>
<tr>
<td>Testall(requests, statuses)</td>
<td>Test for completion of all previously initiated requests</td>
</tr>
<tr>
<td>Testany(requests, status)</td>
<td>Test for completion of any previously initiated request</td>
</tr>
<tr>
<td>Testsome(requests, statuses)</td>
<td>Test for completion of some previously initiated requests</td>
</tr>
<tr>
<td>Wait(status)</td>
<td>Wait for a send or receive to complete</td>
</tr>
<tr>
<td>Waitall(requests, statuses)</td>
<td>Wait for all previously initiated requests to complete</td>
</tr>
<tr>
<td>Waitany(requests, status)</td>
<td>Wait for any previously initiated request to complete</td>
</tr>
<tr>
<td>Waitsome(requests, statuses)</td>
<td>Wait for some previously initiated requests to complete</td>
</tr>
<tr>
<td>f2py(arg)</td>
<td>Cancel a communication request</td>
</tr>
<tr>
<td>get_status(status)</td>
<td>Non-destructive test for the completion of a request</td>
</tr>
<tr>
<td>py2f()</td>
<td>Test for the completion of a send or receive</td>
</tr>
<tr>
<td>test(status)</td>
<td>Test for the completion of a send or receive</td>
</tr>
<tr>
<td>testall(requests, statuses)</td>
<td>Test for completion of all previously initiated requests</td>
</tr>
<tr>
<td>testany(requests, status)</td>
<td>Test for completion of any previously initiated request</td>
</tr>
<tr>
<td>testsome(requests, statuses)</td>
<td>Test for completion of some previously initiated requests</td>
</tr>
<tr>
<td>wait(status)</td>
<td>Wait for a send or receive to complete</td>
</tr>
<tr>
<td>waitall(requests, statuses)</td>
<td>Wait for all previously initiated requests to complete</td>
</tr>
<tr>
<td>waitany(requests, status)</td>
<td>Wait for any previously initiated request to complete</td>
</tr>
<tr>
<td>waitsome(requests, statuses)</td>
<td>Wait for some previously initiated requests to complete</td>
</tr>
</tbody>
</table>

Methods Documentation

Cancel()
Cancel a communication request

Return type
None

Free()
Free a communication request

Return type
None

Get_status(status=None)
Non-destructive test for the completion of a request

Parameters
status (Status | None) –

Return type
bool
Test \((status=None)\)
Test for the completion of a send or receive

Parameters

\(\text{status} \ (\text{Status} \ | \ None)\) –

Return type

bool
classmethod Testall \((\text{requests, statuses=None})\)
Test for completion of all previously initiated requests

Parameters

• \(\text{requests} \ (\text{Sequence}[\text{Request}])\) –
• \(\text{statuses} \ (\text{list}[\text{Status}] \ | \ None)\) –

Return type

bool
classmethod Testany \((\text{requests, status=None})\)
Test for completion of any previously initiated request

Parameters

• \(\text{requests} \ (\text{Sequence}[\text{Request}])\) –
• \(\text{status} \ (\text{Status} \ | \ None)\) –

Return type

tuple[int, bool]
classmethod Testsome \((\text{requests, statuses=None})\)
Test for completion of some previously initiated requests

Parameters

• \(\text{requests} \ (\text{Sequence}[\text{Request}])\) –
• \(\text{statuses} \ (\text{list}[\text{Status}] \ | \ None)\) –

Return type

list[int] | None

Wait \((status=None)\)
Wait for a send or receive to complete

Parameters

\(\text{status} \ (\text{Status} \ | \ None)\) –

Return type

Literal[True]
classmethod Waitall \((\text{requests, statuses=None})\)
Wait for all previously initiated requests to complete

Parameters

• \(\text{requests} \ (\text{Sequence}[\text{Request}])\) –
• \(\text{statuses} \ (\text{list}[\text{Status}] \ | \ None)\) –

Return type

Literal[True]
classmethod Waitany(requests, status=None)
    Wait for any previously initiated request to complete
    Parameters
    • requests (Sequence[Request]) –
    • status (Status | None) –
    Return type
    int

classmethod Waitsome(requests, statuses=None)
    Wait for some previously initiated requests to complete
    Parameters
    • requests (Sequence[Request]) –
    • statuses (list[Status] | None) –
    Return type
    list[int] | None

cancel()
    Cancel a communication request
    Return type
    None

classmethod f2py(arg)
    Parameters
    arg (int) –
    Return type
    Request

get_status(status=None)
    Non-destructive test for the completion of a request
    Parameters
    status (Status | None) –
    Return type
    bool

def test(status=None)
    Test for the completion of a send or receive
    Parameters
    status (Status | None) –
    Return type
    tuple[bool, Any | None]
classmethod testall(requests, statuses=None)
    Test for completion of all previously initiated requests
    Parameters
    • requests (Sequence[Request]) –
    • statuses (list[Status] | None) –
    Return type
tuple[bool, list[dict] | None]

classmethod testany(requests, status=None)
    Test for completion of any previously initiated request
    Parameters
    • requests (Sequence[Request]) –
    • status (Status | None) –
    Return type
tuple[int, bool, dict | None]

classmethod testsome(requests, statuses=None)
    Test for completion of some previously initiated requests
    Parameters
    • requests (Sequence[Request]) –
    • statuses (list[Status] | None) –
    Return type
tuple[list[int] | None, list[dict] | None]

wait(status=None)
    Wait for a send or receive to complete
    Parameters
    status (Status | None) –
    Return type
    dict

classmethod waitall(requests, statuses=None)
    Wait for all previously initiated requests to complete
    Parameters
    • requests (Sequence[Request]) –
    • statuses (list[Status] | None) –
    Return type
    list[dict]

classmethod waitany(requests, status=None)
    Wait for any previously initiated request to complete
    Parameters
    • requests (Sequence[Request]) –
    • status (Status | None) –
classmethod waitsome(requests, statuses=None)

Wait for some previously initiated requests to complete

Parameters

- **requests** *(Sequence[Request]) –*
- **statuses** *(list[Status] | None) –*

Return type
tuple[list[int] | None, list[Any] | None]

mpi4py.MPI.Session

class mpi4py.MPI.Session

Bases: object

Session

static __new__(cls, session=None)

Parameters

- **session** *(Session | None) –*

Return type

Session

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Call_errhandler</strong> (errorcode)</td>
<td>Call the error handler installed on a session</td>
</tr>
<tr>
<td><strong>Create_errhandler</strong> (errhandler_fn)</td>
<td>Create a new error handler for sessions</td>
</tr>
<tr>
<td><strong>Create_group</strong> (pset_name)</td>
<td>Create a new group from session and process set</td>
</tr>
<tr>
<td><strong>Finalize</strong> ()</td>
<td>Finalize a session</td>
</tr>
<tr>
<td><strong>Get_errhandler</strong> ()</td>
<td>Get the error handler for a session</td>
</tr>
<tr>
<td><strong>Get_info</strong> ()</td>
<td>Return the hints for a session</td>
</tr>
<tr>
<td><strong>Get_nth_pset</strong> (n[, info])</td>
<td>Name of the nth process set</td>
</tr>
<tr>
<td><strong>Get_num_psets</strong> ([info])</td>
<td>Number of available process sets</td>
</tr>
<tr>
<td><strong>Get_pset_info</strong> (pset_name)</td>
<td>Return the hints for a session and process set</td>
</tr>
<tr>
<td><strong>Init</strong> ([info, errhandler])</td>
<td>Create a new session</td>
</tr>
<tr>
<td><strong>Set_errhandler</strong> (errhandler)</td>
<td>Set the error handler for a session</td>
</tr>
<tr>
<td><strong>f2py</strong> (arg)</td>
<td></td>
</tr>
<tr>
<td><strong>py2f</strong> ()</td>
<td></td>
</tr>
</tbody>
</table>
Methods Documentation

**Call_errhandler**(errorcode)

Call the error handler installed on a session

**Parameters**

errorcode (int) –

**Return type**

None

classmethod **Create_errhandler**(errhandler_fn)

Create a new error handler for sessions

**Parameters**

errhandler_fn (Callable[[Session, int], None]) –

**Return type**

Errhandler

**Create_group**(pset_name)

Create a new group from session and process set

**Parameters**

pset_name (str) –

**Return type**

Group

**Finalize**()

Finalize a session

**Return type**

None

**Get_errhandler**()

Get the error handler for a session

**Return type**

Errhandler

**Get_info**()

Return the hints for a session

**Return type**

Info

**Get_nth_pset**(n, info=INFO_NULL)

Name of the nth process set

**Parameters**

• n (int) –

• info (Info) –

**Return type**

str

**Get_num_psets**(info=INFO_NULL)

Number of available process sets
Parameters

   info (Info) –

Return type

   int

Get_pset_info(pset_name)

   Return the hints for a session and process set

   Parameters

      pset_name (str) –

   Return type

      Info

classmethod Init(info=INFO_NULL, errhandler=None)

   Create a new session

   Parameters

      • info (Info) –

      • errhandler (Errhandler | None) –

   Return type

      Self

Set_errhandler(errhandler)

   Set the error handler for a session

   Parameters

      errhandler (Errhandler) –

   Return type

      None

classmethod f2py(arg)

   Parameters

      arg (int) –

   Return type

      Session

   py2f()   

   Return type

      int

mpi4py.MPI.Status

class mpi4py.MPI.Status

   Bases: object

   Status object

   static __new__(cls, status=None)

   Parameters

      status (Status | None) –

   Return type

      Status
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_count([datatype])</td>
<td>Get the number of top level elements</td>
</tr>
<tr>
<td>Get_elements(datatype)</td>
<td>Get the number of basic elements in a datatype</td>
</tr>
<tr>
<td>Get_error()</td>
<td>Get message error</td>
</tr>
<tr>
<td>Get_source()</td>
<td>Get message source</td>
</tr>
<tr>
<td>Get_tag()</td>
<td>Get message tag</td>
</tr>
<tr>
<td>Is_cancelled()</td>
<td>Test to see if a request was cancelled</td>
</tr>
<tr>
<td>Set_cancelled(flag)</td>
<td>Set the cancelled state associated with a status</td>
</tr>
<tr>
<td>Set_elements(datatype, count)</td>
<td>Set the number of elements in a status</td>
</tr>
<tr>
<td>Set_error(error)</td>
<td>Set message error</td>
</tr>
<tr>
<td>Set_source(source)</td>
<td>Set message source</td>
</tr>
<tr>
<td>Set_tag(tag)</td>
<td>Set message tag</td>
</tr>
<tr>
<td>f2py(arg)</td>
<td></td>
</tr>
</tbody>
</table>

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancelled</td>
<td>cancelled state</td>
</tr>
<tr>
<td>count</td>
<td>byte count</td>
</tr>
<tr>
<td>error</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td></td>
</tr>
</tbody>
</table>

Methods Documentation

Get_count(data type=BYTE)

Get the number of top level elements

Parameters

datatype (Datatype) –

Return type

int

Get_elements(data type)

Get the number of basic elements in a datatype

Parameters

datatype (Datatype) –

Return type

int

Get_error()

Get message error
Return type
int
Get_source()
Get message source
Return type
int
Get_tag()
Get message tag
Return type
int
Is_cancelled()
Test to see if a request was cancelled
Return type
bool
Set_cancelled(flag)
Set the cancelled state associated with a status

Note: This should be used only when implementing query callback functions for generalized requests

Parameters
flag (bool) –

Return type
None

Set_elements(datatype, count)
Set the number of elements in a status

Note: This should be only used when implementing query callback functions for generalized requests

Parameters
• datatype (Datatype) –
• count (int) –

Return type
None

Set_error(error)
Set message error

Parameters
error (int) –

Return type
None
Set_source(source)
Set message source

Parameters

source (int) –

Return type

None

Set_tag(tag)
Set message tag

Parameters

tag (int) –

Return type

None

classmethod f2py(arg)

Parameters

arg (list[int]) –

Return type

Self

py2f()

Return type

list[int]

Attributes Documentation

cancelled
cancelled state
count
byte count
tag
tag
class mpi4py.MPI.Topocomm

Bases: Intracomm

Topology intracomnicator

static __new__(cls, comm=None)

Parameters

comm (Topocomm | None) –

Return type

Topocomm
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Ineighbor_allgather</code> (sendbuf, recvbuf)</td>
<td>Nonblocking Neighbor Gather to All</td>
</tr>
<tr>
<td><code>Ineighbor_allgatherv</code> (sendbuf, recvbuf)</td>
<td>Nonblocking Neighbor Gather to All Vector</td>
</tr>
<tr>
<td><code>Ineighbor_alltoall</code> (sendbuf, recvbuf)</td>
<td>Nonblocking Neighbor All-to-All</td>
</tr>
<tr>
<td><code>Ineighbor_alltoallv</code> (sendbuf, recvbuf)</td>
<td>Nonblocking Neighbor All-to-All Vector</td>
</tr>
<tr>
<td><code>Ineighbor_alltoallw</code> (sendbuf, recvbuf)</td>
<td>Nonblocking Neighbor All-to-All Generalized</td>
</tr>
<tr>
<td><code>Neighbor_allgather</code> (sendbuf, recvbuf)</td>
<td>Neighbor Gather to All</td>
</tr>
<tr>
<td><code>Neighbor_allgather_init</code> (sendbuf, recvbuf, info)</td>
<td>Persistent Neighbor Gather to All</td>
</tr>
<tr>
<td><code>Neighbor_allgatherv</code> (sendbuf, recvbuf)</td>
<td>Neighbor Gather to All Vector</td>
</tr>
<tr>
<td><code>Neighbor_allgatherv_init</code> (sendbuf, recvbuf, info)</td>
<td>Persistent Neighbor Gather to All Vector</td>
</tr>
<tr>
<td><code>Neighbor_alltoall</code> (sendbuf, recvbuf)</td>
<td>Neighbor All-to-All</td>
</tr>
<tr>
<td><code>Neighbor_alltoall_init</code> (sendbuf, recvbuf, info)</td>
<td>Persistent Neighbor All-to-All</td>
</tr>
<tr>
<td><code>Neighbor_alltoallv</code> (sendbuf, recvbuf)</td>
<td>Neighbor All-to-All Vector</td>
</tr>
<tr>
<td><code>Neighbor_alltoallv_init</code> (sendbuf, recvbuf, info)</td>
<td>Persistent Neighbor All-to-All Vector</td>
</tr>
<tr>
<td><code>Neighbor_alltoallw</code> (sendbuf, recvbuf)</td>
<td>Neighbor All-to-All Generalized</td>
</tr>
<tr>
<td><code>Neighbor_alltoallw_init</code> (sendbuf, recvbuf, info)</td>
<td>Persistent Neighbor All-to-All Generalized</td>
</tr>
<tr>
<td><code>neighbor_allgather</code> (sendobj)</td>
<td>Neighbor Gather to All</td>
</tr>
<tr>
<td><code>neighbor_alltoall</code> (sendobj)</td>
<td>Neighbor All to All Scatter/Gather</td>
</tr>
</tbody>
</table>

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>degrees</code></td>
<td>number of incoming and outgoing neighbors</td>
</tr>
<tr>
<td><code>indegree</code></td>
<td>number of incoming neighbors</td>
</tr>
<tr>
<td><code>inedges</code></td>
<td>incoming neighbors</td>
</tr>
<tr>
<td><code>inoutedges</code></td>
<td>incoming and outgoing neighbors</td>
</tr>
<tr>
<td><code>outdegree</code></td>
<td>number of outgoing neighbors</td>
</tr>
<tr>
<td><code>outedges</code></td>
<td>outgoing neighbors</td>
</tr>
</tbody>
</table>

Methods Documentation

**Ineighbor_allgather** *(sendbuf, recvbuf)*
Nonblocking Neighbor Gather to All

**Parameters**
- `sendbuf` *(BufSpec)* –
- `recvbuf` *(BufSpecB)* –

**Return type**
Request

**Ineighbor_allgatherv** *(sendbuf, recvbuf)*
Nonblocking Neighbor Gather to All Vector

**Parameters**
• \texttt{sendbuf} (\texttt{BufSpec}) –
• \texttt{recvbuf} (\texttt{BufSpecV}) –

\textbf{Return type}
\texttt{Request}

\textbf{Ineighbor\_alltoall}(\texttt{sendbuf, recvbuf})
Nonblocking Neighbor All-to-All

\textbf{Parameters}
• \texttt{sendbuf} (\texttt{BufSpecB}) –
• \texttt{recvbuf} (\texttt{BufSpecB}) –

\textbf{Return type}
\texttt{Request}

\textbf{Ineighbor\_alltoallv}(\texttt{sendbuf, recvbuf})
Nonblocking Neighbor All-to-All Vector

\textbf{Parameters}
• \texttt{sendbuf} (\texttt{BufSpecV}) –
• \texttt{recvbuf} (\texttt{BufSpecV}) –

\textbf{Return type}
\texttt{Request}

\textbf{Ineighbor\_alltoallw}(\texttt{sendbuf, recvbuf})
Nonblocking Neighbor All-to-All Generalized

\textbf{Parameters}
• \texttt{sendbuf} (\texttt{BufSpecW}) –
• \texttt{recvbuf} (\texttt{BufSpecW}) –

\textbf{Return type}
\texttt{Request}

\textbf{Neighbor\_allgather}(\texttt{sendbuf, recvbuf})
Neighbor Gather to All

\textbf{Parameters}
• \texttt{sendbuf} (\texttt{BufSpec}) –
• \texttt{recvbuf} (\texttt{BufSpecB}) –

\textbf{Return type}
\texttt{None}

\textbf{Neighbor\_allgather\_init}(\texttt{sendbuf, recvbuf, info=INFO\_NULL})
Persistent Neighbor Gather to All

\textbf{Parameters}
• \texttt{sendbuf} (\texttt{BufSpec}) –
• \texttt{recvbuf} (\texttt{BufSpecB}) –
• \texttt{info} (\texttt{Info}) –
Return type
Prequest

**Neighbor_allgatherv**(*sendbuf, recvbuf*)
Neighbor Gather to All Vector

**Parameters**
- **sendbuf** (*BufSpec*) –
- **recvbuf** (*BufSpecV*) –

Return type
None

**Neighbor_allgatherv_init**(*sendbuf, recvbuf, info=INFO_NULL*)
Persistent Neighbor Gather to All Vector

**Parameters**
- **sendbuf** (*BufSpec*) –
- **recvbuf** (*BufSpecV*) –
- **info** (*Info*) –

Return type
Prequest

**Neighbor_alltoall**(*sendbuf, recvbuf*)
Neighbor All-to-All

**Parameters**
- **sendbuf** (*BufSpecB*) –
- **recvbuf** (*BufSpecB*) –

Return type
None

**Neighbor_alltoall_init**(*sendbuf, recvbuf, info=INFO_NULL*)
Persistent Neighbor All-to-All

**Parameters**
- **sendbuf** (*BufSpecB*) –
- **recvbuf** (*BufSpecB*) –
- **info** (*Info*) –

Return type
Prequest

**Neighbor_alltoallv**(*sendbuf, recvbuf*)
Neighbor All-to-All Vector

**Parameters**
- **sendbuf** (*BufSpecV*) –
- **recvbuf** (*BufSpecV*) –

Return type
None
**Neighbor_alltoally_init**(sendbuf, recvbuf, info=INFO_NULL)
Persistent Neighbor All-to-All Vector

