

Chapter 1

Errata for MPI-2

This document was processed on July 1, 2008.

The known corrections to MPI-2 are listed in this document. All page and line numbers are for the official version of the MPI-2 document available from the MPI Forum home page at <http://www.mpi-forum.org>. Information on reporting mistakes in the MPI documents is also located on the MPI Forum home page.

- Page 24, lines 20-21 read
MPI_FINALIZE is collective on MPI_COMM_WORLD.
but should read
MPI_FINALIZE is collective over all connected processes. If no processes were spawned, accepted or connected then this means over MPI_COMM_WORLD; otherwise it is collective over the union of all processes that have been and continue to be connected, as explained in Section 5.5.4.
- Page 27, line 26 reads
must be added to line 3 of page 54.
but should read
must be added to line 3 of page 52.
- Add to page 36, after line 3
3.2.11 MPI_GET_COUNT with zero-length datatypes
The value returned as the `count` argument of `MPI_GET_COUNT` for a datatype of length zero where zero bytes have been transferred is zero. If the number of bytes transferred is greater than zero, `MPI_UNDEFINED` is returned.

Rationale. Zero-length datatypes may be created in a number of cases. In MPI-2, an important case is `MPI_TYPE_CREATE_DARRAY`, where the definition of the particular darray results in an empty block on some MPI process. Programs written in an SPMD style will not check for this special case and may want to use `MPI_GET_COUNT` to check the status. (*End of rationale.*)
- Add to page 36, after 3.2.11 (above)
3.2.12 MPI_GROUP_TRANSLATE_RANKS and MPI_PROC_NULL

1 MPI_PROC_NULL is a valid rank for input to MPI_GROUP_TRANSLATE_RANKS, which returns
 2 MPI_PROC_NULL as the translated rank.

- 3 ● Page 51, after line 43, add

4 MPI_Errhandler MPI_Errhandler_f2c(MPI_Fint errhandler)

5 MPI_Fint MPI_Errhandler_c2f(MPI_Errhandler errhandler)

6 These were overlooked.

- 7 ● Page 53, line 7 reads

8 void cpp_lib_call(MPI::Comm& cpp_comm);

9 but should read

10 void cpp_lib_call(MPI::Comm cpp_comm);

- 11 ● Page 60, Line 1 reads

12 char name[MPI_MAX_NAME_STRING];

13 but should read

14 char name[MPI_MAX_OBJECT_NAME];

15 since MPI_MAX_NAME_STRING is not an MPI-defined constant.

- 16 ● Page 61, after line 36. Add the following (paralleling the errata to MPI-1.1):

17 MPI_{COMM,WIN,FILE}_GET_ERRHANDLER behave as if a new error handler object is
 18 created. That is, once the error handler is no longer needed, MPI_ERRHANDLER_FREE
 19 should be called with the error handler returned from MPI_ERRHANDLER_GET or MPI_{COMM,WIN,FILE}_GET.
 20 to mark the error handler for deallocation. This provides behavior similar to that of
 21 MPI_COMM_GROUP and MPI_GROUP_FREE.

- 22 ● Page 69, lines 14-15 read

23 MPI::Datatype MPI::Datatype::Resized(const MPI::Aint lb,
 24 const MPI::Aint extent) const

25 but should read

26 MPI::Datatype MPI::Datatype::Create_resized(const MPI::Aint lb,
 27 const MPI::Aint extent) const

- 28 ● On Page 78, after line 27, add:

29 MPI_BYTE should be used to send and receive data that is packed using MPI_PACK_EXTERNAL.

Rationale. MPI_PACK_EXTERNAL specifies that there is no header on the message and further specifies the exact format of the data. Since MPI_PACK may (and is allowed to) use a header, the datatype MPI_PACKED cannot be used for data packed with MPI_PACK_EXTERNAL. (*End of rationale.*)

- On page 93 after line 48, add

Many of the descriptions of the collective routines provide illustrations in terms of blocking MPI point-to-point routines. These are intended solely to indicate what data is sent or received by what process. Many of these examples are *not* correct MPI programs; for purposes of simplicity, they often assume infinite buffering.