**Parameters**
- **sendbuf** (BufSpecV) –
- **recvbuf** (BufSpecV) –
- **info** (Info) –

**Return type**
Prequest

**Neighbor_alltoallw**(sendbuf, recvbuf)
Neighbor All-to-All Generalized

**Parameters**
- **sendbuf** (BufSpecW) –
- **recvbuf** (BufSpecW) –

**Return type**
None

**Neighbor_alltoallw_init**(sendbuf, recvbuf, info=INFO_NULL)
Persistent Neighbor All-to-All Generalized

**Parameters**
- **sendbuf** (BufSpecW) –
- **recvbuf** (BufSpecW) –
- **info** (Info) –

**Return type**
Prequest

**neighbor_allgather**(sendobj)
Neighbor Gather to All

**Parameters**
- **sendobj** (Any) –

**Return type**
list[Any]

**neighbor_alltoall**(sendobj)
Neighbor All to All Scatter/Gather

**Parameters**
- **sendobj** (list[Any]) –

**Return type**
list[Any]
**Attributes Documentation**

**degrees**  
number of incoming and outgoing neighbors

**indegree**  
number of incoming neighbors

**inedges**  
incoming neighbors

**inoutedges**  
incoming and outgoing neighbors

**outdegree**  
number of outgoing neighbors

**outedges**  
outgoing neighbors

---

**mpi4py.MPI.Win**

**class** mpi4py.MPI.Win  
**Bases:** object

Window handle

**static __new__(cls, win=None)**

**Parameters**

- **win** *(Win | None)*

**Return type**

Win

---

**Methods Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Accumulate</code></td>
<td>Accumulate data into the target process</td>
</tr>
<tr>
<td><code>Allocate</code></td>
<td>Create an window object for one-sided communication</td>
</tr>
<tr>
<td><code>Allocate_shared</code></td>
<td>Create an window object for one-sided communication</td>
</tr>
<tr>
<td><code>Attach</code></td>
<td>Attach a local memory region</td>
</tr>
<tr>
<td><code>Call_errhandler</code></td>
<td>Call the error handler installed on a window</td>
</tr>
<tr>
<td><code>Compare_and_swap</code></td>
<td>Perform one-sided atomic compare-and-swap</td>
</tr>
<tr>
<td><code>Complete</code></td>
<td>Completes an RMA operations begun after an Win.Start()</td>
</tr>
<tr>
<td><code>Create</code></td>
<td>Create an window object for one-sided communication</td>
</tr>
<tr>
<td><code>Create_dynamic</code></td>
<td>Create an window object for one-sided communication</td>
</tr>
<tr>
<td><code>Create_errhandler</code></td>
<td>Create a new error handler for windows</td>
</tr>
<tr>
<td><code>Create_keyval</code></td>
<td>Create a new attribute key for windows</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete_attr</strong> (keyval)</td>
<td>Delete attribute value associated with a key</td>
</tr>
<tr>
<td><strong>Detach</strong> (memory)</td>
<td>Detach a local memory region</td>
</tr>
<tr>
<td><strong>Fence</strong> ([assertion])</td>
<td>Perform an MPI fence synchronization on a window</td>
</tr>
<tr>
<td><strong>Fetch_and_op</strong> (origin, result, target_rank[, ...])</td>
<td>Perform one-sided read-modify-write</td>
</tr>
<tr>
<td><strong>Flush</strong> (rank)</td>
<td>Complete all outstanding RMA operations at the given target</td>
</tr>
<tr>
<td><strong>Flush_all</strong> ()</td>
<td>Complete all outstanding RMA operations at all targets</td>
</tr>
<tr>
<td><strong>Flush_local</strong> (rank)</td>
<td>Complete locally all outstanding RMA operations at the given target</td>
</tr>
<tr>
<td><strong>Flush_local_all</strong> ()</td>
<td>Complete locally all outstanding RMA operations at all targets</td>
</tr>
<tr>
<td><strong>Free</strong> ()</td>
<td>Free a window</td>
</tr>
<tr>
<td><strong>Free_keyval</strong> (keyval)</td>
<td>Free an attribute key for windows</td>
</tr>
<tr>
<td><strong>Get</strong> (origin, target_rank[, target])</td>
<td>Get data from a memory window on a remote process.</td>
</tr>
<tr>
<td><strong>Get_accumulate</strong> (origin, result, target_rank)</td>
<td>Fetch-and-accumulate data into the target process.</td>
</tr>
<tr>
<td><strong>Get_attr</strong> (origin, result, target_rank)</td>
<td>Retrieve attribute value by key</td>
</tr>
<tr>
<td><strong>Get_errhandler</strong> ()</td>
<td>Get the error handler for a window</td>
</tr>
<tr>
<td><strong>Get_group</strong> ()</td>
<td>Return a duplicate of the group of the communicator used to create the window</td>
</tr>
<tr>
<td><strong>Get_info</strong> ()</td>
<td>Return the hints for a windows that are currently in use</td>
</tr>
<tr>
<td><strong>Get_name</strong> ()</td>
<td>Get the print name associated with the window</td>
</tr>
<tr>
<td><strong>Lock</strong> (rank[, lock_type, assertion])</td>
<td>Begin an RMA access epoch at the target process</td>
</tr>
<tr>
<td><strong>Lock_all</strong> ([assertion])</td>
<td>Begin an RMA access epoch at all processes</td>
</tr>
<tr>
<td><strong>Post</strong> (group[, assertion])</td>
<td>Start an RMA exposure epoch</td>
</tr>
<tr>
<td><strong>Put</strong> (origin, target_rank[, target])</td>
<td>Put data into a memory window on a remote process.</td>
</tr>
<tr>
<td><strong>Raccumulate</strong> (origin, target_rank[, target, op])</td>
<td>Fetch-and-accumulate data into the target process.</td>
</tr>
<tr>
<td><strong>Rget</strong> (origin, target_rank[, target])</td>
<td>Get data from a memory window on a remote process.</td>
</tr>
<tr>
<td><strong>Rget_accumulate</strong> (origin, result, target_rank)</td>
<td>Accumulate data into the target process using remote memory access.</td>
</tr>
<tr>
<td><strong>Rput</strong> (origin, target_rank[, target])</td>
<td>Put data into a memory window on a remote process.</td>
</tr>
<tr>
<td><strong>Set_attr</strong> (keyval, attrval)</td>
<td>Store attribute value associated with a key</td>
</tr>
<tr>
<td><strong>Set_errhandler</strong> (errhandler)</td>
<td>Set the error handler for a window</td>
</tr>
<tr>
<td><strong>Set_info</strong> (info)</td>
<td>Set new values for the hints associated with a window</td>
</tr>
<tr>
<td><strong>Set_name</strong> (name)</td>
<td>Set the print name associated with the window</td>
</tr>
<tr>
<td><strong>Shared_query</strong> (rank)</td>
<td>Query the process-local address for remote memory segments created with Win.Allocate_shared()</td>
</tr>
<tr>
<td><strong>Start</strong> (group[, assertion])</td>
<td>Start an RMA access epoch for MPI</td>
</tr>
<tr>
<td><strong>Sync</strong> ()</td>
<td>Synchronize public and private copies of the given window</td>
</tr>
<tr>
<td><strong>Test</strong> ()</td>
<td>Test whether an RMA exposure epoch has completed</td>
</tr>
<tr>
<td><strong>Unlock</strong> (rank)</td>
<td>Complete an RMA access epoch at the target process</td>
</tr>
<tr>
<td><strong>Unlock_all</strong> ()</td>
<td>Complete an RMA access epoch at all processes</td>
</tr>
<tr>
<td><strong>Wait</strong> ()</td>
<td>Complete an RMA exposure epoch begun with Win. Post()</td>
</tr>
<tr>
<td><strong>f2py</strong> (arg)</td>
<td></td>
</tr>
<tr>
<td><strong>py2f</strong> ()</td>
<td></td>
</tr>
<tr>
<td><strong>tomemory</strong> ()</td>
<td>Return window memory buffer</td>
</tr>
</tbody>
</table>
Attributes Summary

<table>
<thead>
<tr>
<th>attr</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrs</td>
<td>window attributes</td>
</tr>
<tr>
<td>flavor</td>
<td>window create flavor</td>
</tr>
<tr>
<td>group</td>
<td>window group</td>
</tr>
<tr>
<td>info</td>
<td>window info</td>
</tr>
<tr>
<td>model</td>
<td>window memory model</td>
</tr>
<tr>
<td>name</td>
<td>window name</td>
</tr>
</tbody>
</table>

Methods Documentation

**Accumulate**(origin, target_rank, target=None, op=SUM)
Accumulate data into the target process

- **Parameters**
  - **origin** (BufSpec) –
  - **target_rank** (int) –
  - **target** (TargetSpec | None) –
  - **op** (Op) –

- **Return type**
  None

**classmethod Allocate**(size, disp_unit=1, info=INFO_NULL, comm=COMM_SELF)
Create an window object for one-sided communication

- **Parameters**
  - **size** (int) –
  - **disp_unit** (int) –
  - **info** (Info) –
  - **comm** (Intracomm) –

- **Return type**
  Self

**classmethod Allocate_shared**(size, disp_unit=1, info=INFO_NULL, comm=COMM_SELF)
Create an window object for one-sided communication

- **Parameters**
  - **size** (int) –
  - **disp_unit** (int) –
  - **info** (Info) –
  - **comm** (Intracomm) –

- **Return type**
  Self
Attach

Attach a local memory region

Parameters
  memory (Buffer) –

Return type
  None

Call_errhandler

Call the error handler installed on a window

Parameters
  errorcode (int) –

Return type
  None

Compare_and_swap

Perform one-sided atomic compare-and-swap

Parameters
  • origin (BufSpec) –
  • compare (BufSpec) –
  • result (BufSpec) –
  • target_rank (int) –
  • target_disp (int) –

Return type
  None

Complete

Completes an RMA operations begun after an Win.Start()

Return type
  None

classmethod Create

Create an window object for one-sided communication

Parameters
  • memory (Buffer | Bottom) –
  • disp_unit (int) –
  • info (Info) –
  • comm (Intracomm) –

Return type
  Self

classmethod Create_dynamic

Create an window object for one-sided communication

Parameters
  • info (Info) –
  • comm (Intracomm) –
Return type

_self_

classmethod Create_errhandler(errhandler_fn)
Create a new error handler for windows

Parameters

errhandler_fn (Callable[[Win, int], None]) –

Return type

Errhandler

classmethod Create_keyval(copy_fn=None, delete_fn=None, nopython=False)
Create a new attribute key for windows

Parameters

• copy_fn (Callable[[Win, int, Any], Any] | None) –
• delete_fn (Callable[[Win, int, Any], None] | None) –
• nopython (bool) –

Return type

int

Delete_attr(keyval)
Delete attribute value associated with a key

Parameters

keyval (int) –

Return type

None

Detach(memory)
Detach a local memory region

Parameters

memory (Buffer) –

Return type

None

Fence(assertion=0)
Perform an MPI fence synchronization on a window

Parameters

assertion (int) –

Return type

None

Fetch_and_op(origin, result, target_rank, target_disp=0, op=SUM)
Perform one-sided read-modify-write

Parameters

• origin (BufSpec) –
• result (BufSpec) –
• target_rank (int) –
• target_disp (int) –
• **op (Op)** –
  
  **Return type**
  
  None

**Flush(rank)**

Complete all outstanding RMA operations at the given target

**Parameters**

  rank (int) –

**Return type**

None

**Flush_all()**

Complete all outstanding RMA operations at all targets

**Return type**

None

**Flush_local(rank)**

Complete locally all outstanding RMA operations at the given target

**Parameters**

  rank (int) –

**Return type**

None

**Flush_local_all()**

Complete locally all outstanding RMA operations at all targets

**Return type**

None

**Free()**

Free a window

**Return type**

None

**classmethod Free_keyval(keyval)**

Free an attribute key for windows

**Parameters**

  keyval (int) –

**Return type**

int

**Get(origin, target_rank, target=None)**

Get data from a memory window on a remote process.

**Parameters**

  • origin (BufSpec) –
  
  • target_rank (int) –
  
  • target (TargetSpec | None) –

**Return type**

None
Get_accumulate(origin, result, target_rank, target=None, op=SUM)

Fetch-and-accumulate data into the target process

Parameters

- origin (BufSpec) –
- result (BufSpec) –
- target_rank (int) –
- target (TargetSpec | None) –
- op (Op) –

Return type

None

Get_attr(keyval)

Retrieve attribute value by key

Parameters

keyval (int) –

Return type

int | Any | None

Get_errhandler()

Get the error handler for a window

Return type

Errhandler

Get_group()

Return a duplicate of the group of the communicator used to create the window

Return type

Group

Get_info()

Return the hints for a windows that are currently in use

Return type

Info

Get_name()

Get the print name associated with the window

Return type

str

Lock(rank, lock_type=LOCK_EXCLUSIVE, assertion=0)

Begin an RMA access epoch at the target process

Parameters

- rank (int) –
- lock_type (int) –
- assertion (int) –

Return type

None
Lock_all(assertion=0)
Begin an RMA access epoch at all processes

Parameters
assertion (int) –

Return type
None

Post(group, assertion=0)
Start an RMA exposure epoch

Parameters
• group (Group) –
• assertion (int) –

Return type
None

Put(origin, target_rank, target=None)
Put data into a memory window on a remote process.

Parameters
• origin (BufSpec) –
• target_rank (int) –
• target (TargetSpec | None) –

Return type
None

Raccumulate(origin, target_rank, target=None, op=SUM)
Fetch-and-accumulate data into the target process

Parameters
• origin (BufSpec) –
• target_rank (int) –
• target (TargetSpec | None) –
• op (Op) –

Return type
Request

Rget(origin, target_rank, target=None)
Get data from a memory window on a remote process.

Parameters
• origin (BufSpec) –
• target_rank (int) –
• target (TargetSpec | None) –

Return type
Request
\textbf{Rget\_accumulate}(\textit{origin, result, target\_rank, target=None, op=SUM})

Accumulate data into the target process using remote memory access.

\textbf{Parameters}

- \textit{origin} (BufSpec) –
- \textit{result} (BufSpec) –
- \textit{target\_rank} (int) –
- \textit{target} (TargetSpec | None) –
- \textit{op} (Op) –

\textbf{Return type}

\textit{Request}

\textbf{Rput}(\textit{origin, target\_rank, target=None})

Put data into a memory window on a remote process.

\textbf{Parameters}

- \textit{origin} (BufSpec) –
- \textit{target\_rank} (int) –
- \textit{target} (TargetSpec | None) –

\textbf{Return type}

\textit{Request}

\textbf{Set\_attr}(\textit{keyval, attrval})

Store attribute value associated with a key

\textbf{Parameters}

- \textit{keyval} (int) –
- \textit{attrval} (Any) –

\textbf{Return type}

\textit{None}

\textbf{Set\_errhandler}(\textit{errhandler})

Set the error handler for a window

\textbf{Parameters}

- \textit{errhandler} (Errhandler) –

\textbf{Return type}

\textit{None}

\textbf{Set\_info}(\textit{info})

Set new values for the hints associated with a window

\textbf{Parameters}

- \textit{info} (Info) –

\textbf{Return type}

\textit{None}
Set\_name\((name)\)
Set the print name associated with the window

Parameters
name \((str)\) –

Return type
None

Shared\_query\((rank)\)
Query the process-local address for remote memory segments created with \texttt{Win.Allocate\_shared()}.

Parameters
rank \((int)\) –

Return type
tuple[memory, int]

Start\((group, assertion=0)\)
Start an RMA access epoch for MPI

Parameters
• group \((\text{Group})\) –
• assertion \((int)\) –

Return type
None

Sync()
Synchronize public and private copies of the given window

Return type
None

Test()
Test whether an RMA exposure epoch has completed

Return type
bool

Unlock\((rank)\)
Complete an RMA access epoch at the target process

Parameters
rank \((int)\) –

Return type
None

Unlock\_all()
Complete an RMA access epoch at all processes

Return type
None

Wait()
Complete an RMA exposure epoch begun with \texttt{Win.Post()}

Return type
\texttt{Literal[True]}
classmethod f2py(arg)

Parameters
arg (int) –

Return type
Win

def py2f():

Return type
int
tomemory():

Return window memory buffer

Return type
memory

Attributes Documentation

attrs
window attributes

flavor
window create flavor

group
window group

info
window info

model
window memory model

name
window name

mpi4py.MPI.memory

class mpi4py.MPI.memory
Bases: object
Memory buffer

static __new__(cls, buf):

Parameters
buf (Buffer) –

Return type
memory
Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allocate(nbytes[, clear])</code></td>
<td>Memory allocation</td>
</tr>
<tr>
<td><code>fromaddress(address, nbytes[, readonly])</code></td>
<td>Memory from address and size in bytes</td>
</tr>
<tr>
<td><code>frombuffer(obj[, readonly])</code></td>
<td>Memory from buffer-like object</td>
</tr>
<tr>
<td><code>release()</code></td>
<td>Release the underlying buffer exposed by the memory object</td>
</tr>
<tr>
<td><code>tobytes([order])</code></td>
<td>Return the data in the buffer as a byte string</td>
</tr>
<tr>
<td><code>toreadonly()</code></td>
<td>Return a readonly version of the memory object</td>
</tr>
</tbody>
</table>

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>address</code></td>
<td>Memory address</td>
</tr>
<tr>
<td><code>format</code></td>
<td>A string with the format of each element</td>
</tr>
<tr>
<td><code>itemsize</code></td>
<td>The size in bytes of each element</td>
</tr>
<tr>
<td><code>nbytes</code></td>
<td>Memory size (in bytes)</td>
</tr>
<tr>
<td><code>obj</code></td>
<td>The underlying object of the memory</td>
</tr>
<tr>
<td><code>readonly</code></td>
<td>Boolean indicating whether the memory is read-only</td>
</tr>
</tbody>
</table>

Methods Documentation

**static allocate(nbytes, clear=False)**

Memory allocation

Parameters

- `nbytes (int)` –
- `clear (bool)` –

Return type

memory

**static fromaddress(address, nbytes, readonly=False)**

Memory from address and size in bytes

Parameters

- `address (int)` –
- `nbytes (int)` –
- `readonly (bool)` –

Return type

memory

**static frombuffer(obj, readonly=False)**

Memory from buffer-like object

Parameters

- `obj (Buffer)` –
- `readonly (bool)` –
Return type
memory
release()
Release the underlying buffer exposed by the memory object

Return type
None
tobytes(order=None)
Return the data in the buffer as a byte string

Parameters
order (str | None) –

Return type
bytes
toreadonly()
Return a readonly version of the memory object

Return type
memory

Attributes Documentation

address
Memory address

format
A string with the format of each element

itemsize
The size in bytes of each element

 nbytes
Memory size (in bytes)

 obj
The underlying object of the memory

 readonly
Boolean indicating whether the memory is read-only

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Exception class</th>
</tr>
</thead>
</table>

165
Exception class

static __new__(cls, ierr=SUCCESS)

Parameters
ierr (int) –

Return type
Exception

Methods Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get_error_class()</td>
<td>Error class</td>
</tr>
<tr>
<td>Get_error_code()</td>
<td>Error code</td>
</tr>
<tr>
<td>Get_error_string()</td>
<td>Error string</td>
</tr>
</tbody>
</table>

Attributes Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error_class</td>
<td>error class</td>
</tr>
<tr>
<td>error_code</td>
<td>error code</td>
</tr>
<tr>
<td>error_string</td>
<td>error string</td>
</tr>
</tbody>
</table>

Methods Documentation

Get_error_class()

   Error class

   Return type
int

Get_error_code()

   Error code

   Return type
int

Get_error_string()

   Error string

   Return type
str
Attributes Documentation

**error_class**
- error class

**error_code**
- error code

**error_string**
- error string

## Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add_error_class()</strong></td>
<td>Add an error class to the known error classes</td>
</tr>
<tr>
<td><strong>Add_error_code(errorclass)</strong></td>
<td>Add an error code to an error class</td>
</tr>
<tr>
<td><strong>Add_error_string(errorcode, string)</strong></td>
<td>Associate an error string with an error class or error code</td>
</tr>
<tr>
<td><strong>Aint_add(base, disp)</strong></td>
<td>Return the sum of base address and displacement</td>
</tr>
<tr>
<td><strong>Aint_diff(addr1, addr2)</strong></td>
<td>Return the difference between absolute addresses</td>
</tr>
<tr>
<td><strong>Alloc_mem(size[, info])</strong></td>
<td>Allocate memory for message passing and RMA</td>
</tr>
<tr>
<td><strong>Attach_buffer(buf)</strong></td>
<td>Attach a user-provided buffer for sending in buffered mode</td>
</tr>
<tr>
<td><strong>Close_port(port_name)</strong></td>
<td>Close a port</td>
</tr>
<tr>
<td><strong>Compute_dims(nnodes, dims)</strong></td>
<td>Return a balanced distribution of processes per coordinate direction</td>
</tr>
<tr>
<td><strong>Detach_buffer()</strong></td>
<td>Remove an existing attached buffer</td>
</tr>
<tr>
<td><strong>Finalize()</strong></td>
<td>Terminate the MPI execution environment</td>
</tr>
<tr>
<td><strong>Free_mem(mem)</strong></td>
<td>Free memory allocated with <strong>Alloc_mem()</strong></td>
</tr>
<tr>
<td><strong>Get_address(location)</strong></td>
<td>Get the address of a location in memory</td>
</tr>
<tr>
<td><strong>Get_error_class(errorcode)</strong></td>
<td>Convert an error code into an error class</td>
</tr>
<tr>
<td><strong>Get_error_string(errorcode)</strong></td>
<td>Return the error string for a given error class or error code</td>
</tr>
<tr>
<td><strong>Get_library_version()</strong></td>
<td>Obtain the version string of the MPI library</td>
</tr>
<tr>
<td><strong>Get_processor_name()</strong></td>
<td>Obtain the name of the calling processor</td>
</tr>
<tr>
<td><strong>Get_version()</strong></td>
<td>Obtain the version number of the MPI standard supported by the implementation as a tuple (version, subversion)</td>
</tr>
<tr>
<td><strong>Init()</strong></td>
<td>Initialize the MPI execution environment</td>
</tr>
<tr>
<td><strong>Init_thread([required])</strong></td>
<td>Initialize the MPI execution environment</td>
</tr>
<tr>
<td><strong>Is_finalized()</strong></td>
<td>Indicates whether <strong>Finalize</strong> has completed</td>
</tr>
<tr>
<td><strong>Is_initialized()</strong></td>
<td>Indicates whether <strong>Init</strong> has been called</td>
</tr>
<tr>
<td><strong>Is_thread_main()</strong></td>
<td>Indicate whether this thread called <strong>Init</strong> or <strong>Init_thread</strong></td>
</tr>
<tr>
<td><strong>Lookup_name(service_name[, info])</strong></td>
<td>Lookup a port name given a service name</td>
</tr>
<tr>
<td><strong>Open_port([info])</strong></td>
<td>Return an address that can be used to establish connections between groups of MPI processes</td>
</tr>
<tr>
<td><strong>Pcontrol(level)</strong></td>
<td>Control profiling</td>
</tr>
<tr>
<td><strong>Publish_name(service_name, port_name[, info])</strong></td>
<td>Publish a service name</td>
</tr>
<tr>
<td><strong>Query_thread()</strong></td>
<td>Return the level of thread support provided by the MPI library</td>
</tr>
<tr>
<td><strong>Register_datarep(datarep, read_fn, write_fn, ...)</strong></td>
<td>Register user-defined data representations</td>
</tr>
</tbody>
</table>
Table 6 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Unpublish_name(service_name, port_name[, info])</code></td>
<td>Unpublish a service name</td>
</tr>
<tr>
<td><code>Wtick()</code></td>
<td>Return the resolution of <code>Wtime</code></td>
</tr>
<tr>
<td><code>Wtime()</code></td>
<td>Return an elapsed time on the calling processor</td>
</tr>
<tr>
<td><code>get_vendor()</code></td>
<td>Information about the underlying MPI implementation</td>
</tr>
</tbody>
</table>

---

**mpi4py.MPI.Add_error_class**

`mpi4py.MPI.Add_error_class()`

Add an *error class* to the known error classes

**Return type**

`int`

**mpi4py.MPI.Add_error_code**

`mpi4py.MPI.Add_error_code(errorclass)`

Add an *error code* to an *error class*

**Parameters**

- `errorclass (int)`

**Return type**

`int`

**mpi4py.MPI.Add_error_string**

`mpi4py.MPI.Add_error_string(errorcode, string)`

Associate an *error string* with an *error class* or *errorcode*

**Parameters**

- `errorcode (int)`
- `string (str)`

**Return type**

`None`

**mpi4py.MPI.Aint_add**

`mpi4py.MPI.Aint_add(base, disp)`

Return the sum of base address and displacement

**Parameters**

- `base (int)`
- `disp (int)`

**Return type**

`int`
**mpi4py.MPI.Aint_diff**

```python
mpi4py.MPI.Aint_diff(addr1, addr2)
```

Return the difference between absolute addresses

**Parameters**

- `addr1 (int)`
- `addr2 (int)`

**Return type**

`int`

---

**mpi4py.MPI.Alloc_mem**

```python
mpi4py.MPI.Alloc_mem(size, info=INFO_NULL)
```

Allocate memory for message passing and RMA

**Parameters**

- `size (int)`
- `info (Info)`

**Return type**

`memory`

---

**mpi4py.MPI.Attach_buffer**

```python
mpi4py.MPI.Attach_buffer(buf)
```

Attach a user-provided buffer for sending in buffered mode

**Parameters**

- `buf (Buffer)`

**Return type**

`None`

---

**mpi4py.MPI.Close_port**

```python
mpi4py.MPI.Close_port(port_name)
```

Close a port

**Parameters**

- `port_name (str)`

**Return type**

`None`
mpipy.MPI.Compute_dims

mpipy.MPI.Compute_dims(nnodes, dims)
   Return a balanced distribution of processes per coordinate direction

   Parameters
   • nnodes (int) –
   • dims (int | Sequence[int]) –

   Return type
   list[int]

mpipy.MPI.Detach_buffer

mpipy.MPI.Detach_buffer()
   Remove an existing attached buffer

   Return type
   Buffer

mpipy.MPI.Finalize

mpipy.MPI.Finalize()
   Terminate the MPI execution environment

   Return type
   None

mpipy.MPI.Free_mem

mpipy.MPI.Free_mem(mem)
   Free memory allocated with Alloc_mem()

   Parameters
   • mem (memory) –

   Return type
   None

mpipy.MPI.Get_address

mpipy.MPI.Get_address(location)
   Get the address of a location in memory

   Parameters
   • location (Buffer | Bottom) –

   Return type
   int
mpi4py.MPI.Get_error_class

mpi4py.MPI.Get_error_class(errorcode)
Convert an error code into an error class

Parameters
  errorcode (int) –

Return type
  int

mpi4py.MPI.Get_error_string

mpi4py.MPI.Get_error_string(errorcode)
Return the error string for a given error class or error code

Parameters
  errorcode (int) –

Return type
  str

mpi4py.MPI.Get_library_version

mpi4py.MPI.Get_library_version()
Obtain the version string of the MPI library

Return type
  str

mpi4py.MPI.Get_processor_name

mpi4py.MPI.Get_processor_name()
Obtain the name of the calling processor

Return type
  str

mpi4py.MPI.Get_version

mpi4py.MPI.Get_version()
Obtain the version number of the MPI standard supported by the implementation as a tuple (version, subversion)

Return type
  tuple[int, int]
mpi4py.MPI.Init

mpi4py.MPI.Init()
Initialize the MPI execution environment

    Return type
    None

mpi4py.MPI.Init_thread

mpi4py.MPI.Init_thread(required=THREAD_MULTIPLE)
Initialize the MPI execution environment

    Parameters
    required (int) --

    Return type
    int

mpi4py.MPI.Is_finalized

mpi4py.MPI.Is_finalized()
Indicates whether Finalize has completed

    Return type
    bool

mpi4py.MPI.Is_initialized

mpi4py.MPI.Is_initialized()
Indicates whether Init has been called

    Return type
    bool

mpi4py.MPI.Is_thread_main

mpi4py.MPI.Is_thread_main()
Indicate whether this thread called Init or Init_thread

    Return type
    bool

mpi4py.MPI.Lookup_name

mpi4py.MPI.Lookup_name(service_name, info=INFO_NULL)
Lookup a port name given a service name

    Parameters
    • service_name (str) --
    • info (Info) --
mpi4py.MPI.Open_port

mpi4py.MPI.Open_port(info=INFO_NULL)

Return an address that can be used to establish connections between groups of MPI processes

Parameters
info (Info) –

Return type
str

mpi4py.MPI.Pcontrol

mpi4py.MPI.Pcontrol(level)

Control profiling

Parameters
level (int) –

Return type
None

mpi4py.MPI.Publish_name

mpi4py.MPI.Publish_name(service_name, port_name, info=INFO_NULL)

Publish a service name

Parameters
• service_name (str) –
• port_name (str) –
• info (Info) –

Return type
None

mpi4py.MPI.Query_thread

mpi4py.MPI.Query_thread()

Return the level of thread support provided by the MPI library

Return type
int
mpi4py.MPI.Register_datarep

mpi4py.MPI.Register_datarep(datarep, read_fn, write_fn, extent_fn)

Register user-defined data representations

Parameters

- **datarep (str)**
- **read_fn (Callable[[Buffer, Datatype, int, Buffer, int], None])**
- **write_fn (Callable[[Buffer, Datatype, int, Buffer, int], None])**
- **extent_fn (Callable[[Datatype], int])**

Return type

None

mpi4py.MPI.Unpublish_name

mpi4py.MPI.Unpublish_name(service_name, port_name, info=INFO_NULL)

Unpublish a service name

Parameters

- **service_name (str)**
- **port_name (str)**
- **info (Info)**

Return type

None

mpi4py.MPI.Wtick

mpi4py.MPI.Wtick()

Return the resolution of Wtime

Return type

float

mpi4py.MPI.Wtime

mpi4py.MPI.Wtime()

Return an elapsed time on the calling processor

Return type

float
mpi4py.MPI.get_vendor

mpi4py.MPI.get_vendor()

Information about the underlying MPI implementation

Returns

- a string with the name of the MPI implementation
- an integer 3-tuple version (major, minor, micro)

Return type
tuple[str, tuple[int, int, int]]

Attributes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDEFINED</td>
<td>Constant UNDEFINED of type int</td>
</tr>
<tr>
<td>ANY_SOURCE</td>
<td>Constant ANY_SOURCE of type int</td>
</tr>
<tr>
<td>ANY_TAG</td>
<td>Constant ANY_TAG of type int</td>
</tr>
<tr>
<td>PROC_NULL</td>
<td>Constant PROC_NULL of type int</td>
</tr>
<tr>
<td>ROOT</td>
<td>Constant ROOT of type int</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>Constant BOTTOM of type BottomType</td>
</tr>
<tr>
<td>IN_PLACE</td>
<td>Constant IN_PLACE of type InPlaceType</td>
</tr>
<tr>
<td>KEYVAL_INVALID</td>
<td>Constant KEYVAL_INVALID of type int</td>
</tr>
<tr>
<td>TAG_UB</td>
<td>Constant TAG_UB of type int</td>
</tr>
<tr>
<td>HOST</td>
<td>Constant HOST of type int</td>
</tr>
<tr>
<td>IO</td>
<td>Constant IO of type int</td>
</tr>
<tr>
<td>WTIME_IS_GLOBAL</td>
<td>Constant WTIME_IS_GLOBAL of type int</td>
</tr>
<tr>
<td>UNIVERSE_SIZE</td>
<td>Constant UNIVERSE_SIZE of type int</td>
</tr>
<tr>
<td>APPNUM</td>
<td>Constant APPNUM of type int</td>
</tr>
<tr>
<td>LASTUSEDCODE</td>
<td>Constant LASTUSEDCODE of type int</td>
</tr>
<tr>
<td>WIN_BASE</td>
<td>Constant WIN_BASE of type int</td>
</tr>
<tr>
<td>WIN_SIZE</td>
<td>Constant WIN_SIZE of type int</td>
</tr>
<tr>
<td>WIN_DISP_UNIT</td>
<td>Constant WIN_DISP_UNIT of type int</td>
</tr>
<tr>
<td>WIN_CREATE_FLAVOR</td>
<td>Constant WIN_CREATE_FLAVOR of type int</td>
</tr>
<tr>
<td>WIN_FLAVOR</td>
<td>Constant WIN_FLAVOR of type int</td>
</tr>
<tr>
<td>WIN_MODEL</td>
<td>Constant WIN_MODEL of type int</td>
</tr>
<tr>
<td>SUCCESS</td>
<td>Constant SUCCESS of type int</td>
</tr>
<tr>
<td>ERR_LASTCODE</td>
<td>Constant ERR_LASTCODE of type int</td>
</tr>
<tr>
<td>ERR_TYPE</td>
<td>Constant ERR_TYPE of type int</td>
</tr>
<tr>
<td>ERR_REQUEST</td>
<td>Constant ERR_REQUEST of type int</td>
</tr>
<tr>
<td>ERR_OP</td>
<td>Constant ERR_OP of type int</td>
</tr>
<tr>
<td>ERR_GROUP</td>
<td>Constant ERR_GROUP of type int</td>
</tr>
<tr>
<td>ERR_INFO</td>
<td>Constant ERR_INFO of type int</td>
</tr>
<tr>
<td>ERR_ERRHANDLER</td>
<td>Constant ERR_ERRHANDLER of type int</td>
</tr>
<tr>
<td>ERR_SESSION</td>
<td>Constant ERR_SESSION of type int</td>
</tr>
<tr>
<td>ERR_COMM</td>
<td>Constant ERR_COMM of type int</td>
</tr>
<tr>
<td>ERR_WIN</td>
<td>Constant ERR_WIN of type int</td>
</tr>
<tr>
<td>ERR_FILE</td>
<td>Constant ERR_FILE of type int</td>
</tr>
<tr>
<td>ERR_BUFFER</td>
<td>Constant ERR_BUFFER of type int</td>
</tr>
<tr>
<td>ERR_COUNT</td>
<td>Constant ERR_COUNT of type int</td>
</tr>
<tr>
<td>ERR_TAG</td>
<td>Constant ERR_TAG of type int</td>
</tr>
<tr>
<td>ERR_RANK</td>
<td>Constant ERR_RANK of type int</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR_ROOT</td>
<td>Constant ERR_ROOT of type int</td>
</tr>
<tr>
<td>ERR_TRUNCATE</td>
<td>Constant ERR_TRUNCATE of type int</td>
</tr>
<tr>
<td>ERR_IN_STATUS</td>
<td>Constant ERR_IN_STATUS of type int</td>
</tr>
<tr>
<td>ERR_PENDING</td>
<td>Constant ERR_PENDING of type int</td>
</tr>
<tr>
<td>ERR_TOPOLOGY</td>
<td>Constant ERR_TOPOLOGY of type int</td>
</tr>
<tr>
<td>ERR_DIMS</td>
<td>Constant ERR_DIMS of type int</td>
</tr>
<tr>
<td>ERR_ARG</td>
<td>Constant ERR_ARG of type int</td>
</tr>
<tr>
<td>ERR_OTHER</td>
<td>Constant ERR_OTHER of type int</td>
</tr>
<tr>
<td>ERR_UNKNOWN</td>
<td>Constant ERR_UNKNOWN of type int</td>
</tr>
<tr>
<td>ERR_INTERN</td>
<td>Constant ERR_INTERN of type int</td>
</tr>
<tr>
<td>ERR_KEYVAL</td>
<td>Constant ERR_KEYVAL of type int</td>
</tr>
<tr>
<td>ERR_NO_MEM</td>
<td>Constant ERR_NO_MEM of type int</td>
</tr>
<tr>
<td>ERR_INFO_KEY</td>
<td>Constant ERR_INFO_KEY of type int</td>
</tr>
<tr>
<td>ERR_INFO_VALUE</td>
<td>Constant ERR_INFO_VALUE of type int</td>
</tr>
<tr>
<td>ERR_INFO_NOKEY</td>
<td>Constant ERR_INFO_NOKEY of type int</td>
</tr>
<tr>
<td>ERR_SPAWN</td>
<td>Constant ERR_SPAWN of type int</td>
</tr>
<tr>
<td>ERR_PORT</td>
<td>Constant ERR_PORT of type int</td>
</tr>
<tr>
<td>ERR_SERVICE</td>
<td>Constant ERR_SERVICE of type int</td>
</tr>
<tr>
<td>ERR_NAME</td>
<td>Constant ERR_NAME of type int</td>
</tr>
<tr>
<td>ERR_PROC_ABORTED</td>
<td>Constant ERR_PROC_ABORTED of type int</td>
</tr>
<tr>
<td>ERR_BASE</td>
<td>Constant ERR_BASE of type int</td>
</tr>
<tr>
<td>ERR_SIZE</td>
<td>Constant ERR_SIZE of type int</td>
</tr>
<tr>
<td>ERR_DISP</td>
<td>Constant ERR_DISP of type int</td>
</tr>
<tr>
<td>ERR_ASSERT</td>
<td>Constant ERR_ASSERT of type int</td>
</tr>
<tr>
<td>ERR_LOCKTYPE</td>
<td>Constant ERR_LOCKTYPE of type int</td>
</tr>
<tr>
<td>ERR_RMA_CONFLICT</td>
<td>Constant ERR_RMA_CONFLICT of type int</td>
</tr>
<tr>
<td>ERR_RMA_SYNC</td>
<td>Constant ERR_RMA_SYNC of type int</td>
</tr>
<tr>
<td>ERR_RMA_RANGE</td>
<td>Constant ERR_RMA_RANGE of type int</td>
</tr>
<tr>
<td>ERR_RMA_ATTACH</td>
<td>Constant ERR_RMA_ATTACH of type int</td>
</tr>
<tr>
<td>ERR_RMA_SHARED</td>
<td>Constant ERR_RMA_SHARED of type int</td>
</tr>
<tr>
<td>ERR_RMA_FLAVOR</td>
<td>Constant ERR_RMA_FLAVOR of type int</td>
</tr>
<tr>
<td>ERR_BAD_FILE</td>
<td>Constant ERR_BAD_FILE of type int</td>
</tr>
<tr>
<td>ERR_NO_SUCH_FILE</td>
<td>Constant ERR_NO_SUCH_FILE of type int</td>
</tr>
<tr>
<td>ERR_FILE_EXISTS</td>
<td>Constant ERR_FILE_EXISTS of type int</td>
</tr>
<tr>
<td>ERR_FILE_IN_USE</td>
<td>Constant ERR_FILE_IN_USE of type int</td>
</tr>
<tr>
<td>ERR_AMODE</td>
<td>Constant ERR_AMODE of type int</td>
</tr>
<tr>
<td>ERR_ACCESS</td>
<td>Constant ERR_ACCESS of type int</td>
</tr>
<tr>
<td>ERR_READ_ONLY</td>
<td>Constant ERR_READ_ONLY of type int</td>
</tr>
<tr>
<td>ERR_NO_SPACE</td>
<td>Constant ERR_NO_SPACE of type int</td>
</tr>
<tr>
<td>ERR_QUOTA</td>
<td>Constant ERR_QUOTA of type int</td>
</tr>
<tr>
<td>ERR_NOTSAME</td>
<td>Constant ERR_NOTSAME of type int</td>
</tr>
<tr>
<td>ERR_IO</td>
<td>Constant ERR_IO of type int</td>
</tr>
<tr>
<td>ERR_UNSUPPORTED_OPERATION</td>
<td>Constant ERR_UNSUPPORTED_OPERATION of type int</td>
</tr>
<tr>
<td>ERR_UNSUPPORTED_DATAREP</td>
<td>Constant ERR_UNSUPPORTED_DATAREP of type int</td>
</tr>
<tr>
<td>ERR_CONVERSION</td>
<td>Constant ERR_CONVERSION of type int</td>
</tr>
<tr>
<td>ERR_DUP_DATAREP</td>
<td>Constant ERR_DUP_DATAREP of type int</td>
</tr>
<tr>
<td>ERR_VALUE_TOO_LARGE</td>
<td>Constant ERR_VALUE_TOO_LARGE of type int</td>
</tr>
<tr>
<td>ERR_REVOKED</td>
<td>Constant ERR_REVOKED of type int</td>
</tr>
<tr>
<td>ERR_PROC_FAILED</td>
<td>Constant ERR_PROC_FAILED of type int</td>
</tr>
<tr>
<td>ERR_PROC_FAILED_PENDING</td>
<td>Constant ERR_PROC_FAILED_PENDING of type int</td>
</tr>
<tr>
<td>ORDER_C</td>
<td>Constant ORDER_C of type int</td>
</tr>
<tr>
<td>ORDER_FORTRAN</td>
<td>Constant ORDER_FORTRAN of type int</td>
</tr>
<tr>
<td>Constant</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ORDER_F</td>
<td>Constant ORDER_F of type int</td>
</tr>
<tr>
<td>TYPECLASS_INTEGER</td>
<td>Constant TYPECLASS_INTEGER of type int</td>
</tr>
<tr>
<td>TYPECLASS_REAL</td>
<td>Constant TYPECLASS_REAL of type int</td>
</tr>
<tr>
<td>TYPECLASS_COMPLEX</td>
<td>Constant TYPECLASS_COMPLEX of type int</td>
</tr>
<tr>
<td>DISTRIBUT_NONE</td>
<td>Constant DISTRIBUT_NONE of type int</td>
</tr>
<tr>
<td>DISTRIBUT_BLOCK</td>
<td>Constant DISTRIBUT_BLOCK of type int</td>
</tr>
<tr>
<td>DISTRIBUT_CYCLIC</td>
<td>Constant DISTRIBUT_CYCLIC of type int</td>
</tr>
<tr>
<td>DISTRIBUT_DFLT_DARG</td>
<td>Constant DISTRIBUT_DFLT_DARG of type int</td>
</tr>
<tr>
<td>COMBINER_NAMED</td>
<td>Constant COMBINER_NAMED of type int</td>
</tr>
<tr>
<td>COMBINER_DUP</td>
<td>Constant COMBINER_DUP of type int</td>
</tr>
<tr>
<td>COMBINER_CONTIGUOUS</td>
<td>Constant COMBINER_CONTIGUOUS of type int</td>
</tr>
<tr>
<td>COMBINER_VECTOR</td>
<td>Constant COMBINER_VECTOR of type int</td>
</tr>
<tr>
<td>COMBINER_HVECTOR</td>
<td>Constant COMBINER_HVECTOR of type int</td>
</tr>
<tr>
<td>COMBINER_INDEXED</td>
<td>Constant COMBINER_INDEXED of type int</td>
</tr>
<tr>
<td>COMBINER_HINDEXED</td>
<td>Constant COMBINER_HINDEXED of type int</td>
</tr>
<tr>
<td>COMBINER_INDEXED_BLOCK</td>
<td>Constant COMBINER_INDEXED_BLOCK of type int</td>
</tr>
<tr>
<td>COMBINER_HINDEXED_BLOCK</td>
<td>Constant COMBINER_HINDEXED_BLOCK of type int</td>
</tr>
<tr>
<td>COMBINER_STRUCT</td>
<td>Constant COMBINER_STRUCT of type int</td>
</tr>
<tr>
<td>COMBINER_SUBARRAY</td>
<td>Constant COMBINER_SUBARRAY of type int</td>
</tr>
<tr>
<td>COMBINER_DARRAY</td>
<td>Constant COMBINER_DARRAY of type int</td>
</tr>
<tr>
<td>COMBINER_RESIZED</td>
<td>Constant COMBINER_RESIZED of type int</td>
</tr>
<tr>
<td>COMBINER_F90_INTEGER</td>
<td>Constant COMBINER_F90_INTEGER of type int</td>
</tr>
<tr>
<td>COMBINER_F90_REAL</td>
<td>Constant COMBINER_F90_REAL of type int</td>
</tr>
<tr>
<td>COMBINER_F90_COMPLEX</td>
<td>Constant COMBINER_F90_COMPLEX of type int</td>
</tr>
<tr>
<td>F_SOURCE</td>
<td>Constant F_SOURCE of type int</td>
</tr>
<tr>
<td>F_TAG</td>
<td>Constant F_TAG of type int</td>
</tr>
<tr>
<td>F_ERROR</td>
<td>Constant F_ERROR of type int</td>
</tr>
<tr>
<td>F_STATUS_SIZE</td>
<td>Constant F_STATUS_SIZE of type int</td>
</tr>
<tr>
<td>IDENT</td>
<td>Constant IDENT of type int</td>
</tr>
<tr>
<td>CONGRUENT</td>
<td>Constant CONGRUENT of type int</td>
</tr>
<tr>
<td>SIMILAR</td>
<td>Constant SIMILAR of type int</td>
</tr>
<tr>
<td>UNEQUAL</td>
<td>Constant UNEQUAL of type int</td>
</tr>
<tr>
<td>CART</td>
<td>Constant CART of type int</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Constant GRAPH of type int</td>
</tr>
<tr>
<td>DIST_GRAPH</td>
<td>Constant DIST_GRAPH of type int</td>
</tr>
<tr>
<td>UNWEIGHTED</td>
<td>Constant UNWEIGHTED of type int</td>
</tr>
<tr>
<td>WEIGHTS_EMPTY</td>
<td>Constant WEIGHTS_EMPTY of type int</td>
</tr>
<tr>
<td>COMM_TYPE_SHARED</td>
<td>Constant COMM_TYPE_SHARED of type int</td>
</tr>
<tr>
<td>COMM_TYPE_HW_GUIDED</td>
<td>Constant COMM_TYPE_HW_GUIDED of type int</td>
</tr>
<tr>
<td>COMM_TYPE_HW_UNGUIDED</td>
<td>Constant COMM_TYPE_HW_UNGUIDED of type int</td>
</tr>
<tr>
<td>BSEND_OVERHEAD</td>
<td>Constant BSEND_OVERHEAD of type int</td>
</tr>
<tr>
<td>WIN_FLAVOR_CREATE</td>
<td>Constant WIN_FLAVOR_CREATE of type int</td>
</tr>
<tr>
<td>WIN_FLAVOR_ALLOCATE</td>
<td>Constant WIN_FLAVOR_ALLOCATE of type int</td>
</tr>
<tr>
<td>WIN_FLAVOR_DYNAMIC</td>
<td>Constant WIN_FLAVOR_DYNAMIC of type int</td>
</tr>
<tr>
<td>WIN_FLAVOR_SHARED</td>
<td>Constant WIN_FLAVOR_SHARED of type int</td>
</tr>
<tr>
<td>WIN_SEPARATE</td>
<td>Constant WIN_SEPARATE of type int</td>
</tr>
<tr>
<td>WIN_UNIFIED</td>
<td>Constant WIN_UNIFIED of type int</td>
</tr>
<tr>
<td>MODE_NOCHECK</td>
<td>Constant MODE_NOCHECK of type int</td>
</tr>
<tr>
<td>MODE_NOSTORE</td>
<td>Constant MODE_NOSTORE of type int</td>
</tr>
<tr>
<td>MODE_NOPUT</td>
<td>Constant MODE_NOPUT of type int</td>
</tr>
<tr>
<td>MODE_NOPRECEDE</td>
<td>Constant MODE_NOPRECEDE of type int</td>
</tr>
<tr>
<td>MODE_NOSUCCEED</td>
<td>Constant MODE_NOSUCCEED of type int</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK_EXCLUSIVE</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>LOCK_SHARED</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_RDONLY</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_WRONLY</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_RDWR</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_CREATE</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_EXCL</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_DELETE_ON_CLOSE</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_UNIQUE_OPEN</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_SEQUENTIAL</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MODE_APPEND</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>SEEK_SET</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>SEEK_CUR</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>SEEK_END</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>DISPLACEMENT_CURRENT</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>DISP_CUR</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>THREAD_SINGLE</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>THREAD_FUNNELED</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>THREAD_SERIALIZED</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>THREAD_MULTIPLE</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>VERSION</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>SUBVERSION</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_PROCESSOR_NAME</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_ERROR_STRING</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_PORT_NAME</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_INFO_KEY</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_INFO_VAL</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_OBJECT_NAME</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_DATAREP_STRING</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_LIBRARY_VERSION_STRING</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_PSET_NAME_LEN</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>MAX_STRING_TAG_LEN</td>
<td>Constant of type int</td>
</tr>
<tr>
<td>DATATYPE_NULL</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>PACKED</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>BYTE</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>AINT</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>OFFSET</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>COUNT</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>CHAR</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>WCHAR</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>SIGNED_CHAR</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>SHORT</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>INT</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>LONG</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>LONG_LONG</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_CHAR</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_SHORT</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_LONG</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>UNSIGNED_LONG_LONG</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Object of type Datatype</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>Object of type Datatype</td>
</tr>
</tbody>
</table>

continues on next page
Table 7 – continued from previous page

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG_DOUBLE</td>
<td>Object LONG_DOUBLE of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>C_BOOL</td>
<td>Object C_BOOL of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INT8_T</td>
<td>Object INT8_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INT16_T</td>
<td>Object INT16_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INT32_T</td>
<td>Object INT32_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INT64_T</td>
<td>Object INT64_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>UINT8_T</td>
<td>Object UINT8_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>UINT16_T</td>
<td>Object UINT16_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>UINT32_T</td>
<td>Object UINT32_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>UINT64_T</td>
<td>Object UINT64_T of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>C_COMPLEX</td>
<td>Object C_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>C_FLOAT_COMPLEX</td>
<td>Object C_FLOAT_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>C_LONG_DOUBLE_COMPLEX</td>
<td>Object C_LONG_DOUBLE_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>CXX_BOOL</td>
<td>Object CXX_BOOL of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>CXX_FLOAT_COMPLEX</td>
<td>Object CXX_FLOAT_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>CXX_DOUBLE_COMPLEX</td>
<td>Object CXX_DOUBLE_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>CXX_LONG_DOUBLE_COMPLEX</td>
<td>Object CXX_LONG_DOUBLE_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>SHORT_INT</td>
<td>Object SHORT_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INT_INT</td>
<td>Object INT_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>TWOINT</td>
<td>Object TWOINT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LONG_INT</td>
<td>Object LONG_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>FLOAT_INT</td>
<td>Object FLOAT_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>DOUBLE_INT</td>
<td>Object DOUBLE_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LONG_DOUBLE_INT</td>
<td>Object LONG_DOUBLE_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>Object CHARACTER of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>Object LOGICAL of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Object INTEGER of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>REAL</td>
<td>Object REAL of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>DOUBLE_PRECISION</td>
<td>Object DOUBLE_PRECISION of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>COMPLEX</td>
<td>Object COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>DOUBLE_COMPLEX</td>
<td>Object DOUBLE_COMPLEX of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LOGICAL1</td>
<td>Object LOGICAL1 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LOGICAL2</td>
<td>Object LOGICAL2 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LOGICAL4</td>
<td>Object LOGICAL4 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>LOGICAL8</td>
<td>Object LOGICAL8 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INTEGER1</td>
<td>Object INTEGER1 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INTEGER2</td>
<td>Object INTEGER2 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INTEGER4</td>
<td>Object INTEGER4 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INTEGER8</td>
<td>Object INTEGER8 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>INTEGER16</td>
<td>Object INTEGER16 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>REAL2</td>
<td>Object REAL2 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>REAL4</td>
<td>Object REAL4 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>REAL8</td>
<td>Object REAL8 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>REAL16</td>
<td>Object REAL16 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>COMPLEX4</td>
<td>Object COMPLEX4 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>COMPLEX8</td>
<td>Object COMPLEX8 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>COMPLEX16</td>
<td>Object COMPLEX16 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>COMPLEX32</td>
<td>Object COMPLEX32 of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>UNSIGNED_INT</td>
<td>Object UNSIGNED_INT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>SIGNED_SHORT</td>
<td>Object SIGNED_SHORT of type</td>
<td>Datatype</td>
</tr>
<tr>
<td>SIGNED_INT</td>
<td>Object SIGNED_INT of type</td>
<td>Datatype</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Object</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNED_LONG</td>
<td>Object SIGNED_LONG of type Datatype</td>
</tr>
<tr>
<td>SIGNED_LONG_LONG</td>
<td>Object SIGNED_LONG_LONG of type Datatype</td>
</tr>
<tr>
<td>BOOL</td>
<td>Object BOOL of type Datatype</td>
</tr>
<tr>
<td>SINT8_T</td>
<td>Object SINT8_T of type Datatype</td>
</tr>
<tr>
<td>SINT16_T</td>
<td>Object SINT16_T of type Datatype</td>
</tr>
<tr>
<td>SINT32_T</td>
<td>Object SINT32_T of type Datatype</td>
</tr>
<tr>
<td>SINT64_T</td>
<td>Object SINT64_T of type Datatype</td>
</tr>
<tr>
<td>F_BOOL</td>
<td>Object F_BOOL of type Datatype</td>
</tr>
<tr>
<td>F_INT</td>
<td>Object F_INT of type Datatype</td>
</tr>
<tr>
<td>F_FLOAT</td>
<td>Object F_FLOAT of type Datatype</td>
</tr>
<tr>
<td>F_DOUBLE</td>
<td>Object F_DOUBLE of type Datatype</td>
</tr>
<tr>
<td>F_COMPLEX</td>
<td>Object F_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>F_FLOAT_COMPLEX</td>
<td>Object F_FLOAT_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>F_DOUBLE_COMPLEX</td>
<td>Object F_DOUBLE_COMPLEX of type Datatype</td>
</tr>
<tr>
<td>REQUEST_NULL</td>
<td>Object REQUEST_NULL of type Request</td>
</tr>
<tr>
<td>MESSAGE_NULL</td>
<td>Object MESSAGE_NULL of type Message</td>
</tr>
<tr>
<td>MESSAGE_NO_PROC</td>
<td>Object MESSAGE_NO_PROC of type Message</td>
</tr>
<tr>
<td>OP_NULL</td>
<td>Object OP_NULL of type Op</td>
</tr>
<tr>
<td>MAX</td>
<td>Object MAX of type Op</td>
</tr>
<tr>
<td>MIN</td>
<td>Object MIN of type Op</td>
</tr>
<tr>
<td>SUM</td>
<td>Object SUM of type Op</td>
</tr>
<tr>
<td>PROD</td>
<td>Object PROD of type Op</td>
</tr>
<tr>
<td>LAND</td>
<td>Object LAND of type Op</td>
</tr>
<tr>
<td>BAND</td>
<td>Object BAND of type Op</td>
</tr>
<tr>
<td>LOR</td>
<td>Object LOR of type Op</td>
</tr>
<tr>
<td>BOR</td>
<td>Object BOR of type Op</td>
</tr>
<tr>
<td>LXOR</td>
<td>Object LXOR of type Op</td>
</tr>
<tr>
<td>BXOR</td>
<td>Object BXOR of type Op</td>
</tr>
<tr>
<td>MAXLOC</td>
<td>Object MAXLOC of type Op</td>
</tr>
<tr>
<td>MINLOC</td>
<td>Object MINLOC of type Op</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Object REPLACE of type Op</td>
</tr>
<tr>
<td>NO_OP</td>
<td>Object NO_OP of type Op</td>
</tr>
<tr>
<td>GROUP_NULL</td>
<td>Object GROUP_NULL of type Group</td>
</tr>
<tr>
<td>GROUP_EMPTY</td>
<td>Object GROUP_EMPTY of type Group</td>
</tr>
<tr>
<td>INFO_NULL</td>
<td>Object INFO_NULL of type Info</td>
</tr>
<tr>
<td>INFO_ENV</td>
<td>Object INFO_ENV of type Info</td>
</tr>
<tr>
<td>ERRHANDLER_NULL</td>
<td>Object ERRHANDLER_NULL of type Errhandler</td>
</tr>
<tr>
<td>ERRORS_RETURN</td>
<td>Object ERRORS_RETURN of type Errhandler</td>
</tr>
<tr>
<td>ERRORS_ABORT</td>
<td>Object ERRORS_ABORT of type Errhandler</td>
</tr>
<tr>
<td>ERRORS_ARE_FATAL</td>
<td>Object ERRORS_ARE_FATAL of type Errhandler</td>
</tr>
<tr>
<td>SESSION_NULL</td>
<td>Object SESSION_NULL of type Session</td>
</tr>
<tr>
<td>COMM_NULL</td>
<td>Object COMM_NULL of type Comm</td>
</tr>
<tr>
<td>COMM_SELF</td>
<td>Object COMM_SELF of type Intracomm</td>
</tr>
<tr>
<td>COMM_WORLD</td>
<td>Object COMM_WORLD of type Intracomm</td>
</tr>
<tr>
<td>WIN_NULL</td>
<td>Object WIN_NULL of type Win</td>
</tr>
<tr>
<td>FILE_NULL</td>
<td>Object FILE_NULL of type File</td>
</tr>
<tr>
<td>pickle</td>
<td>Object pickle of type Pickle</td>
</tr>
</tbody>
</table>
mpi4py.MPI.UNDEFINED

mpi4py.MPI.UNDEFINED: int = UNDEFINED
Constant UNDEFINED of type int

mpi4py.MPI.ANY_SOURCE

mpi4py.MPI.ANY_SOURCE: int = ANY_SOURCE
Constant ANY_SOURCE of type int

mpi4py.MPI.ANY_TAG

mpi4py.MPI.ANY_TAG: int = ANY_TAG
Constant ANY_TAG of type int

mpi4py.MPI.PROC_NULL

mpi4py.MPI.PROC_NULL: int = PROC_NULL
Constant PROC_NULL of type int

mpi4py.MPI.ROOT

mpi4py.MPI.ROOT: int = ROOT
Constant ROOT of type int

mpi4py.MPI.BOTTOM

mpi4py.MPI.BOTTOM: BottomType = BOTTOM
Constant BOTTOM of type BottomType

mpi4py.MPI.IN_PLACE

mpi4py.MPI.IN_PLACE: InPlaceType = IN_PLACE
Constant IN_PLACE of type InPlaceType

mpi4py.MPI.KEYVAL_INVALID

mpi4py.MPI.KEYVAL_INVALID: int = KEYVAL_INVALID
Constant KEYVAL_INVALID of type int
mpi4py.MPI.WIN_SIZE

mpi4py.MPI.WIN_SIZE: int = WIN_SIZE
Constant WIN_SIZE of type int

mpi4py.MPI.WIN_DISP_UNIT

mpi4py.MPI.WIN_DISP_UNIT: int = WIN_DISP_UNIT
Constant WIN_DISP_UNIT of type int

mpi4py.MPI.WIN_CREATE_FLAVOR

mpi4py.MPI.WIN_CREATE_FLAVOR: int = WIN_CREATE_FLAVOR
Constant WIN_CREATE_FLAVOR of type int

mpi4py.MPI.WIN_FLAVOR

mpi4py.MPI.WIN_FLAVOR: int = WIN_FLAVOR
Constant WIN_FLAVOR of type int

mpi4py.MPI.WIN_MODEL

mpi4py.MPI.WIN_MODEL: int = WIN_MODEL
Constant WIN_MODEL of type int

mpi4py.MPI.SUCCESS

mpi4py.MPI.SUCCESS: int = SUCCESS
Constant SUCCESS of type int

mpi4py.MPI.ERR_LASTCODE

mpi4py.MPI.ERR_LASTCODE: int = ERR_LASTCODE
Constant ERR_LASTCODE of type int

mpi4py.MPI.ERR_TYPE

mpi4py.MPI.ERR_TYPE: int = ERR_TYPE
Constant ERR_TYPE of type int
mpi4py.MPI.ERR_REQUEST

mpi4py.MPI.ERR_REQUEST: int = ERR_REQUEST
Constant ERR_REQUEST of type int

mpi4py.MPI.ERR_OP

mpi4py.MPI.ERR_OP: int = ERR_OP
Constant ERR_OP of type int

mpi4py.MPI.ERR_GROUP

mpi4py.MPI.ERR_GROUP: int = ERR_GROUP
Constant ERR_GROUP of type int

mpi4py.MPI.ERR_INFO

mpi4py.MPI.ERR_INFO: int = ERR_INFO
Constant ERR_INFO of type int

mpi4py.MPI.ERR_ERRHANDLER

mpi4py.MPI.ERR_ERRHANDLER: int = ERR_ERRHANDLER
Constant ERR_ERRHANDLER of type int

mpi4py.MPI.ERR_SESSION

mpi4py.MPI.ERR_SESSION: int = ERR_SESSION
Constant ERR_SESSION of type int

mpi4py.MPI.ERR_COMM

mpi4py.MPI.ERR_COMM: int = ERR_COMM
Constant ERR_COMM of type int

mpi4py.MPI.ERR_WIN

mpi4py.MPI.ERR_WIN: int = ERR_WIN
Constant ERR_WIN of type int
mpi4py.MPI.ERR_FILE

mpi4py.MPI.ERR_FILE: int = ERR_FILE
Constant ERR_FILE of type int

mpi4py.MPI.ERR_BUFFER

mpi4py.MPI.ERR_BUFFER: int = ERR_BUFFER
Constant ERR_BUFFER of type int

mpi4py.MPI.ERR_COUNT

mpi4py.MPI.ERR_COUNT: int = ERR_COUNT
Constant ERR_COUNT of type int

mpi4py.MPI.ERR_TAG

mpi4py.MPI.ERR_TAG: int = ERR_TAG
Constant ERR_TAG of type int

mpi4py.MPI.ERR_RANK

mpi4py.MPI.ERR_RANK: int = ERR_RANK
Constant ERR_RANK of type int

mpi4py.MPI.ERR_ROOT

mpi4py.MPI.ERR_ROOT: int = ERR_ROOT
Constant ERR_ROOT of type int

mpi4py.MPI.ERR_TRUNCATE

mpi4py.MPI.ERR_TRUNCATE: int = ERR_TRUNCATE
Constant ERR_TRUNCATE of type int

mpi4py.MPI.ERR_IN_STATUS

mpi4py.MPI.ERR_IN_STATUS: int = ERR_IN_STATUS
Constant ERR_IN_STATUS of type int
mpi4py.MPI.ERR_PENDING
mpi4py.MPI.ERR_PENDING: int = ERR_PENDING
    Constant ERR_PENDING of type int

mpi4py.MPI.ERR_TOPOLOGY
mpi4py.MPI.ERR_TOPOLOGY: int = ERR_TOPOLOGY
    Constant ERR_TOPOLOGY of type int

mpi4py.MPI.ERR_DIMS
mpi4py.MPI.ERR_DIMS: int = ERR_DIMS
    Constant ERR_DIMS of type int

mpi4py.MPI.ERR_ARG
mpi4py.MPI.ERR_ARG: int = ERR_ARG
    Constant ERR_ARG of type int

mpi4py.MPI.ERR_OTHER
mpi4py.MPI.ERR_OTHER: int = ERR_OTHER
    Constant ERR_OTHER of type int

mpi4py.MPI.ERR_UNKNOWN
mpi4py.MPI.ERR_UNKNOWN: int = ERR_UNKNOWN
    Constant ERR_UNKNOWN of type int

mpi4py.MPI.ERR_INTERN
mpi4py.MPI.ERR_INTERN: int = ERR_INTERN
    Constant ERR_INTERN of type int

mpi4py.MPI.ERR_KEYVAL
mpi4py.MPI.ERR_KEYVAL: int = ERR_KEYVAL
    Constant ERR_KEYVAL of type int
mpi4py.MPI.ERR_NO_MEM

mpi4py.MPI.ERR_NO_MEM: int = ERR_NO_MEM
Constant ERR_NO_MEM of type int

mpi4py.MPI.ERR_INFO_KEY

mpi4py.MPI.ERR_INFO_KEY: int = ERR_INFO_KEY
Constant ERR_INFO_KEY of type int

mpi4py.MPI.ERR_INFO_VALUE

mpi4py.MPI.ERR_INFO_VALUE: int = ERR_INFO_VALUE
Constant ERR_INFO_VALUE of type int

mpi4py.MPI.ERR_INFO_NOKEY

mpi4py.MPI.ERR_INFO_NOKEY: int = ERR_INFO_NOKEY
Constant ERR_INFO_NOKEY of type int

mpi4py.MPI.ERR_SPAWN

mpi4py.MPI.ERR_SPAWN: int = ERR_SPAWN
Constant ERR_SPAWN of type int

mpi4py.MPI.ERR_PORT

mpi4py.MPI.ERR_PORT: int = ERR_PORT
Constant ERR_PORT of type int

mpi4py.MPI.ERR_SERVICE

mpi4py.MPI.ERR_SERVICE: int = ERR_SERVICE
Constant ERR_SERVICE of type int

mpi4py.MPI.ERR_NAME

mpi4py.MPI.ERR_NAME: int = ERR_NAME
Constant ERR_NAME of type int
mpi4py.MPI.ERR_PROC_ABORTED

mpi4py.MPI.ERR_PROC_ABORTED: int = ERR_PROC_ABORTED
    Constant ERR_PROC_ABORTED of type int

mpi4py.MPI.ERR_BASE

mpi4py.MPI.ERR_BASE: int = ERR_BASE
    Constant ERR_BASE of type int

mpi4py.MPI.ERR_SIZE

mpi4py.MPI.ERR_SIZE: int = ERR_SIZE
    Constant ERR_SIZE of type int

mpi4py.MPI.ERR_DISP

mpi4py.MPI.ERR_DISP: int = ERR_DISP
    Constant ERR_DISP of type int

mpi4py.MPI.ERR_ASSERT

mpi4py.MPI.ERR_ASSERT: int = ERR_ASSERT
    Constant ERR_ASSERT of type int

mpi4py.MPI.ERR_LOCKTYPE

mpi4py.MPI.ERR_LOCKTYPE: int = ERR_LOCKTYPE
    Constant ERR_LOCKTYPE of type int

mpi4py.MPI.ERR_RMA_CONFLICT

mpi4py.MPI.ERR_RMA_CONFLICT: int = ERR_RMA_CONFLICT
    Constant ERR_RMA_CONFLICT of type int

mpi4py.MPI.ERR_RMA_SYNC

mpi4py.MPI.ERR_RMA_SYNC: int = ERR_RMA_SYNC
    Constant ERR_RMA_SYNC of type int
mpi4py.MPI.ERR_RMA_RANGE

mpi4py.MPI.ERR_RMA_RANGE: int = ERR_RMA_RANGE
  Constant ERR_RMA_RANGE of type int

mpi4py.MPI.ERR_RMA_ATTACH

mpi4py.MPI.ERR_RMA_ATTACH: int = ERR_RMA_ATTACH
  Constant ERR_RMA_ATTACH of type int

mpi4py.MPI.ERR_RMA_SHARED

mpi4py.MPI.ERR_RMA_SHARED: int = ERR_RMA_SHARED
  Constant ERR_RMA_SHARED of type int

mpi4py.MPI.ERR_RMA_FLAVOR

mpi4py.MPI.ERR_RMA_FLAVOR: int = ERR_RMA_FLAVOR
  Constant ERR_RMA_FLAVOR of type int

mpi4py.MPI.ERR_BAD_FILE

mpi4py.MPI.ERR_BAD_FILE: int = ERR_BAD_FILE
  Constant ERR_BAD_FILE of type int

mpi4py.MPI.ERR_NO_SUCH_FILE

mpi4py.MPI.ERR_NO_SUCH_FILE: int = ERR_NO_SUCH_FILE
  Constant ERR_NO_SUCH_FILE of type int

mpi4py.MPI.ERR_FILE_EXISTS

mpi4py.MPI.ERR_FILE_EXISTS: int = ERR_FILE_EXISTS
  Constant ERR_FILE_EXISTS of type int

mpi4py.MPI.ERR_FILE_IN_USE

mpi4py.MPI.ERR_FILE_IN_USE: int = ERR_FILE_IN_USE
  Constant ERR_FILE_IN_USE of type int
mpi4py.MPI.ERR_AMODE

mpi4py.MPI.ERR_AMODE: int = ERR_AMODE
    Constant ERR_AMODE of type int

mpi4py.MPI.ERR_ACCESS

mpi4py.MPI.ERR_ACCESS: int = ERR_ACCESS
    Constant ERR_ACCESS of type int

mpi4py.MPI.ERR_READ_ONLY

mpi4py.MPI.ERR_READ_ONLY: int = ERR_READ_ONLY
    Constant ERR_READ_ONLY of type int

mpi4py.MPI.ERR_NO_SPACE

mpi4py.MPI.ERR_NO_SPACE: int = ERR_NO_SPACE
    Constant ERR_NO_SPACE of type int

mpi4py.MPI.ERR_QUOTA

mpi4py.MPI.ERR_QUOTA: int = ERR_QUOTA
    Constant ERR_QUOTA of type int

mpi4py.MPI.ERR_NOT_SAME

mpi4py.MPI.ERR_NOT_SAME: int = ERR_NOT_SAME
    Constant ERR_NOT_SAME of type int

mpi4py.MPI.ERR_IO

mpi4py.MPI.ERR_IO: int = ERR_IO
    Constant ERR_IO of type int

mpi4py.MPI.ERR_UNSUPPORTED_OPERATION

mpi4py.MPI.ERR_UNSUPPORTED_OPERATION: int = ERR_UNSUPPORTED_OPERATION
    Constant ERR_UNSUPPORTED_OPERATION of type int
mpi4py.MPI.ERR_UNSUPPORTED_DATAREP

mpi4py.MPI.ERR_UNSUPPORTED_DATAREP: int = ERR_UNSUPPORTED_DATAREP
    Constant ERR_UNSUPPORTED_DATAREP of type int

mpi4py.MPI.ERR_CONVERSION

mpi4py.MPI.ERR_CONVERSION: int = ERR_CONVERSION
    Constant ERR_CONVERSION of type int

mpi4py.MPI.ERR_DUP_DATAREP

mpi4py.MPI.ERR_DUP_DATAREP: int = ERR_DUP_DATAREP
    Constant ERR_DUP_DATAREP of type int

mpi4py.MPI.ERR_VALUE_TOO_LARGE

mpi4py.MPI.ERR_VALUE_TOO_LARGE: int = ERR_VALUE_TOO_LARGE
    Constant ERR_VALUE_TOO_LARGE of type int

mpi4py.MPI.ERR_REVOKED

mpi4py.MPI.ERR_REVOKED: int = ERR_REVOKED
    Constant ERR_REVOKED of type int

mpi4py.MPI.ERR_PROC_FAILED

mpi4py.MPI.ERR_PROC_FAILED: int = ERR_PROC_FAILED
    Constant ERR_PROC_FAILED of type int

mpi4py.MPI.ERR_PROC_FAILED_PENDING

mpi4py.MPI.ERR_PROC_FAILED_PENDING: int = ERR_PROC_FAILED_PENDING
    Constant ERR_PROC_FAILED_PENDING of type int

mpi4py.MPI.ORDER_C

mpi4py.MPI.ORDER_C: int = ORDER_C
    Constant ORDER_C of type int
\begin{verbatim}
mpi4py.MPI.ORDER_FORTRAN

mpi4py.MPI.ORDER_FORTRAN: int = ORDER_FORTRAN
   Constant ORDER_FORTRAN of type int

mpi4py.MPI.ORDER_F

mpi4py.MPI.ORDER_F: int = ORDER_F
   Constant ORDER_F of type int

mpi4py.MPI.TYPECLASS_INTEGER

mpi4py.MPI.TYPECLASS_INTEGER: int = TYPECLASS_INTEGER
   Constant TYPECLASS_INTEGER of type int

mpi4py.MPI.TYPECLASS_REAL

mpi4py.MPI.TYPECLASS_REAL: int = TYPECLASS_REAL
   Constant TYPECLASS_REAL of type int

mpi4py.MPI.TYPECLASS_COMPLEX

mpi4py.MPI.TYPECLASS_COMPLEX: int = TYPECLASS_COMPLEX
   Constant TYPECLASS_COMPLEX of type int

mpi4py.MPI.DISTRIBUTE_NONE

mpi4py.MPI.DISTRIBUTE_NONE: int = DISTRIBUTION_NONE
   Constant DISTRIBUTION_NONE of type int

mpi4py.MPI.DISTRIBUTE_BLOCK

mpi4py.MPI.DISTRIBUTION_BLOCK: int = DISTRIBUTION_BLOCK
   Constant DISTRIBUTION_BLOCK of type int

mpi4py.MPI.DISTRIBUTE_CYCLIC

mpi4py.MPI.DISTRIBUTION_CYCLIC: int = DISTRIBUTION_CYCLIC
   Constant DISTRIBUTION_CYCLIC of type int
\end{verbatim}
mpi4py.MPI.DISTRIBUTE_DFLT_DARG

mpi4py.MPI.DISTRIBUTE_DFLT_DARG: int = DISTRIBUTE_DFLT_DARG
  Constant DISTRIBUTE_DFLT_DARG of type int

mpi4py.MPI.COMBINER_NAMED

mpi4py.MPI.COMBINER_NAMED: int = COMBINER_NAMED
  Constant COMBINER_NAMED of type int

mpi4py.MPI.COMBINER_DUP

mpi4py.MPI.COMBINER_DUP: int = COMBINER_DUP
  Constant COMBINER_DUP of type int

mpi4py.MPI.COMBINER_CONTIGUOUS

mpi4py.MPI.COMBINER_CONTIGUOUS: int = COMBINER_CONTIGUOUS
  Constant COMBINER_CONTIGUOUS of type int

mpi4py.MPI.COMBINER_VECTOR

mpi4py.MPI.COMBINER_VECTOR: int = COMBINER_VECTOR
  Constant COMBINER_VECTOR of type int

mpi4py.MPI.COMBINER_HVECTOR

mpi4py.MPI.COMBINER_HVECTOR: int = COMBINER_HVECTOR
  Constant COMBINER_HVECTOR of type int

mpi4py.MPI.COMBINER_INDEXED

mpi4py.MPI.COMBINER_INDEXED: int = COMBINER_INDEXED
  Constant COMBINER_INDEXED of type int

mpi4py.MPI.COMBINER_HINDEXED

mpi4py.MPI.COMBINER_HINDEXED: int = COMBINER_HINDEXED
  Constant COMBINER_HINDEXED of type int
mpi4py.MPI.COMBINER_INDEXED_BLOCK

mpi4py.MPI.COMBINER_INDEXED_BLOCK: int = COMBINER_INDEXED_BLOCK
    Constant COMBINER_INDEXED_BLOCK of type int

mpi4py.MPI.COMBINER_HINDEXED_BLOCK

mpi4py.MPI.COMBINER_HINDEXED_BLOCK: int = COMBINER_HINDEXED_BLOCK
    Constant COMBINER_HINDEXED_BLOCK of type int

mpi4py.MPI.COMBINER_STRUCT

mpi4py.MPI.COMBINER_STRUCT: int = COMBINER_STRUCT
    Constant COMBINER_STRUCT of type int

mpi4py.MPI.COMBINER_SUBARRAY

mpi4py.MPI.COMBINER_SUBARRAY: int = COMBINER_SUBARRAY
    Constant COMBINER_SUBARRAY of type int

mpi4py.MPI.COMBINER_DARRAY

mpi4py.MPI.COMBINER_DARRAY: int = COMBINER_DARRAY
    Constant COMBINER_DARRAY of type int

mpi4py.MPI.COMBINER_RESIZED

mpi4py.MPI.COMBINER_RESIZED: int = COMBINER_RESIZED
    Constant COMBINER_RESIZED of type int

mpi4py.MPI.COMBINER_F90_INTEGER

mpi4py.MPI.COMBINER_F90_INTEGER: int = COMBINER_F90_INTEGER
    Constant COMBINER_F90_INTEGER of type int

mpi4py.MPI.COMBINER_F90_REAL

mpi4py.MPI.COMBINER_F90_REAL: int = COMBINER_F90_REAL
    Constant COMBINER_F90_REAL of type int
mpi4py.MPI.COMBINER_F90_COMPLEX

mpi4py.MPI.COMBINER_F90_COMPLEX: int = COMBINER_F90_COMPLEX
  Constant COMBINER_F90_COMPLEX of type int

mpi4py.MPI.F_SOURCE

mpi4py.MPI.F_SOURCE: int = F_SOURCE
  Constant F_SOURCE of type int

mpi4py.MPI.F_TAG

mpi4py.MPI.F_TAG: int = F_TAG
  Constant F_TAG of type int

mpi4py.MPI.F_ERROR

mpi4py.MPI.F_ERROR: int = F_ERROR
  Constant F_ERROR of type int

mpi4py.MPI.F_STATUS_SIZE

mpi4py.MPI.F_STATUS_SIZE: int = F_STATUS_SIZE
  Constant F_STATUS_SIZE of type int

mpi4py.MPI.IDENT

mpi4py.MPI.IDENT: int = IDENT
  Constant IDENT of type int

mpi4py.MPI.CONGRUENT

mpi4py.MPI.CONGRUENT: int = CONGRUENT
  Constant CONGRUENT of type int

mpi4py.MPI.SIMILAR

mpi4py.MPI.SIMILAR: int = SIMILAR
  Constant SIMILAR of type int
mpi4py.MPI.UNEQUAL

mpi4py.MPI.UNEQUAL: int = UNEQUAL
    Constant UNEQUAL of type int

mpi4py.MPI.CART

mpi4py.MPI.CART: int = CART
    Constant CART of type int

mpi4py.MPI.GRAPH

mpi4py.MPI.GRAPH: int = GRAPH
    Constant GRAPH of type int

mpi4py.MPI.DIST_GRAPH

mpi4py.MPI.DIST_GRAPH: int = DIST_GRAPH
    Constant DIST_GRAPH of type int

mpi4py.MPI.UNWEIGHTED

mpi4py.MPI.UNWEIGHTED: int = UNWEIGHTED
    Constant UNWEIGHTED of type int

mpi4py.MPI.WEIGHTS_EMPTY

mpi4py.MPI.WEIGHTS_EMPTY: int = WEIGHTS_EMPTY
    Constant WEIGHTS_EMPTY of type int

mpi4py.MPI.COMM_TYPE_SHARED

mpi4py.MPI.COMM_TYPE_SHARED: int = COMM_TYPE_SHARED
    Constant COMM_TYPE_SHARED of type int

mpi4py.MPI.COMM_TYPE_HW_GUIDED

mpi4py.MPI.COMM_TYPE_HW_GUIDED: int = COMM_TYPE_HW_GUIDED
    Constant COMM_TYPE_HW_GUIDED of type int
mpi4py.MPI.COMM_TYPE_HW_UNGUIDED

mpi4py.MPI.COMM_TYPE_HW_UNGUIDED: int = COMM_TYPE_HW_UNGUIDED
Constant COMM_TYPE_HW_UNGUIDED of type int

mpi4py.MPI.BSEND_OVERHEAD

mpi4py.MPI.BSEND_OVERHEAD: int = BSEND_OVERHEAD
Constant BSEND_OVERHEAD of type int

mpi4py.MPI.WIN_FLAVOR_CREATE

mpi4py.MPI.WIN_FLAVOR_CREATE: int = WIN_FLAVOR_CREATE
Constant WIN_FLAVOR_CREATE of type int

mpi4py.MPI.WIN_FLAVOR_ALLOCATE

mpi4py.MPI.WIN_FLAVOR_ALLOCATE: int = WIN_FLAVOR_ALLOCATE
Constant WIN_FLAVOR_ALLOCATE of type int

mpi4py.MPI.WIN_FLAVOR_DYNAMIC

mpi4py.MPI.WIN_FLAVOR_DYNAMIC: int = WIN_FLAVOR_DYNAMIC
Constant WIN_FLAVOR_DYNAMIC of type int

mpi4py.MPI.WIN_FLAVOR_SHARED

mpi4py.MPI.WIN_FLAVOR_SHARED: int = WIN_FLAVOR_SHARED
Constant WIN_FLAVOR_SHARED of type int

mpi4py.MPI.WIN_SEPARATE

mpi4py.MPI.WIN_SEPARATE: int = WIN_SEPARATE
Constant WIN_SEPARATE of type int

mpi4py.MPI.WIN_UNIFIED

mpi4py.MPI.WIN_UNIFIED: int = WIN_UNIFIED
Constant WIN_UNIFIED of type int
mpi4py.MPI.MODE_WRONLY

mpi4py.MPI.MODE_WRONLY: int = MODE_WRONLY
  Constant MODE_WRONLY of type int

mpi4py.MPI.MODE_RDWR

mpi4py.MPI.MODE_RDWR: int = MODE_RDWR
  Constant MODE_RDWR of type int

mpi4py.MPI.MODE_CREATE

mpi4py.MPI.MODE_CREATE: int = MODE_CREATE
  Constant MODE_CREATE of type int

mpi4py.MPI.MODE_EXCL

mpi4py.MPI.MODE_EXCL: int = MODE_EXCL
  Constant MODE_EXCL of type int

mpi4py.MPI.MODE_DELETE_ON_CLOSE

mpi4py.MPI.MODE_DELETE_ON_CLOSE: int = MODE_DELETE_ON_CLOSE
  Constant MODE_DELETE_ON_CLOSE of type int

mpi4py.MPI.MODE_UNIQUE_OPEN

mpi4py.MPI.MODE_UNIQUE_OPEN: int = MODE_UNIQUE_OPEN
  Constant MODE_UNIQUE_OPEN of type int

mpi4py.MPI.MODE_SEQUENTIAL

mpi4py.MPI.MODE_SEQUENTIAL: int = MODE_SEQUENTIAL
  Constant MODE_SEQUENTIAL of type int

mpi4py.MPI.MODE_APPEND

mpi4py.MPI.MODE_APPEND: int = MODE_APPEND
  Constant MODE_APPEND of type int
mpi4py.MPI.SEEK_SET

mpi4py.MPI.SEEK_SET: int = SEEK_SET
    Constant SEEK_SET of type int

mpi4py.MPI.SEEK_CUR

mpi4py.MPI.SEEK_CUR: int = SEEK_CUR
    Constant SEEK_CUR of type int

mpi4py.MPI.SEEK_END

mpi4py.MPI.SEEK_END: int = SEEK_END
    Constant SEEK_END of type int

mpi4py.MPI.DISPLACEMENT_CURRENT

mpi4py.MPI.DISPLACEMENT_CURRENT: int = DISPLACEMENT_CURRENT
    Constant DISPLACEMENT_CURRENT of type int

mpi4py.MPI.DISP_CUR

mpi4py.MPI.DISP_CUR: int = DISP_CUR
    Constant DISP_CUR of type int

mpi4py.MPI.THREAD_SINGLE

mpi4py.MPI.THREAD_SINGLE: int = THREAD_SINGLE
    Constant THREAD_SINGLE of type int

mpi4py.MPI.THREAD_FUNNELED

mpi4py.MPI.THREAD_FUNNELED: int = THREAD_FUNNELED
    Constant THREAD_FUNNELED of type int

mpi4py.MPI.THREAD_SERIALIZED

mpi4py.MPI.THREAD_SERIALIZED: int = THREAD_SERIALIZED
    Constant THREAD_SERIALIZED of type int
mpi4py.MPI.THREAD_MULTIPLE

mpi4py.MPI.THREAD_MULTIPLE: int = THREAD_MULTIPLE
    Constant THREAD_MULTIPLE of type int

mpi4py.MPI.VERSION

mpi4py.MPI.VERSION: int = VERSION
    Constant VERSION of type int

mpi4py.MPI.SUBVERSION

mpi4py.MPI.SUBVERSION: int = SUBVERSION
    Constant SUBVERSION of type int

mpi4py.MPI.MAX_PROCESSOR_NAME

mpi4py.MPI.MAX_PROCESSOR_NAME: int = MAX_PROCESSOR_NAME
    Constant MAX_PROCESSOR_NAME of type int

mpi4py.MPI.MAX_ERROR_STRING

mpi4py.MPI.MAX_ERROR_STRING: int = MAX_ERROR_STRING
    Constant MAX_ERROR_STRING of type int

mpi4py.MPI.MAX_PORT_NAME

mpi4py.MPI.MAX_PORT_NAME: int = MAX_PORT_NAME
    Constant MAX_PORT_NAME of type int

mpi4py.MPI.MAX_INFO_KEY

mpi4py.MPI.MAX_INFO_KEY: int = MAX_INFO_KEY
    Constant MAX_INFO_KEY of type int

mpi4py.MPI.MAX_INFO_VAL

mpi4py.MPI.MAX_INFO_VAL: int = MAX_INFO_VAL
    Constant MAX_INFO_VAL of type int
mpi4py.MPI.MAX_OBJECT_NAME

mpi4py.MPI.MAX_OBJECT_NAME: int = MAX_OBJECT_NAME
    Constant MAX_OBJECT_NAME of type int

mpi4py.MPI.MAX_DATAREP_STRING

mpi4py.MPI.MAX_DATAREP_STRING: int = MAX_DATAREP_STRING
    Constant MAX_DATAREP_STRING of type int

mpi4py.MPI.MAX_LIBRARY_VERSION_STRING

mpi4py.MPI.MAX_LIBRARY_VERSION_STRING: int = MAX_LIBRARY_VERSION_STRING
    Constant MAX_LIBRARY_VERSION_STRING of type int

mpi4py.MPI.MAX_PSET_NAME_LEN

mpi4py.MPI.MAX_PSET_NAME_LEN: int = MAX_PSET_NAME_LEN
    Constant MAX_PSET_NAME_LEN of type int

mpi4py.MPI.MAX_STRINGTAG_LEN

mpi4py.MPI.MAX_STRINGTAG_LEN: int = MAX_STRINGTAG_LEN
    Constant MAX_STRINGTAG_LEN of type int

mpi4py.MPI.DATATYPE_NULL

mpi4py.MPI.DATATYPE_NULL: Datatype = DATATYPE_NULL
    Object DATATYPE_NULL of type Datatype

mpi4py.MPI.PACKED

mpi4py.MPI.PACKED: Datatype = PACKED
    Object PACKED of type Datatype

mpi4py.MPI.BYTE

mpi4py.MPI.BYTE: Datatype = BYTE
    Object BYTE of type Datatype
mpi4py.MPI.AINT

mpi4py.MPI.AINT: Datatype = AINT
Object AINT of type Datatype

mpi4py.MPI.OFFSET

mpi4py.MPI.OFFSET: Datatype = OFFSET
Object OFFSET of type Datatype

mpi4py.MPI.COUNT

mpi4py.MPI.COUNT: Datatype = COUNT
Object COUNT of type Datatype

mpi4py.MPI.CHAR

mpi4py.MPI.CHAR: Datatype = CHAR
Object CHAR of type Datatype

mpi4py.MPI.WCHAR

mpi4py.MPI.WCHAR: Datatype = WCHAR
Object WCHAR of type Datatype

mpi4py.MPI.SIGNED_CHAR

mpi4py.MPI.SIGNED_CHAR: Datatype = SIGNED_CHAR
Object SIGNED_CHAR of type Datatype

mpi4py.MPI.SHORT

mpi4py.MPI.SHORT: Datatype = SHORT
Object SHORT of type Datatype

mpi4py.MPI.INT

mpi4py.MPI.INT: Datatype = INT
Object INT of type Datatype
mpi4py.MPI.LONG

mpi4py.MPI.LONG: Datatype = LONG
Object LONG of type Datatype

mpi4py.MPI.LONG_LONG

mpi4py.MPI.LONG_LONG: Datatype = LONG_LONG
Object LONG_LONG of type Datatype

mpi4py.MPI.UNSIGNED_CHAR

mpi4py.MPI.UNSIGNED_CHAR: Datatype = UNSIGNED_CHAR
Object UNSIGNED_CHAR of type Datatype

mpi4py.MPI.UNSIGNED_SHORT

mpi4py.MPI.UNSIGNED_SHORT: Datatype = UNSIGNED_SHORT
Object UNSIGNED_SHORT of type Datatype

mpi4py.MPI.UNSIGNED

mpi4py.MPI.UNSIGNED: Datatype = UNSIGNED
Object UNSIGNED of type Datatype

mpi4py.MPI.UNSIGNED_LONG

mpi4py.MPI.UNSIGNED_LONG: Datatype = UNSIGNED_LONG
Object UNSIGNED_LONG of type Datatype

mpi4py.MPI.UNSIGNED_LONG_LONG

mpi4py.MPI.UNSIGNED_LONG_LONG: Datatype = UNSIGNED_LONG_LONG
Object UNSIGNED_LONG_LONG of type Datatype

mpi4py.MPI.FLOAT

mpi4py.MPI.FLOAT: Datatype = FLOAT
Object FLOAT of type Datatype
mpi4py.MPI.DOUBLE

mpi4py.MPI.DOUBLE: Datatype = DOUBLE
    Object DOUBLE of type Datatype

mpi4py.MPI.LONGDOUBLE

mpi4py.MPI.LONGDOUBLE: Datatype = LONG_DOUBLE
    Object LONG_DOUBLE of type Datatype

mpi4py.MPI.C_BOOL

mpi4py.MPI.C_BOOL: Datatype = C_BOOL
    Object C_BOOL of type Datatype

mpi4py.MPI.INT8_T

mpi4py.MPI.INT8_T: Datatype = INT8_T
    Object INT8_T of type Datatype

mpi4py.MPI.INT16_T

mpi4py.MPI.INT16_T: Datatype = INT16_T
    Object INT16_T of type Datatype

mpi4py.MPI.INT32_T

mpi4py.MPI.INT32_T: Datatype = INT32_T
    Object INT32_T of type Datatype

mpi4py.MPI.INT64_T

mpi4py.MPI.INT64_T: Datatype = INT64_T
    Object INT64_T of type Datatype

mpi4py.MPI.UINT8_T

mpi4py.MPI.UINT8_T: Datatype = UINT8_T
    Object UINT8_T of type Datatype
mpi4py.MPI.UINT16_T

mpi4py.MPI.UINT16_T: Datatype = UINT16_T
   Object UINT16_T of type Datatype

mpi4py.MPI.UINT32_T

mpi4py.MPI.UINT32_T: Datatype = UINT32_T
   Object UINT32_T of type Datatype

mpi4py.MPI.UINT64_T

mpi4py.MPI.UINT64_T: Datatype = UINT64_T
   Object UINT64_T of type Datatype

mpi4py.MPI.C_COMPLEX

mpi4py.MPI.C_COMPLEX: Datatype = C_COMPLEX
   Object C_COMPLEX of type Datatype

mpi4py.MPI.C_FLOAT_COMPLEX

mpi4py.MPI.C_FLOAT_COMPLEX: Datatype = C_FLOAT_COMPLEX
   Object C_FLOAT_COMPLEX of type Datatype

mpi4py.MPI.C_DOUBLE_COMPLEX

mpi4py.MPI.C_DOUBLE_COMPLEX: Datatype = C_DOUBLE_COMPLEX
   Object C_DOUBLE_COMPLEX of type Datatype

mpi4py.MPI.C_LONG_DOUBLE_COMPLEX

mpi4py.MPI.C_LONG_DOUBLE_COMPLEX: Datatype = C_LONG_DOUBLE_COMPLEX
   Object C_LONG_DOUBLE_COMPLEX of type Datatype

mpi4py.MPI.CXX_BOOL

mpi4py.MPI.CXX_BOOL: Datatype = CXX_BOOL
   Object CXX_BOOL of type Datatype
mpi4py.MPI.CXX_FLOAT_COMPLEX

mpi4py.MPI.CXX_FLOAT_COMPLEX: Datatype = CXX_FLOAT_COMPLEX
Object CXX_FLOAT_COMPLEX of type Datatype

mpi4py.MPI.CXX_DOUBLE_COMPLEX

mpi4py.MPI.CXX_DOUBLE_COMPLEX: Datatype = CXX_DOUBLE_COMPLEX
Object CXX_DOUBLE_COMPLEX of type Datatype

mpi4py.MPI.CXX_LONG_DOUBLE_COMPLEX

mpi4py.MPI.CXX_LONG_DOUBLE_COMPLEX: Datatype = CXX_LONG_DOUBLE_COMPLEX
Object CXX_LONG_DOUBLE_COMPLEX of type Datatype

mpi4py.MPI.SHORT_INT

mpi4py.MPI.SHORT_INT: Datatype = SHORT_INT
Object SHORT_INT of type Datatype

mpi4py.MPI.INT_INT

mpi4py.MPI.INT_INT: Datatype = INT_INT
Object INT_INT of type Datatype

mpi4py.MPI.TWOINT

mpi4py.MPI.TWOINT: Datatype = TWOINT
Object TWOINT of type Datatype

mpi4py.MPI.LONG_INT

mpi4py.MPI.LONG_INT: Datatype = LONG_INT
Object LONG_INT of type Datatype

mpi4py.MPI.FLOAT_INT

mpi4py.MPI.FLOAT_INT: Datatype = FLOAT_INT
Object FLOAT_INT of type Datatype
mpi4py.MPI.DOUBLE_INT

mpi4py.MPI.DOUBLE_INT: Datatype = DOUBLE_INT
    Object DOUBLE_INT of type Datatype

mpi4py.MPI.LONG_DOUBLE_INT

mpi4py.MPI.LONG_DOUBLE_INT: Datatype = LONG_DOUBLE_INT
    Object LONG_DOUBLE_INT of type Datatype

mpi4py.MPI.CHARACTER

mpi4py.MPI.CHARACTER: Datatype = CHARACTER
    Object CHARACTER of type Datatype

mpi4py.MPI.LOGICAL

mpi4py.MPI.LOGICAL: Datatype = LOGICAL
    Object LOGICAL of type Datatype

mpi4py.MPI.INTEGER

mpi4py.MPI.INTEGER: Datatype = INTEGER
    Object INTEGER of type Datatype

mpi4py.MPI.REAL

mpi4py.MPI.REAL: Datatype = REAL
    Object REAL of type Datatype

mpi4py.MPI.DOUBLE_PRECISION

mpi4py.MPI.DOUBLE_PRECISION: Datatype = DOUBLE_PRECISION
    Object DOUBLE_PRECISION of type Datatype

mpi4py.MPI.COMPLEX

mpi4py.MPI.COMPLEX: Datatype = COMPLEX
    Object COMPLEX of type Datatype
mpi4py.MPI.DOUBLE_COMPLEX

mpi4py.MPI.DOUBLE_COMPLEX: Datatype = DOUBLE_COMPLEX
Object DOUBLE_COMPLEX of type Datatype

mpi4py.MPI.LOGICAL1

mpi4py.MPI.LOGICAL1: Datatype = LOGICAL1
Object LOGICAL1 of type Datatype

mpi4py.MPI.LOGICAL2

mpi4py.MPI.LOGICAL2: Datatype = LOGICAL2
Object LOGICAL2 of type Datatype

mpi4py.MPI.LOGICAL4

mpi4py.MPI.LOGICAL4: Datatype = LOGICAL4
Object LOGICAL4 of type Datatype

mpi4py.MPI.LOGICAL8

mpi4py.MPI.LOGICAL8: Datatype = LOGICAL8
Object LOGICAL8 of type Datatype

mpi4py.MPI.INTEGER1

mpi4py.MPI.INTEGER1: Datatype = INTEGER1
Object INTEGER1 of type Datatype

mpi4py.MPI.INTEGER2

mpi4py.MPI.INTEGER2: Datatype = INTEGER2
Object INTEGER2 of type Datatype

mpi4py.MPI.INTEGER4

mpi4py.MPI.INTEGER4: Datatype = INTEGER4
Object INTEGER4 of type Datatype
mpi4py.MPI.INTEGER8

mpi4py.MPI.INTEGER8:  Datatype = INTEGER8
  Object INTEGER8 of type Datatype

mpi4py.MPI.INTEGER16

mpi4py.MPI.INTEGER16:  Datatype = INTEGER16
  Object INTEGER16 of type Datatype

mpi4py.MPI-REAL2

mpi4py.MPI-REAL2:  Datatype = REAL2
  Object REAL2 of type Datatype

mpi4py.MPI-REAL4

mpi4py.MPI-REAL4:  Datatype = REAL4
  Object REAL4 of type Datatype

mpi4py.MPI-REAL8

mpi4py.MPI-REAL8:  Datatype = REAL8
  Object REAL8 of type Datatype

mpi4py.MPI-REAL16

mpi4py.MPI-REAL16:  Datatype = REAL16
  Object REAL16 of type Datatype

mpi4py.MPI.COMPLEX4

mpi4py.MPI.COMPLEX4:  Datatype = COMPLEX4
  Object COMPLEX4 of type Datatype

mpi4py.MPI.COMPLEX8

mpi4py.MPI.COMPLEX8:  Datatype = COMPLEX8
  Object COMPLEX8 of type Datatype
mpi4py.MPI.COMPLEX16

mpi4py.MPI.COMPLEX16: Datatype = COMPLEX16
Object COMPLEX16 of type Datatype

mpi4py.MPI.COMPLEX32

mpi4py.MPI.COMPLEX32: Datatype = COMPLEX32
Object COMPLEX32 of type Datatype

mpi4py.MPI.UNSIGNED_INT

mpi4py.MPI.UNSIGNED_INT: Datatype = UNSIGNED_INT
Object UNSIGNED_INT of type Datatype

mpi4py.MPI.SIGNED_SHORT

mpi4py.MPI.SIGNED_SHORT: Datatype = SIGNED_SHORT
Object SIGNED_SHORT of type Datatype

mpi4py.MPI.SIGNED_INT

mpi4py.MPI.SIGNED_INT: Datatype = SIGNED_INT
Object SIGNED_INT of type Datatype

mpi4py.MPI.SIGNED_LONG

mpi4py.MPI.SIGNED_LONG: Datatype = SIGNED_LONG
Object SIGNED_LONG of type Datatype

mpi4py.MPI.SIGNED_LONG_LONG

mpi4py.MPI.SIGNED_LONG_LONG: Datatype = SIGNED_LONG_LONG
Object SIGNED_LONG_LONG of type Datatype

mpi4py.MPI.BOOL

mpi4py.MPI.BOOL: Datatype = BOOL
Object BOOL of type Datatype
mpi4py.MPI.SINT8_T

mpi4py.MPI.SINT8_T: Datatype = SINT8_T
Object SINT8_T of type Datatype

mpi4py.MPI.SINT16_T

mpi4py.MPI.SINT16_T: Datatype = SINT16_T
Object SINT16_T of type Datatype

mpi4py.MPI.SINT32_T

mpi4py.MPI.SINT32_T: Datatype = SINT32_T
Object SINT32_T of type Datatype

mpi4py.MPI.SINT64_T

mpi4py.MPI.SINT64_T: Datatype = SINT64_T
Object SINT64_T of type Datatype

mpi4py.MPI.F_BOOL

mpi4py.MPI.F_BOOL: Datatype = F_BOOL
Object F_BOOL of type Datatype

mpi4py.MPI.F_INT

mpi4py.MPI.F_INT: Datatype = F_INT
Object F_INT of type Datatype

mpi4py.MPI.F_FLOAT

mpi4py.MPI.F_FLOAT: Datatype = F_FLOAT
Object F_FLOAT of type Datatype

mpi4py.MPI.F_DOUBLE

mpi4py.MPI.F_DOUBLE: Datatype = F_DOUBLE
Object F_DOUBLE of type Datatype
mpi4py.MPI.F_COMPLEX

mpi4py.MPI.F_COMPLEX: Datatype = F_COMPLEX
  Object F_COMPLEX of type Datatype

mpi4py.MPI.F_FLOAT_COMPLEX

mpi4py.MPI.F_FLOAT_COMPLEX: Datatype = F_FLOAT_COMPLEX
  Object F_FLOAT_COMPLEX of type Datatype

mpi4py.MPI.F_DOUBLE_COMPLEX

mpi4py.MPI.F_DOUBLE_COMPLEX: Datatype = F_DOUBLE_COMPLEX
  Object F_DOUBLE_COMPLEX of type Datatype

mpi4py.MPI.REQUEST_NULL

mpi4py.MPI.REQUEST_NULL: Request = REQUEST_NULL
  Object REQUEST_NULL of type Request

mpi4py.MPI.MESSAGE_NULL

mpi4py.MPI.MESSAGE_NULL: Message = MESSAGE_NULL
  Object MESSAGE_NULL of type Message

mpi4py.MPI.MESSAGE_NO_PROC

mpi4py.MPI.MESSAGE_NO_PROC: Message = MESSAGE_NO_PROC
  Object MESSAGE_NO_PROC of type Message

mpi4py.MPI.OP_NULL

mpi4py.MPI.OP_NULL: Op = OP_NULL
  Object OP_NULL of type Op
  Parameters
    • x (Any) –
    • y (Any) –
  Return type
    Any
mpi4py.MPI.MAX

mpi4py.MPI.MAX:  $Op = \text{MAX}$

Object MAX of type $Op$

Parameters

• $x$ (Any) –
• $y$ (Any) –

Return type

Any

mpi4py.MPI.MIN

mpi4py.MPI.MIN:  $Op = \text{MIN}$

Object MIN of type $Op$

Parameters

• $x$ (Any) –
• $y$ (Any) –

Return type

Any

mpi4py.MPI.SUM

mpi4py.MPI.SUM:  $Op = \text{SUM}$

Object SUM of type $Op$

Parameters

• $x$ (Any) –
• $y$ (Any) –

Return type

Any

mpi4py.MPI.PROD

mpi4py.MPI.PROD:  $Op = \text{PROD}$

Object PROD of type $Op$

Parameters

• $x$ (Any) –
• $y$ (Any) –

Return type

Any
mpi4py.MPI.LAND

mpi4py.MPI.LAND: \( Op = \text{LAND} \)

Object \text{LAND} of type \( Op \)

Parameters

• \( x \) (Any) –
• \( y \) (Any) –

Return type

Any

mpi4py.MPI.BAND

mpi4py.MPI.BAND: \( Op = \text{BAND} \)

Object \text{BAND} of type \( Op \)

Parameters

• \( x \) (Any) –
• \( y \) (Any) –

Return type

Any

mpi4py.MPI.LOR

mpi4py.MPI.LOR: \( Op = \text{LOR} \)

Object \text{LOR} of type \( Op \)

Parameters

• \( x \) (Any) –
• \( y \) (Any) –

Return type

Any

mpi4py.MPI.BOR

mpi4py.MPI.BOR: \( Op = \text{BOR} \)

Object \text{BOR} of type \( Op \)

Parameters

• \( x \) (Any) –
• \( y \) (Any) –

Return type

Any
mpi4py.MPI.LXOR

mpi4py.MPI.LXOR: \( Op = LXOR \)
Object LXOR of type \( Op \)

Parameters
• \( x (\text{Any}) \)
• \( y (\text{Any}) \)

Return type
\( \text{Any} \)

mpi4py.MPI.BXOR

mpi4py.MPI.BXOR: \( Op = BXOR \)
Object BXOR of type \( Op \)

Parameters
• \( x (\text{Any}) \)
• \( y (\text{Any}) \)

Return type
\( \text{Any} \)

mpi4py.MPI.MAXLOC

mpi4py.MPI.MAXLOC: \( Op = MAXLOC \)
Object MAXLOC of type \( Op \)

Parameters
• \( x (\text{Any}) \)
• \( y (\text{Any}) \)

Return type
\( \text{Any} \)

mpi4py.MPI.MINLOC

mpi4py.MPI.MINLOC: \( Op = MINLOC \)
Object MINLOC of type \( Op \)

Parameters
• \( x (\text{Any}) \)
• \( y (\text{Any}) \)

Return type
\( \text{Any} \)
mpi4py.MPI.REPLACE

mpi4py.MPI.REPLACE: \( Op = \text{REPLACE} \)
Object REPLACE of type \( Op \)

Parameters
  • \( x \) (Any) –
  • \( y \) (Any) –

Return type
  Any

mpi4py.MPI.NO_OP

mpi4py.MPI.NO_OP: \( Op = \text{NO}_{\text{OP}} \)
Object \( \text{NO}_{\text{OP}} \) of type \( Op \)

Parameters
  • \( x \) (Any) –
  • \( y \) (Any) –

Return type
  Any

mpi4py.MPI.GROUP_NULL

mpi4py.MPI.GROUP_NULL: \( Group = \text{GROUP}_{\text{NULL}} \)
Object \( \text{GROUP}_{\text{NULL}} \) of type \( Group \)

mpi4py.MPI.GROUP_EMPTY

mpi4py.MPI.GROUP_EMPTY: \( Group = \text{GROUP}_{\text{EMPTY}} \)
Object \( \text{GROUP}_{\text{EMPTY}} \) of type \( Group \)

mpi4py.MPI.INFO_NULL

mpi4py.MPI.INFO_NULL: \( Info = \text{INFO}_{\text{NULL}} \)
Object \( \text{INFO}_{\text{NULL}} \) of type \( Info \)

mpi4py.MPI.INFO_ENV

mpi4py.MPI.INFO_ENV: \( Info = \text{INFO}_{\text{ENV}} \)
Object \( \text{INFO}_{\text{ENV}} \) of type \( Info \)
mpi4py.MPI.ERRHANDLER_NULL

mpi4py.MPI.ERRHANDLER_NULL: `Errhandler = ERRHANDLER_NULL`
Object ERRHANDLER_NULL of type `Errhandler`

mpi4py.MPI.ERRORS_RETURN

mpi4py.MPI.ERRORS_RETURN: `Errhandler = ERRORS_RETURN`
Object ERRORS_RETURN of type `Errhandler`

mpi4py.MPI.ERRORS_ABORT

mpi4py.MPI.ERRORS_ABORT: `Errhandler = ERRORS_ABORT`
Object ERRORS_ABORT of type `Errhandler`

mpi4py.MPI.ERRORS_ARE_FATAL

mpi4py.MPI.ERRORS_ARE_FATAL: `Errhandler = ERRORS_ARE_FATAL`
Object ERRORS_ARE_FATAL of type `Errhandler`

mpi4py.MPI.SESSION_NULL

mpi4py.MPI.SESSION_NULL: `Session = SESSION_NULL`
Object SESSION_NULL of type `Session`

mpi4py.MPI.COMM_NULL

mpi4py.MPI.COMM_NULL: `Comm = COMM_NULL`
Object COMM_NULL of type `Comm`

mpi4py.MPI.COMM_SELF

mpi4py.MPI.COMM_SELF: `Intracomm = COMM_SELF`
Object COMM_SELF of type `Intracomm`

mpi4py.MPI.COMM_WORLD

mpi4py.MPI.COMM_WORLD: `Intracomm = COMM_WORLD`
Object COMM_WORLD of type `Intracomm`
mpi4py.MPI.WIN_NULL

mpi4py.MPI.WIN_NULL: \( \textbf{Win} = \text{WIN\_NULL} \)
Object \text{WIN\_NULL} of type \text{Win}

mpi4py.MPI.FILE_NULL

mpi4py.MPI.FILE_NULL: \( \textbf{File} = \text{FILE\_NULL} \)
Object \text{FILE\_NULL} of type \text{File}

mpi4py.MPI.pickle

mpi4py.MPI.pickle: \textbf{Pickle} = \text{<mpi4py.MPI.Pickle object>}
Object \text{pickle} of type \text{Pickle}

12 Citation

If MPI for Python been significant to a project that leads to an academic publication, please acknowledge that fact by citing the project.


13 Installation

13.1 Build backends

mpi4py supports two different build backends: setuptools (default), scikit-build-core (CMake-based), and meson-python (Meson-based). The build backend can be selected by setting the \texttt{MPI4PY\_BUILD\_BACKEND} environment variable.

\texttt{MPI4PY\_BUILD\_BACKEND}

- Choices: 
  - "setuptools", "scikit-build-core", "meson-python"
- Default: 
  - "setuptools"

Request a build backend for building mpi4py from sources.
Using setuptools

Tip: Set the MPI4PY_BUILD_BACKEND environment variable to "setuptools" to use the setuptools build backend.

When using the default setuptools build backend, mpi4py relies on the legacy Python distutils framework to build C extension modules. The following environment variables affect the build configuration.

**MPI4PY_BUILD_MPICC**

The mpicc compiler wrapper command is searched for in the executable search path (PATH environment variable) and used to compile the mpi4py.MPI C extension module. Alternatively, use the MPI4PY_BUILD_MPICC environment variable to the full path or command corresponding to the MPI-aware C compiler.

**MPI4PY_BUILD_MPILD**

The mpicc compiler wrapper command is also used for linking the mpi4py.MPI C extension module. Alternatively, use the MPI4PY_BUILD_MPILD environment variable to specify the full path or command corresponding to the MPI-aware C linker.

**MPI4PY_BUILD_MPICFG**

If the MPI implementation does not provide a compiler wrapper, or it is not installed in a default system location, all relevant build information like include/library locations and library lists can be provided in an ini-style configuration file under a [mpi] section. mpi4py can then be asked to use the custom build information by setting the MPI4PY_BUILD_MPICFG environment variable to the full path of the configuration file. As an example, see the mpi.cfg file located in the top level mpi4py source directory.

**MPI4PY_BUILD_CONFIGURE**

Some vendor MPI implementations may not provide complete coverage of the MPI standard, or may provide partial features of newer MPI standard versions while advertising support for an older version. Setting the MPI4PY_BUILD_CONFIGURE environment variable to a non-empty string will trigger the run of exhaustive checks for the availability of all MPI constants, predefined handles, and routines.

The following environment variables are aliases for the ones described above. Having shorter names, they are convenient for occasional use in the command line. Its usage is not recommended in automation scenarios like packaging recipes, deployment scripts, and container image creation.

**MPICC**

Convenience alias for MPI4PY_BUILD_MPICC.

**MPILD**

Convenience alias for MPI4PY_BUILD_MPILD.

**MPICFG**

Convenience alias for MPI4PY_BUILD_MPICFG.

Using scikit-build-core

Tip: Set the MPI4PY_BUILD_BACKEND environment variable to "scikit-build-core" to use the scikit-build-core build backend.

When using the scikit-build-core build backend, mpi4py delegates all of MPI build configuration to CMake’s FindMPI module. Besides the obvious advantage of cross-platform support, this delegation to CMake may be convenient in build environments exposing vendor software stacks via intricate module systems. Note however that mpi4py will not be able to look for MPI routines available beyond the MPI standard version the MPI implementation advertises...
to support (via the `MPI_VERSION` and `MPI_SUBVERSION` macro constants in the `mpi.h` header file), any missing MPI constant or symbol will prevent a successful build.

**Using meson-python**

**Tip:** Set the `MPI4PY_BUILD_BACKEND` environment variable to "meson-python" to use the meson-python build backend.

When using the meson-python build backend, mpi4py delegates build tasks to the Meson build system.

**Warning:** mpi4py support for the meson-python build backend is experimental. For the time being, users must set the `CC` environment variable to the command or path corresponding to the `mpicc` C compiler wrapper.

13.2 Using pip

You can install mpi4py from its source distribution using pip:

```
$ python -m pip install mpi4py
```

You can also install the in-development version with:

```
$ python -m pip install git+https://github.com/mpi4py/mpi4py
```

or:

```
$ python -m pip install https://github.com/mpi4py/mpi4py/tarball/master
```

**Note:** Installing mpi4py from sources requires a C compiler and an MPI implementation with development headers and libraries.

**Warning:** pip keeps previously built wheel files on its cache for future reuse. If you want to reinstall the mpi4py package using a different or updated MPI implementation, you have to either first remove the cached wheel file with:

```
$ python -m pip cache remove mpi4py
```

or ask pip to disable the cache:

```
$ python -m pip install --no-cache-dir mpi4py
```
13.3 Using conda

The conda-forge community provides ready-to-use binary packages from an ever-growing collection of software libraries built around the multi-platform conda package manager. Three MPI implementations are available on conda-forge: Open MPI (Linux and macOS), MPICH (Linux and macOS), and Microsoft MPI (Windows). You can install mpi4py and your preferred MPI implementation using the conda package manager:

- to use MPICH do:
  
  $ conda install -c conda-forge mpi4py mpich

- to use Open MPI do:
  
  $ conda install -c conda-forge mpi4py openmpi

- to use Microsoft MPI do:
  
  $ conda install -c conda-forge mpi4py msmpi

MPICH and many of its derivatives are ABI-compatible. You can provide the package specification `mpich=X.Y.*=external_*` (where `X` and `Y` are the major and minor version numbers) to request the conda package manager to use system-provided MPICH (or derivative) libraries. Similarly, you can provide the package specification `openmpi=X.Y.*=external_*` to use system-provided Open MPI libraries.

The `openmpi` package on conda-forge has built-in CUDA support, but it is disabled by default. To enable it, follow the instruction outlined during `conda install`. Additionally, UCX support is also available once the `ucx` package is installed.

**Warning:** Binary conda-forge packages are built with a focus on compatibility. The MPICH and Open MPI packages are build in a constrained environment with relatively dated OS images. Therefore, they may lack support for high-performance features like cross-memory attach (XPMEM/CMA). In production scenarios, it is recommended to use external (either custom-built or system-provided) MPI installations. See the relevant conda-forge documentation about using external MPI libraries.

13.4 Linux

On Fedora Linux systems (as well as RHEL and their derivatives using the EPEL software repository), you can install binary packages with the system package manager:

- using dnf and the `mpich` package:
  
  $ sudo dnf install python3-mpi4py-mpich

- using dnf and the `openmpi` package:
  
  $ sudo dnf install python3-mpi4py-openmpi

Please remember to load the correct MPI module for your chosen MPI implementation:

- for the `mpich` package do:
  
  $ module load mpi/mpich-$(arch)
  $ python -c "from mpi4py import MPI"

- for the `openmpi` package do:
On Ubuntu Linux and Debian Linux systems, binary packages are available for installation using the system package manager:

```
$ sudo apt install python3-mpi4py
```

Note that on Ubuntu/Debian systems, the mpi4py package uses Open MPI. To use MPICH, install the *libmpich-dev* and *python3-dev* packages (and any other required development tools). Afterwards, install mpi4py from sources using *pip*.

### 13.5 macOS

macOS users can install mpi4py using the Homebrew package manager:

```
$ brew install mpi4py
```

Note that the Homebrew mpi4py package uses Open MPI. Alternatively, install the *mpich* package and next install mpi4py from sources using *pip*.

### 13.6 Windows

Windows users can install mpi4py from binary wheels hosted on the Python Package Index (PyPI) using *pip*:

```
$ python -m pip install mpi4py
```

Windows wheels require a separate, system-wide installation of the Microsoft MPI runtime package.

### 14 Development

#### 14.1 Prerequisites

You need to have the following software properly installed in order to build *MPI for Python*:

- Python 3.6 or above.
- The Cython compiler.
- A working MPI implementation like MPICH or Open MPI, preferably supporting MPI-4 and built with shared/dynamic libraries.

**Note:** If you want to build some MPI implementation from sources, check the instructions at *Building MPI from sources* in the appendix.

**Note:** Some MPI-1 implementations do require the actual command line arguments to be passed in *MPI_Init()*.* In this case, you will need to use a rebuilt, MPI-enabled, Python interpreter executable. *MPI for Python* has some support for alleviating you from this task. Check the instructions at *MPI-enabled Python interpreter* in the appendix.
Optionally, consider installing the following packages:

- **NumPy** for enabling comprehensive testing of MPI communication.
- **CuPy** for enabling comprehensive testing with a GPU-aware MPI.
- **Sphinx** to build documentation.

### 14.2 Building

*MPI for Python* uses the `setuptools`-based build system that relies on the `setup.py` file. Some `setuptools` commands (e.g., `build`) accept additional options:

---

`--mpi=`

- Lets you pass a section with MPI configuration within a special configuration file. Alternatively, you can use the `MPICFG` environment variable.

`--mpicc=`

- Specify the path or name of the `mpicc` C compiler wrapper. Alternatively, use the `MPICC` environment variable.

`--mpild=`

- Specify the full path or name for the MPI-aware C linker. Alternatively, use the `MPILD` environment variable. If not set, the `mpicc` C compiler wrapper is used for linking.

`--configure`

- Runs exhaustive tests for checking about missing MPI types, constants, and functions. This option should be passed in order to build *MPI for Python* against old MPI-1, MPI-2, or MPI-3 implementations, possibly providing a subset of MPI-4.

If you use a MPI implementation providing a `mpicc` C compiler wrapper (e.g., MPICH or Open MPI), it will be used for compilation and linking. This is the preferred and easiest way to build *MPI for Python*.

If `mpicc` is found in the executable search path (PATH environment variable), simply run the `build` command:

```
$ python setup.py build
```

If `mpicc` is not in your search path or the compiler wrapper has a different name, you can run the `build` command specifying its location, either via the `--mpicc` command option or using the `MPICC` environment variable:

```
$ python setup.py build --mpicc=/path/to/mpicc
$ MPICC=/path/to/mpicc python setup.py build
```

Alternatively, you can provide all the relevant information about your MPI implementation by editing the `mpi.cfg` file located in the top level source directory. You can use the default section `[mpi]` or add a new custom section, for example `[other_mpi]` (see the examples provided in the `mpi.cfg` file as a starting point to write your own section):

```
[mypi]
include_dirs   = /usr/local/mpi/include
libraries      = mpi
library_dirs   = /usr/local/mpi/lib
runtime_library_dirs = /usr/local/mpi/lib

[other_mpi]
include_dirs   = /opt/mpi/include ...
libraries      = mpi ...
library_dirs   = /opt/mpi/lib ...
```

(continues on next page)
and then run the *build* command specifying your custom configuration section:

```bash
$ python setup.py build --mpi=other_mpi
$ MPICFG=other_mpi python setup.py build
```

After building, the package is ready for installation in development mode:

```bash
$ python setup.py develop --user
```

Alternatively, you can generate a binary wheel file in the *dist/* directory with:

```bash
$ python setup.py bdist_wheel
```

### 14.3 Testing

To quickly test the installation:

```bash
$ mpiexec -n 5 python -m mpi4py.bench helloworld
Hello, World! I am process 0 of 5 on localhost.
Hello, World! I am process 1 of 5 on localhost.
Hello, World! I am process 2 of 5 on localhost.
Hello, World! I am process 3 of 5 on localhost.
Hello, World! I am process 4 of 5 on localhost.
```

```bash
$ mpiexec -n 5 python -m mpi4py.bench ringtest -l 10 -n 1048576
time for 10 loops = 0.00361614 seconds (5 processes, 1048576 bytes)
```

If you installed from a git clone or the source distribution, issuing at the command line:

```bash
$ mpiexec -n 5 python demo/helloworld.py
```

will launch a five-process run of the Python interpreter and run the test script *demo/helloworld.py* from the source distribution.

You can also run all the *unittest* scripts:

```bash
$ mpiexec -n 5 python test/runtests.py
```

or, if you have *nose* unit testing framework installed:

```bash
$ mpiexec -n 5 nosetests
```

or, if you have *py.test* unit testing framework installed:

```bash
$ mpiexec -n 5 pytest
```
15 Appendix

15.1 MPI-enabled Python interpreter

**Warning:** These days it is no longer required to use the MPI-enabled Python interpreter in most cases, and, therefore, it is not built by default anymore because it is too difficult to reliably build a Python interpreter across different distributions. If you know that you still **really** need it, see below on how to use the `build_exe` and `install_exe` commands.

Some MPI-1 implementations (notably, MPICH 1) do require the actual command line arguments to be passed at the time `MPI_Init()` is called. In this case, you will need to use a re-built, MPI-enabled, Python interpreter binary executable. A basic implementation (targeting Python 3.9) of what is required is shown below:

```c
#include <Python.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    int status, flag;
    MPI_Init(&argc, &argv);
    status = Py_BytesMain(argc, argv);
    MPI_Finalized(&flag);
    if (!flag) MPI_Finalize();
    return status;
}
```

The source code above is straightforward; compiling it should also be. However, the linking step is more tricky: special flags have to be passed to the linker depending on your platform. In order to alleviate you for such low-level details, *MPI for Python* provides some pure-distutils based support to build and install an MPI-enabled Python interpreter executable:

```
$ cd mpi4py-X.X.X
$ python setup.py build_exe [--mpi=<name>|--mpicc=/path/to/mpicc]
$ [sudo] python setup.py install_exe [--install-dir=$HOME/bin]
```

After the above steps you should have the MPI-enabled interpreter installed as `prefix/bin/pythonX.X-mpi` (or `$HOME/bin/pythonX.X-mpi`). Assuming that `prefix/bin` (or `$HOME/bin`) is listed on your `PATH`, you should be able to enter your MPI-enabled Python interactively, for example:

```
$ python3.9-mpi
Python 3.9.6 (default, Jul 16 2021, 00:00:00)
[GCC 11.1.1 20210531 (Red Hat 11.1.1-3)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.executable
'/usr/local/bin/python3.9-mpi'
```
15.2 Building MPI from sources

In the list below you have some executive instructions for building some of the open-source MPI implementations out there with support for shared/dynamic libraries on POSIX environments.

- **MPICH**
  
  ```
  $ tar -zxf mpich-X.X.X.tar.gz
  $ cd mpich-X.X.X
  $ ./configure --enable-shared --prefix=/usr/local/mpich
  $ make
  $ make install
  ```

- **Open MPI**
  
  ```
  $ tar -zxf openmpi-X.X.tar.gz
  $ cd openmpi-X.X
  $ ./configure --prefix=/usr/local/openmpi
  $ make all
  $ make install
  ```

- **MPICH 1**
  
  ```
  $ tar -zxf mpich-X.X.tar.gz
  $ cd mpich-X.X
  $ ./configure --enable-sharedlib --prefix=/usr/local/mpich1
  $ make
  $ make install
  ```

Perhaps you will need to set the `LD_LIBRARY_PATH` environment variable (using `export`, `setenv` or what applies to your system) pointing to the directory containing the MPI libraries. In case of getting runtime linking errors when running MPI programs, the following lines can be added to the user login shell script (`~/.profile`, `.bashrc`, etc.).

- **MPICH**
  
  ```
  MPI_DIR=/usr/local/mpich
  export LD_LIBRARY_PATH=$MPI_DIR/lib:$LD_LIBRARY_PATH
  ```

- **Open MPI**
  
  ```
  MPI_DIR=/usr/local/openmpi
  export LD_LIBRARY_PATH=$MPI_DIR/lib:$LD_LIBRARY_PATH
  ```

- **MPICH 1**
  
  ```
  MPI_DIR=/usr/local/mpich1
  export LD_LIBRARY_PATH=$MPI_DIR/lib/shared:$LD_LIBRARY_PATH:
  export MPICH_USE_SHLIB=yes
  ```

**Warning:** MPICH 1 support for dynamic libraries is not completely transparent. Users should set the environment variable `MPICH_USE_SHLIB` to `yes` in order to avoid link problems when using the `mpicc` compiler wrapper.
16 LICENSE

Copyright (c) 2023, Lisandro Dalcin.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

17 CHANGES

17.1 Release 4.0.0 [2023-XX-XX]

- New features:
  - Add support for the MPI-4.0 standard.
    - Use large count MPI-4 routines.
    - Add persistent collective communication.
    - Add partitioned point-to-point communication.
    - Add new communicator constructors.
    - Add the Session class and its methods.
  - Add preliminary support for the upcoming MPI-5.0 standard.
    - User-level failure mitigation (ULFM).
  - mpi4py.util.pool: New drop-in replacement for multiprocessing.pool.
  - Add runtime check for mismatch between mpiexec and MPI library.
  - Support scikit-build-core as an alternative build backend.
  - Support meson-python as an alternative build backend.

- Enhancements:
  - mpi4py.futures: Report exception tracebacks in workers.
  - mpi4py.util.pk15: Add support for collective communication.
  - Add methods Datatype.fromcode(), Datatype.tocode() and attributes Datatype.typestr, Datatype.typechar to simplify NumPy interoperability for simple cases.

- Add support for pickle serialization of instances of MPI types. All instances of `Datatype`, `Info`, and `Status` can be serialized. Instances of `Op` can be serialized only if created through `mpi4py` by calling `Op.Create()`. Instances of other MPI types can be serialized only if they reference predefined handles.

- **Backward-incompatible changes:**
  - Python 2 is no longer supported, Python 3.6+ is required, but typing stubs are supported for Python 3.7+.
  - The `Intracomm.Create_group()` method is no longer defined in the base `Comm` class.
  - `Group.Compare()` and `Comm.Compare()` are no longer class methods but instance methods. Existing codes using the former class methods are expected to continue working.
  - `Group.Translate_ranks()` is no longer a class method but a instance method. Existing codes using the former class method are expected to continue working.
  - The LB and UB datatypes are no longer available, use `Datatype.Create_resized()` instead.
  - The `mpi4py.dl` module is no longer available.

## 17.2 Release 3.1.4 [2022-11-02]

**Warning:** This is the last release supporting Python 2.

- Rebuild C sources with Cython 0.29.32 to support Python 3.11.
- Fix contiguity check for DLPack and CAI buffers.
- Workaround build failures with setuptools v60.

## 17.3 Release 3.1.3 [2021-11-25]

**Warning:** This is the last release supporting Python 2.

- Add missing support for `MPI.BOTTOM` to generalized all-to-all collectives.

## 17.4 Release 3.1.2 [2021-11-04]

**Warning:** This is the last release supporting Python 2.

- `mpi4py.futures`: Add `_max_workers` property to `MPIPoolExecutor`.
- `mpi4py.util.dtlib`: Fix computation of alignment for predefined datatypes.
- `mpi4py.util.pkl5`: Fix deadlock when using `ssend() + mprobe()`.
- `mpi4py.util.pkl5`: Add environment variable `MPI4PY_PICKLE_THRESHOLD`.
- `mpi4py.rc`: Interpret "y" and "n" strings as boolean values.
• Fix/add typemap/typestr for \texttt{MPI.WCHAR/MPI.COUNT} datatypes.
• Minor fixes and additions to documentation.
• Minor fixes to typing support.
• Support for local version identifier (PEP-440).

17.5 Release 3.1.1 [2021-08-14]

\begin{itemize}
\item Warning: This is the last release supporting Python 2.
\end{itemize}

• Fix typo in Requires-Python package metadata.
• Regenerate C sources with Cython 0.29.24.

17.6 Release 3.1.0 [2021-08-12]

\begin{itemize}
\item Warning: This is the last release supporting Python 2.
\end{itemize}

• New features:
  – \texttt{mpi4py.util}: New package collecting miscellaneous utilities.
• Enhancements:
  – Add pickle-based \texttt{Request.waitsome()} and \texttt{Request.testsome()}.  
  – Add lowercase methods \texttt{Request.get_status()} and \texttt{Request.cancel()}.
  – Support for passing Python GPU arrays compliant with the DLPack data interchange mechanism (link) and the \texttt{__cuda_array_interface__} (CAI) standard (link) to uppercase methods. This support requires that mpi4py is built against CUDA-aware MPI implementations. This feature is currently experimental and subject to future changes.
  – \texttt{mpi4py.futures}: Add support for initializers and canceling futures at shutdown. Environment variables names now follow the pattern \texttt{MPI4PY_FUTURES_*}, the previous \texttt{MPI4PY_*} names are deprecated.
  – Add type annotations to Cython code. The first line of the docstring of functions and methods displays a signature including type annotations.
  – Add companion stub files to support type checkers.
  – Support for weak references.
• Miscellaneous:
  – Add a new mpi4py publication (link) to the citation listing.
17.7 Release 3.0.3 [2019-11-04]

• Regenerate Cython wrappers to support Python 3.8.

17.8 Release 3.0.2 [2019-06-11]

• Bug fixes:
  – Fix handling of readonly buffers in support for Python 2 legacy buffer interface. The issue triggers only when using a buffer-like object that is readonly and does not export the new Python 3 buffer interface.
  – Fix build issues with Open MPI 4.0.x series related to removal of many MPI-1 symbols deprecated in MPI-2 and removed in MPI-3.
  – Minor documentation fixes.

17.9 Release 3.0.1 [2019-02-15]

• Bug fixes:
  – Fix Comm.scatter() and other collectives corrupting input send list. Add safety measures to prevent related issues in global reduction operations.
  – Fix error-checking code for counts in Op.Reduce_local().

• Enhancements:
  – Map size-specific Python/NumPy typecodes to MPI datatypes.
  – Allow partial specification of target list/tuple arguments in the various Win RMA methods.
  – Workaround for removal of MPI_{LB|UB} in Open MPI 4.0.
  – Support for Microsoft MPI v10.0.

17.10 Release 3.0.0 [2017-11-08]

• New features:
  – mpi4py.futures: Execute computations asynchronously using a pool of MPI processes. This package is based on concurrent.futures from the Python standard library.
  – mpi4py.run: Run Python code and abort execution in case of unhandled exceptions to prevent deadlocks.
  – mpi4py.bench: Run basic MPI benchmarks and tests.

• Enhancements:
  – Lowercase, pickle-based collective communication calls are now thread-safe through the use of fine-grained locking.
  – The MPI module now exposes a memory type which is a lightweight variant of the built-in memoryview type, but exposes both the legacy Python 2 and the modern Python 3 buffer interface under a Python 2 runtime.
  – The MPI.Comm.Alltoallw() method now uses count=1 and displ=0 as defaults, assuming that messages are specified through user-defined datatypes.
  – The Request.Wait[all]() methods now return True to match the interface of Request.Test[all]().
  – The Win class now implements the Python buffer interface.
• Backward-incompatible changes:
  – The buf argument of the MPI.Comm.recv() method is deprecated, passing anything but None emits a warning.
  – The MPI.Win.memory property was removed, use the MPI.Win.tomemory() method instead.
  – Executing python -m mpi4py in the command line is now equivalent to python -m mpi4py.run. For the former behavior, use python -m mpi4py.bench.
  – Python 2.6 and 3.2 are no longer supported. The mpi4py.MPI module may still build and partially work, but other pure-Python modules under the mpi4py namespace will not.
  – Windows: Remove support for legacy MPICH2, Open MPI, and DeinoMPI.

17.11 Release 2.0.0 [2015-10-18]

• Support for MPI-3 features.
  – Matched probes and receives.
  – Nonblocking collectives.
  – Neighborhood collectives.
  – New communicator constructors.
  – Request-based RMA operations.
  – New RMA communication and synchronisation calls.
  – New window constructors.
  – New datatype constructor.
  – New C++ boolean and floating complex datatypes.

• Support for MPI-2 features not included in previous releases.
  – Generalized All-to-All collective (Comm.Alltoallw())
  – User-defined data representations (Register_datarep())

• New scalable implementation of reduction operations for Python objects. This code is based on binomial tree algorithms using point-to-point communication and duplicated communicator contexts. To disable this feature, use mpi4py.rc.fast_reduce = False.

• Backward-incompatible changes:
  – Python 2.4, 2.5, 3.0 and 3.1 are no longer supported.
  – Default MPI error handling policies are overridden. After import, mpi4py sets the ERRORS_RETURN error handler in COMM_SELF and COMM_WORLD, as well as any new Comm, Win, or File instance created through mpi4py, thus effectively ignoring the MPI rules about error handler inheritance. This way, MPI errors translate to Python exceptions. To disable this behavior and use the standard MPI error handling rules, use mpi4py.rc.errors = 'default'.
  – Change signature of all send methods, dest is a required argument.
  – Change signature of all receive and probe methods, source defaults to ANY_SOURCE, tag defaults to ANY_TAG.
  – Change signature of send lowercase-spelling methods, obj arguments are not mandatory.
  – Change signature of recv lowercase-spelling methods, renamed ‘obj’ arguments to ‘buf’.
- Change `Request.Waitsome()` and `Request.Testsome()` to return `None` or list.
- Change signature of all lowercase-spelling collectives, `sendobj` arguments are now mandatory, `recvobj` arguments were removed.
- Reduction operations `MAXLOC` and `MINLOC` are no longer special-cased in lowercase-spelling methods `Comm.[all]reduce()` and `Comm.[ex]scan()`, the input object must be specified as a tuple `(obj, location)`.
- Change signature of name publishing functions. The new signatures are `Publish_name(service_name, port_name, info=INFO_NULL)` and `Unpublish_name(service_name, port_name, info=INFO_NULL)`.
- `Win` instances now cache Python objects exposing memory by keeping references instead of using MPI attribute caching.
- Change signature of `Win.Lock()`. The new signature is `Win.Lock(rank, lock_type=LOCK_EXCLUSIVE, assertion=0)`.
- Move `Graphcomm.Map()` to `Intracomm.Graph_map()`.
- Remove the `mpi4py.MPE` module.
- Rename the Cython definition file for use with `cimport` statement from `mpi_c.pxd` to `libmpi.pxd`.

### 17.12 Release 1.3.1 [2013-08-07]

- Regenerate C wrappers with Cython 0.19.1 to support Python 3.3.
- Install `*.pxd` files in `<site-packages>/mpi4py` to ease the support for Cython's `cimport` statement in code requiring to access mpi4py internals.
- As a side-effect of using Cython 0.19.1, ancient Python 2.3 is no longer supported. If you really need it, you can install an older Cython and run `python setup.py build_src --force`.

### 17.13 Release 1.3 [2012-01-20]

- Now `Comm.recv()` accept a buffer to receive the message.
- Add `Comm.irecv()` and `Request.{wait|test}[any|all]()`.
- Add `Intracomm.Spawn_multiple()`.
- Better buffer handling for PEP 3118 and legacy buffer interfaces.
- Add support for attribute attribute caching on communicators, datatypes and windows.
- Install MPI-enabled Python interpreter as `<path>/mpi4py/bin/python-mpi`.
- Windows: Support for building with Open MPI.
17.14 Release 1.2.2 [2010-09-13]

- Add mpi4py.get_config() to retrieve information (compiler wrappers, includes, libraries, etc) about the MPI implementation employed to build mpi4py.
- Workaround Python libraries with missing GILState-related API calls in case of non-threaded Python builds.
- Windows: look for MPICH2, DeinoMPI, Microsoft HPC Pack at their default install locations under %ProgramFiles%.
- MPE: fix hacks related to old API’s, these hacks are broken when MPE is built with a MPI implementations other than MPICH2.
- HP-MPI: fix for missing Fortran datatypes, use dlopen() to load the MPI shared library before MPI_Init()
- Many distutils-related fixes, cleanup, and enhancements, better logics to find MPI compiler wrappers.
- Support for pip install mpi4py.

17.15 Release 1.2.1 [2010-02-26]

- Fix declaration in Cython include file. This declaration, while valid for Cython, broke the simple-minded parsing used in conf/mpidistutils.py to implement configure-tests for availability of MPI symbols.
- Update SWIG support and make it compatible with Python 3. Also generate an warning for SWIG < 1.3.28.
- Fix distutils-related issues in Mac OS X. Now ARCHFLAGS environment variable is honored of all Python’s config/Makefile variables.
- Fix issues with Open MPI < 1.4.2 related to error checking and MPI___NULL handles.

17.16 Release 1.2 [2009-12-29]

- Automatic MPI datatype discovery for NumPy arrays and PEP-3118 buffers. Now buffer-like objects can be messaged directly, it is no longer required to explicitly pass a 2/3-list/tuple like [data, MPI.DOUBLE], or [data, count, MPI.DOUBLE]. Only basic types are supported, i.e., all C/C99-native signed/unsigned integral types and single/double precision real/complex floating types. Many thanks to Eilif Muller for the initial feedback.
- Nonblocking send of pickled Python objects. Many thanks to Andreas Kloeckner for the initial patch and enlightening discussion about this enhancement.
- Request instances now hold a reference to the Python object exposing the buffer involved in point-to-point communication or parallel I/O. Many thanks to Andreas Kloeckner for the initial feedback.
- Support for logging of user-defined states and events using MPE. Runtime (i.e., without requiring a recompile!) activation of logging of all MPI calls is supported in POSIX platforms implementing dlopen().
- Support for all the new features in MPI-2.2 (new C99 and F90 datatypes, distributed graph topology, local reduction operation, and other minor enhancements).
- Fix the annoying issues related to Open MPI and Python dynamic loading of extension modules in platforms supporting dlopen().
- Fix SLURM dynamic loading issues on SiCortex. Many thanks to Ian Langmore for providing me shell access.
17.17 Release 1.1.0 [2009-06-06]

- Fix bug in Comm. Iprobe() that caused segfaults as Python C-API calls were issued with the GIL released (issue #2).
- Add Comm. bsend() and Comm. ssend() for buffered and synchronous send semantics when communicating general Python objects.
- Now the call Info. Get(key) return a single value (i.e., instead of a 2-tuple); this value is None if key is not in the Info object, or a string otherwise. Previously, the call redundantly returned (None, False) for missing key-value pairs; None is enough to signal a missing entry.
- Add support for parametrized Fortran datatypes.
- Add support for decoding user-defined datatypes.
- Add support for user-defined reduction operations on memory buffers. However, at most 16 user-defined reduction operations can be created. Ask the author for more room if you need it.

17.18 Release 1.0.0 [2009-03-20]

This is the first release of the all-new, Cython-based, implementation of MPI for Python. Unfortunately, this implementation is not backward-compatible with the previous one. The list below summarizes the more important changes that can impact user codes.

- Some communication calls had overloaded functionality. Now there is a clear distinction between communication of general Python object with pickle, and (fast, near C-speed) communication of buffer-like objects (e.g., NumPy arrays).
  - for communicating general Python objects, you have to use all-lowercase methods, like send(), recv(), bcast(), etc.
  - for communicating array data, you have to use Send(), Recv(), Bcast(), etc. methods. Buffer arguments to these calls must be explicitly specified by using a 2/3-list/tuple like [data, MPI.DOUBLE], or [data, count, MPI.DOUBLE] (the former one uses the byte-size of data and the extent of the MPI datatype to define the count).
- Indexing a communicator with an integer returned a special object associating the communication with a target rank, alleviating you from specifying source/destination/root arguments in point-to-point and collective communications. This functionality is no longer available, expressions like:

  ```
  MPI.COMM_WORLD[0].Send(...) 
  MPI.COMM_WORLD[0].Recv(...) 
  MPI.COMM_WORLD[0].Bcast(...) 
  ```

  have to be replaced by:

  ```
  MPI.COMM_WORLD.Send(..., dest=0) 
  MPI.COMM_WORLD.Recv(..., source=0) 
  MPI.COMM_WORLD.Bcast(..., root=0) 
  ```
- Automatic MPI initialization (i.e., at import time) requests the maximum level of MPI thread support (i.e., it is done by calling MPI_Init_thread() and passing MPI_THREAD_MULTIPLE). In case you need to change this behavior, you can tweak the contents of the mp14py.rc module.
- In order to obtain the values of predefined attributes attached to the world communicator, now you have to use the Get_attr() method on the MPI.COMM_WORLD instance:
In the previous implementation, MPI.COMM_WORLD and MPI.COMM_SELF were associated to duplicates of the (C-level) MPI_COMM_WORLD and MPI_COMM_SELF predefined communicator handles. Now this is no longer the case, MPI.COMM_WORLD and MPI.COMM_SELF proxies the actual MPI_COMM_WORLD and MPI_COMM_SELF handles.

- Convenience aliases MPI.WORLD and MPI.SELF were removed. Use instead MPI.COMM_WORLD and MPI.COMM_SELF.
- Convenience constants MPI.WORLD_SIZE and MPI.WORLD_RANK were removed. Use instead MPI.COMM_WORLD.Get_size() and MPI.COMM_WORLD.Get_rank().

References


Python Module Index

m
mpi4py, 20
mpi4py.bench, 58
mpi4py.futures, 38
mpi4py.MPI, 58
mpi4py.run, 56
mpi4py.typing, 34
mpi4py.util, 47
mpi4py.util.dtlib, 47
mpi4py.util.pkl5, 47
mpi4py.util.pool, 54
Index

Symbols

_A__init__() (mpi4py.util.pool.Pool method), 54
_A__new__() (mpi4py.MPI.BottomType static method), 59
_A__new__() (mpi4py.MPI.Cartcomm static method), 59
_A__new__() (mpi4py.MPI.Comm static method), 61
_A__new__() (mpi4py.MPI.Datatype static method), 91
_A__new__() (mpi4py.MPI.Distgraphcomm static method), 101
_A__new__() (mpi4py.MPI.Errhandler static method), 101
_A__new__() (mpi4py.MPI.Exception static method), 166
_A__new__() (mpi4py.MPI.File static method), 102
_A__new__() (mpi4py.MPI.Graphcomm static method), 114
_A__new__() (mpi4py.MPI.Group static method), 116
_A__new__() (mpi4py.MPI.InPlaceType static method), 120
_A__new__() (mpi4py.MPI.Info static method), 120
_A__new__() (mpi4py.MPI.Intercomm static method), 124
_A__new__() (mpi4py.MPI.Intracomm static method), 125
_A__new__() (mpi4py.MPI.Message static method), 131
_A__new__() (mpi4py.MPI.Op static method), 133
_A__new__() (mpi4py.MPI.Pickle static method), 135
_A__new__() (mpi4py.MPI.Prequest static method), 137
_A__new__() (mpi4py.MPI.Request static method), 138
_A__new__() (mpi4py.MPI.Session static method), 143
_A__new__() (mpi4py.MPI.Status static method), 145
_A__new__() (mpi4py.MPI.Topocomm static method), 148
_A__new__() (mpi4py.MPI.Win static method), 153
_A__new__() (mpi4py.MPI.Win class method), 163
--configure
   command line option, 224
   --mpi
      command line option, 224
   --mpicc
      command line option, 224
   --mptild
      command line option, 224
   -c
      command line option, 57
   -m
      command line option, 57
A

Abort() (mpi4py.MPI.Comm method), 65
Accept() (mpi4py.MPI.Intracomm method), 126
Accumulate() (mpi4py.MPI.Win method), 155
Ack_failed() (mpi4py.MPI.Comm method), 65
Add_error_class() (in module mpi4py.MPI), 168
Add_error_code() (in module mpi4py.MPI), 168
Add_error_string() (in module mpi4py.MPI), 168
address (mpi4py.MPI.memory attribute), 165
Agree() (mpi4py.MPI.Comm method), 65
AINT (in module mpi4py.MPI), 203
Aint (in module mpi4py.typing), 36
Aint_add() (in module mpi4py.MPI), 168
Aint_diff() (in module mpi4py.MPI), 169
Allgather() (mpi4py.MPI.Comm method), 65
allgather() (mpi4py.MPI.Comm method), 86
allgather() (mpi4py.util.pkl5.Comm method), 52
Allgather_init() (mpi4py.MPI.Comm method), 65
Allgatherv() (mpi4py.MPI.Comm method), 65
Allgatherv_init() (mpi4py.MPI.Comm method), 66
Alloc_mem() (in module mpi4py.MPI), 169
allocate() (mpi4py.MPI.memory static method), 164
Allocate() (mpi4py.MPI.Win class method), 155
Allocate_shared() (mpi4py.MPI.Win class method), 155
Allreduce() (mpi4py.MPI.Comm method), 66
allreduce() (mpi4py.MPI.Comm method), 86
Allreduce_init() (mpi4py.MPI.Comm method), 66
Alltoall() (mpi4py.MPI.Comm method), 66
alltoall() (mpi4py.MPI.Comm method), 86
alltoall() (mpi4py.util.pkl5.Comm method), 52
Alltoall_init() (mpi4py.MPI.Comm method), 67
Alltoallv() (mpi4py.MPI.Comm method), 67
Alltoallow() (mpi4py.MPI.Comm method), 67
Alltoallv_init() (mpi4py.MPI.Comm method), 67
Alltoallow_init() (mpi4py.MPI.Comm method), 67
amode (mpi4py.MPI.File attribute), 114
ANY_SOURCE (in module mpi4py.MPI), 181
ANY_TAG (in module mpi4py.MPI), 181
apply() (mpi4py.util.pool.Pool method), 54
apply_async() (mpi4py.util.pool.Pool method), 54
AsyncResult (class in mpi4py.util.pool), 56
APPNUM (in module mpi4py.MPI), 182
AsyncResult (class in mpi4py.util.pool), 55
atomicity (mpi4py.MPI.File attribute), 114
Attach() (mpi4py.MPI.Win method), 155
Attach_buffer() (in module mpi4py.MPI), 169
attrs (mpi4py.MPI.Win attribute), 163
B

BAND (in module mpi4py.MPI), 215
Barrier() (mpi4py.MPI.Comm method), 68
barrier() (mpi4py.MPI.Comm method), 86
Barrier_init() (mpi4py.MPI.Comm method), 68
Bcast() (mpi4py.MPI.Comm method), 68
bcast() (mpi4py.MPI.Comm method), 87
bcast() (mpi4py.util.pkl5.Comm method), 52
Bcast_init() (mpi4py.MPI.Comm method), 68
BOOL (in module mpi4py.MPI), 211
bootup() (mpi4py.futures.MPIPoolExecutor method), 40
Bor (in module mpi4py.MPI), 215
BOTTOM (in module mpi4py.MPI), 181
Bottom (in module mpi4py.typing), 35
BottomType (class in mpi4py.MPI), 59
Bsend() (mpi4py.MPI.Comm method), 68
bsend() (mpi4py.MPI.Comm method), 87
bsend() (mpi4py.util.pkl5.Comm method), 49
Bsend_init() (mpi4py.MPI.Comm method), 68
BSEND_OVERHEAD (in module mpi4py.MPI), 197
Buffer (in module mpi4py.MPI), 35
BufSpec (in module mpi4py.MPI), 36
BufSpecB (in module mpi4py.MPI), 36
BufSpecV (in module mpi4py.MPI), 36
BufSpecW (in module mpi4py.MPI), 37
BXOR (in module mpi4py.MPI), 216
BYTE (in module mpi4py.MPI), 202

C
C_BOOL (in module mpi4py.MPI), 205
C_COMPLEX (in module mpi4py.MPI), 206
C_DOUBLE_COMPLEX (in module mpi4py.MPI), 206
C_FLOAT_COMPLEX (in module mpi4py.MPI), 206
C_LONG_DOUBLE_COMPLEX (in module mpi4py.MPI), 206
Call_errhandler() (mpi4py.MPI.Comm method), 69
Call_errhandler() (mpi4py.MPI.File method), 104
Call_errhandler() (mpi4py.MPI.Session method), 144
Call_errhandler() (mpi4py.MPI.Win method), 156
Cancel() (mpi4py.MPI.Request method), 139
cancel() (mpi4py.MPI.Request method), 141
cancelled (mpi4py.MPI.Status attribute), 148
CART (in module mpi4py.MPI), 196
cart_map() (mpi4py.MPI.Intracomm method), 126
cartcomm (class in mpi4py.MPI), 59
CC, 221
CHAR (in module mpi4py.MPI), 203
CHARACTER (in module mpi4py.MPI), 208
clear() (mpi4py.MPI.Info method), 122
Clone() (mpi4py.MPI.Comm method), 69
Close() (mpi4py.MPI.File method), 104
close() (mpi4py.util.pool.Pool method), 55
close_port() (in module mpi4py.MPI), 169
combiner (mpi4py.MPI.Datatype attribute), 100
COMBINER_CONTIGUOUS (in module mpi4py.MPI), 193
COMBINER_DARRAY (in module mpi4py.MPI), 194
COMBINER_DUP (in module mpi4py.MPI), 193
COMBINER_F90_COMPLEX (in module mpi4py.MPI), 195
COMBINER_F90_INTEGER (in module mpi4py.MPI), 194
COMBINER_F90_REAL (in module mpi4py.MPI), 194
COMBINER_HINDEXED (in module mpi4py.MPI), 193
COMBINER_HINDEXED_BLOCK (in module mpi4py.MPI), 194
COMBINER_HVECTOR (in module mpi4py.MPI), 193
COMBINER_INDEXED (in module mpi4py.MPI), 193
COMBINER_INDEXED_BLOCK (in module mpi4py.MPI), 194
COMBINER_NAMED (in module mpi4py.MPI), 193
COMBINER_RESIZED (in module mpi4py.MPI), 194
COMBINER_STRUCT (in module mpi4py.MPI), 194
COMBINER_SUBARRAY (in module mpi4py.MPI), 194
COMBINER_VECTOR (in module mpi4py.MPI), 193
Comm (class in mpi4py.MPI), 61
Comm (class in mpi4py.util.pkl5), 49
COMM_NULL (in module mpi4py.MPI), 218
COMM_SELF (in module mpi4py.MPI), 218
COMM_TYPE_HW_GUIDED (in module mpi4py.MPI), 196
COMM_TYPE_HW_UNGUIDED (in module mpi4py.MPI), 197
COMM_TYPE_SHARED (in module mpi4py.MPI), 196
COMM_WORLD (in module mpi4py.MPI), 218
command line option
--configure, 224
--mpi, 224
--mpicc, 224
--mpild, 224
-c, 57
-m, 57
Commit() (mpi4py.MPI.Datatype method), 93
Compare() (mpi4py.MPI.Comm method), 69
Compare() (mpi4py.MPI.Group method), 118
Compare_and_swap() (mpi4py.MPI.Win method), 156
Complete() (mpi4py.MPI.Grequest method), 116
Complete() (mpi4py.MPI.Win method), 156
COMPLEX (in module mpi4py.MPI), 208
COMPLEX16 (in module mpi4py.MPI), 211
COMPLEX32 (in module mpi4py.MPI), 211
COMPLEX4 (in module mpi4py.MPI), 210
COMPLEX8 (in module mpi4py.MPI), 210
Compute_dims() (in module mpi4py.MPI), 170
CONGRUENT (in module mpi4py.MPI), 195
Connect() (mpi4py.MPI.Intracomm method), 126
contents (mpi4py.MPI.Datatype attribute), 100
coords (mpi4py.MPI.Cartcomm attribute), 61
copy() (mpi4py.MPI.Info method), 122
COUNT (in module mpi4py.MPI), 203
Count (in module mpi4py.MPI), 36
count (mpi4py.MPI.Status attribute), 148
Create() (mpi4py.MPI.Comm method), 69
Create() (mpi4py.MPI.Info class method), 121
Create() (mpi4py.MPI.Op class method), 134
Create() (mpi4py.MPI.Win class method), 156
Create_cart() (mpi4py.MPI.Intracomm method), 127
Create_contiguous() (mpi4py.MPI.Datatype method), 93
Create_darray() (mpi4py.MPI.Datatype method), 93
G

Gather() (mpi4py.MPI.Comm method), 70

gather() (mpi4py.MPI.Comm method), 87

gather() (mpi4py.util.pkl5.Comm method), 52

Gather_init() (mpi4py.MPI.Comm method), 70

GatherProc() (mpi4py.MPI.Comm method), 71

Gather_init() (mpi4py.MPI.Comm method), 71

Get() (mpi4py.MPI.Info method), 122

get() (mpi4py.MPI.Win method), 158

Get accumulate() (mpi4py.MPI.Win method), 158

Get_address() (mpi4py.MPI.Win method), 170

Get_amode() (mpi4py.MPI.File method), 105

Get_atomicity() (mpi4py.MPI.File method), 105

Get_attr() (mpi4py.MPI.Comm method), 71

Get_attr() (mpi4py.MPI.Datatype method), 97

Get_attr() (mpi4py.MPI.Win method), 159

Get_byte_offset() (mpi4py.MPI.File method), 105

Get_cart_rank() (mpi4py.MPI.Cartcomm method), 60

get_comm_workers() (mpi4py.MPI.File method), 43

get_config() (mpi4py.MPI.File method), 25

Get_contents() (mpi4py.MPI.Datatype method), 97

Get_coords() (mpi4py.MPI.Cartcomm method), 60

Get_count() (mpi4py.MPI.Status method), 146

Get_dim() (mpi4py.MPI.Cartcomm method), 60

Get_dims() (mpi4py.MPI.Graphcomm method), 115

Get_dist_neighbors() (mpi4py.MPI.Distgraphcomm method), 101

Get_dist_neighbors_count() (mpi4py.MPI.Distgraphcomm method), 101

Get_elements() (mpi4py.MPI.Status method), 146

Get_envelope() (mpi4py.MPI.Datatype method), 97

Get_errhandler() (mpi4py.MPI.Comm method), 71

Get_errhandler() (mpi4py.MPI.File method), 105

Get_errhandler() (mpi4py.MPI.Session method), 144

Get_error() (mpi4py.MPI.Status method), 146

Get_error_class() (in module mpi4py.MPI), 171

Get_error_class() (mpi4py.MPI.Exception method), 166

Get_error_code() (mpi4py.MPI.Exception method), 166

Get_error_string() (in module mpi4py.MPI), 171

Get_error_string() (mpi4py.MPI.Exception method), 166

Get_extent() (mpi4py.MPI.Datatype method), 97

Get_failed() (mpi4py.MPI.Comm method), 71

Get_group() (mpi4py.MPI.Comm method), 72

Get_group() (mpi4py.MPI.File method), 105

Get_group() (mpi4py.MPI.Win method), 159

get_include() (mpi4py.MPI.File method), 25

Get_info() (mpi4py.MPI.Comm method), 72

Get_info() (mpi4py.MPI.File method), 105
InPlaceType (class in mpi4py.MPI), 120
INT (in module mpi4py.MPI), 203
INT16_T (in module mpi4py.MPI), 205
INT32_T (in module mpi4py.MPI), 205
INT64_T (in module mpi4py.MPI), 205
INTER (in module mpi4py.MPI), 208
INTER1 (in module mpi4py.MPI), 209
INTER16 (in module mpi4py.MPI), 210
INTER2 (in module mpi4py.MPI), 209
INTER4 (in module mpi4py.MPI), 209
INTER8 (in module mpi4py.MPI), 210
Intersection (class in mpi4py.MPI), 124
Intersection (class in mpi4py.util.pkl5), 53
Intersection (class in mpi4py.MPI.Group class method), 119
Intracomm (class in mpi4py.MPI), 125
Intracomm (class in mpi4py.util.pkl5), 53
I0 (in module mpi4py.MPI), 182
iprobe() (mpi4py.MPI.Comm method), 75
iprobe() (mpi4py.MPI.Commp method), 88
iprobe() (mpi4py.MPI.Message class method), 132
iprobe() (mpi4py.MPI.Message class method), 132
iprobe() (mpi4py.util.pkl5.Message class method), 49
Iread() (mpi4py.MPI.File method), 106
Iread_all() (mpi4py.MPI.File method), 106
Iread_at() (mpi4py.MPI.File method), 106
Iread_at_all() (mpi4py.MPI.File method), 106
Iread_shared() (mpi4py.MPI.File method), 106
Irecv() (mpi4py.MPI.Commp method), 75
Irecv() (mpi4py.MPI.Commp method), 88
Irecv() (mpi4py.MPI.Message class method), 132
Irecv() (mpi4py.MPI.Message class method), 133
Irecv() (mpi4py.util.pkl5.Commp method), 51
Irecv() (mpi4py.util.pkl5.Message method), 49
Ireduce() (mpi4py.MPI.Commp method), 75
Ireduce_scatter() (mpi4py.MPI.Commp method), 76
Ireduce_scatter_block() (mpi4py.MPI.Commp method), 76
Irsend() (mpi4py.MPI.Commp method), 76
Is_cancelled() (mpi4py.MPI.Status method), 147
is_commutative (mpi4py.MPI.Op attribute), 135
is_commutative (mpi4py.MPI.Op method), 134
Is_finalized() (in module mpi4py.MPI), 172
Is_initialized() (in module mpi4py.MPI), 172
Is_inter (mpi4py.MPI.Commp attribute), 91
Is_inter() (mpi4py.MPI.Commp method), 76
is_intra (mpi4py.MPI.Commp attribute), 91
Is_intra() (mpi4py.MPI.Commp method), 76
is_named (mpi4py.MPI.Datatype attribute), 100
is_predefined (mpi4py.MPI.Datatype attribute), 100
is_predefined (mpi4py.MPI.Op attribute), 135
Is_revoked() (mpi4py.MPI.Commp method), 76
Is_thread_main() (in module mpi4py.MPI), 172
is_topo (mpi4py.MPI.Commp attribute), 91
iscan() (mpi4py.MPI.Intracomm method), 129
Isscatter() (mpi4py.MPI.Commp method), 77
Isscatterv() (mpi4py.MPI.Commp method), 77
Isend() (mpi4py.MPI.Commp method), 77
Isend() (mpi4py.MPI.Commp method), 88
Isend() (mpi4py.util.pkl5.Commp method), 50
Isendrecv() (mpi4py.MPI.Commp method), 77
Isendrecv_replace() (mpi4py.MPI.Commp method), 77
Ishrink() (mpi4py.MPI.Commp method), 78
issend() (mpi4py.MPI.Commp method), 78
issend() (mpi4py.MPI.Commp method), 88
issend() (mpi4py.util.pkl5.Commp method), 50
istarmap() (mpi4py.util.pool.Pool method), 55
istarmap_unordered() (mpi4py.util.pool.Pool method), 55
items() (mpi4py.MPI.Info method), 123
itemsize (mpi4py.MPI.memory attribute), 165
Iwrite() (mpi4py.MPI.File method), 107
Iwrite_all() (mpi4py.MPI.File method), 107
Iwrite_at() (mpi4py.MPI.File method), 107
Iwrite_at_all() (mpi4py.MPI.File method), 107
Iwrite_shared() (mpi4py.MPI.File method), 107
J
Join() (mpi4py.MPI.Commp class method), 78
join() (mpi4py.util.pool.Pool method), 55
K
keys() (mpi4py.MPI.Info method), 123
KEYVAL_INVALID (in module mpi4py.MPI), 181
L
LAND (in module mpi4py.MPI), 215
LASTUSED_CODE (in module mpi4py.MPI), 182
lb (mpi4py.MPI.Datatype attribute), 100
LD_LIBRARY_PATH, 227
loads() (mpi4py.MPI.Pickle method), 136
loads_oob() (mpi4py.MPI.Pickle method), 136
Lock() (mpi4py.MPI.Win method), 159
Lock_all() (mpi4py.MPI.Win method), 136
LOCK_EXCLUSIVE (in module mpi4py.MPI), 198
LOCK_SHARED (in module mpi4py.MPI), 198
LOGICAL (in module mpi4py.MPI), 208
LOGICAL1 (in module mpi4py.MPI), 209
LOGICAL2 (in module mpi4py.MPI), 209
LOGICAL4 (in module mpi4py.MPI), 209
LOGICAL8 (in module mpi4py.MPI), 209
LONG (in module mpi4py.MPI), 204
LONG_DOUBLE (in module mpi4py.MPI), 205
LONG_DOUBLE_INT (in module mpi4py.MPI), 208
LONG_INT (in module mpi4py.MPI), 207
LONG_LONG (in module mpi4py.MPI), 204
Lookup_name() (in module mpi4py.MPI), 172
LOR (in module mpi4py.MPI), 215
LXOR (in module mpi4py.MPI), 216

M

map() (mpi4py.futures.MPIPoolExecutor method), 39
map() (mpi4py.util.pool.Pool method), 55
map_async() (mpi4py.util.pool.Pool method), 55
MapResult (class in mpi4py.util.pool), 56
Match_size() (mpi4py.MPI.Datatypeclassmethod), 97
MAX (in module mpi4py.MPI), 214
MAX_DATAREP_STRING (in module mpi4py.MPI), 202
MAX_ERROR_STRING (in module mpi4py.MPI), 201
MAX_INFO_KEY (in module mpi4py.MPI), 201
MAX_INFO_VAL (in module mpi4py.MPI), 201
MAX_LIBRARY_VERSION_STRING (in module mpi4py.MPI), 202
MAX_OBJECT_NAME (in module mpi4py.MPI), 202
MAX_PORT_NAME (in module mpi4py.MPI), 201
MAX_PROCESSOR_NAME (in module mpi4py.MPI), 201
MAX_PSET_NAME_LEN (in module mpi4py.MPI), 202
MAX_STRINGTAG_LEN (in module mpi4py.MPI), 202
MAXLOC (in module mpi4py.MPI), 216
memory (class in mpi4py.MPI), 163
Merge() (mpi4py.MPI.Intercomm method), 125
Message (class in mpi4py.MPI), 131
Message (class in mpi4py.util.pkl5), 48
MESSAGE_NO_PROC (in module mpi4py.MPI), 213
MESSAGE_NULL (in module mpi4py.MPI), 213
MIN (in module mpi4py.MPI), 214
MINLOC (in module mpi4py.MPI), 216
MODE_APPEND (in module mpi4py.MPI), 199
MODE_CREATE (in module mpi4py.MPI), 199
MODE_DELETE_ON_CLOSE (in module mpi4py.MPI), 199
MODE_EXCL (in module mpi4py.MPI), 199
MODE_NOCHECK (in module mpi4py.MPI), 198
MODE_NOPRECEDE (in module mpi4py.MPI), 198
MODE_NOPUT (in module mpi4py.MPI), 198
MODE_NOSTORE (in module mpi4py.MPI), 198
MODE_RONLY (in module mpi4py.MPI), 198
MODE_RDWR (in module mpi4py.MPI), 199
MODE_SEQUENTIAL (in module mpi4py.MPI), 199
MODE_UNIQUE_OPEN (in module mpi4py.MPI), 199
MODE_WONLY (in module mpi4py.MPI), 199
model (mpi4py.MPI.Win attribute), 163
module
  mpi4py, 20
  mpi4py.bench, 58
  mpi4py.futures, 38
  mpi4py.MPI, 58
  mpi4py.run, 56
Unpack_external() (mpi4py.MPI.Datatype method), 99
Unpublish_name() (in module mpi4py.MPI), 174
UNSIGNED (in module mpi4py.MPI), 204
UNSIGNED_CHAR (in module mpi4py.MPI), 204
UNSIGNED_INT (in module mpi4py.MPI), 211
UNSIGNED_LONG (in module mpi4py.MPI), 204
UNSIGNED_LONG_LONG (in module mpi4py.MPI), 204
UNSIGNED_SHORT (in module mpi4py.MPI), 204
UNWEIGHTED (in module mpi4py.MPI), 196
Update() (mpi4py.MPI.Info method), 123
values() (mpi4py.MPI.Info method), 124
VERSION (in module mpi4py.MPI), 201
Wait() (mpi4py.MPI.Request method), 140
wait() (mpi4py.MPI.Request method), 142
Wait() (mpi4py.MPI.Win method), 162
wait() (mpi4py.util.poolAsyncResult method), 56
waitall() (mpi4py.MPI.Request class method), 140
waitall() (mpi4py.MPI.Request class method), 142
waitall() (mpi4py.util.poolAsyncResult method), 48
waitany() (mpi4py.MPI.Request class method), 140
waitany() (mpi4py.MPI.Request class method), 142
waitany() (mpi4py.MPI.Request class method), 141
waitsome() (mpi4py.MPI.Request class method), 143
WCHAR (in module mpi4py.MPI), 203
WEIGHTS_EMPTY (in module mpi4py.MPI), 196
Win (class in mpi4py.MPI), 153
WIN_BASE (in module mpi4py.MPI), 182
WIN_CREATE_FLAVOR (in module mpi4py.MPI), 183
WIN_DISP_UNIT (in module mpi4py.MPI), 183
WIN_FLAVOR (in module mpi4py.MPI), 183
WIN_FLAVOR_ALLOCATE (in module mpi4py.MPI), 197
WIN_FLAVOR_CREATE (in module mpi4py.MPI), 197
WIN_FLAVOR_DYNAMIC (in module mpi4py.MPI), 197
WIN_FLAVOR_SHARED (in module mpi4py.MPI), 197
WIN_MODEL (in module mpi4py.MPI), 183
WIN_NULL (in module mpi4py.MPI), 219
WIN_SEPARATE (in module mpi4py.MPI), 197
WIN_SIZE (in module mpi4py.MPI), 183
WIN_UNIFIED (in module mpi4py.MPI), 197
Write() (mpi4py.MPI.File method), 111
Write_all() (mpi4py.MPI.File method), 111
Write_all_begin() (mpi4py.MPI.File method), 112
Write_all_end() (mpi4py.MPI.File method), 112
Write_at() (mpi4py.MPI.File method), 112
Write_at_all() (mpi4py.MPI.File method), 112
Write_at_all_begin() (mpi4py.MPI.File method), 112
Write_at_all_end() (mpi4py.MPI.File method), 113
Wtick() (in module mpi4py.MPI), 174
Wtime() (in module mpi4py.MPI), 174
WTIME_IS_GLOBAL (in module mpi4py.MPI), 182