- Page 94, line 29 reads
are the original sets of of processes.
but should read
are the original sets of processes.

- Page 114, after line 4, add

MPI_PROC_NULL is a valid target rank in the MPI RMA calls MPI_ACCUMULATE, MPI_GET, and MPI_PUT. The effect is the same as for MPI_PROC_NULL in MPI point-to-point communication.

- Page 116, line 31, reads

```
void MPI::Win::Get(const void *origin_addr, int origin_count, const
MPI::Datatype& origin_datatype, int target_rank, MPI::Aint target_disp,
int target_count, const MPI::Datatype& target_datatype) const
```

but should read

```
void MPI::Win::Get(void *origin_addr, int origin_count, const
MPI::Datatype& origin_datatype, int target_rank, MPI::Aint target_disp,
int target_count, const MPI::Datatype& target_datatype) const
```

- Page 120, after line 13:

MPI_REPLACE, like the other predefined operations, is defined only for the predefined MPI datatypes.

Rationale. The rationale for this is that, for consistency, MPI_REPLACE should have the same limitations as the other operations. Extending it to all datatypes doesn't provide any real benefit. (*End of rationale.*)

- Page 162, lines 43–44 curenly read

The “in place” option for intracommunicators is specified by passing the value MPI_IN_PLACE to the argument `sendbuf` at the root.

but should read

The “in place” option for intracommunicators is specified by passing the value MPI_IN_PLACE to the argument `sendbuf` at all processes.

- Page 162, line 48 reads

Both groups should provide the same `count` value.

1 but should read

2 Both groups should provide count and datatype arguments that specify the same type
3 signature.

- 4
5 • Page 165, lines 4–22 read

6
7
8

9	IN	sendcounts	integer array equal to the group size specifying the number of elements to send to each processor (integer)
10			
11			
12	IN	sdispls	integer array (of length group size). Entry j specifies the displacement in bytes (relative to sendbuf) from which to take the outgoing data destined for process j
13			
14			
15			
16	IN	sendtypes	array of datatypes (of length group size). Entry j specifies the type of data to send to process j (handle)
17			
18			
19	OUT	recvbuf	address of receive buffer (choice)
20	IN	recvcounts	integer array equal to the group size specifying the number of elements that can be received from each processor (integer)
21			
22			
23			
24	IN	rdispls	integer array (of length group size). Entry i specifies the displacement in bytes (relative to recvbuf) at which to place the incoming data from process i
25			
26			
27			
28	IN	recvtypes	array of datatypes (of length group size). Entry i specifies the type of data received from process i (handle)
29			
30			
31			

32 but should read

33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

IN	sendcounts	integer array equal to the group size specifying the number of elements to send to each processor (array of integers)	1 2 3
IN	sdispls	integer array (of length group size). Entry j specifies the displacement in bytes (relative to sendbuf) from which to take the outgoing data destined for process j (array of integers)	4 5 6 7
IN	sendtypes	array of datatypes (of length group size). Entry j specifies the type of data to send to process j (array of handles)	8 9 10
OUT	recvbuf	address of receive buffer (choice)	11
IN	recvcounts	integer array equal to the group size specifying the number of elements that can be received from each processor (array of integers)	12 13 14 15
IN	rdispls	integer array (of length group size). Entry i specifies the displacement in bytes (relative to recvbuf) at which to place the incoming data from process i (array of integers)	16 17 18 19
IN	recvtypes	array of datatypes (of length group size). Entry i specifies the type of data received from process i (array of handles)	20 21 22 23
<ul style="list-style-type: none"> • Page 199, after line 11, add: 			24
<p><i>Advice to implementors.</i> High quality implementations should raise an error when a keyval that was created by a call to <code>MPI_XXX_CREATE_KEYVAL</code> is used with an object of the wrong type with a call to <code>MPI_YYY_GET_ATTR</code>, <code>MPI_YYY_SET_ATTR</code>, <code>MPI_YYY_DELETE_ATTR</code>, or <code>MPI_YYY_FREE_KEYVAL</code>. To do so, it is necessary to maintain, with each keyval, information on the type of the associated user function. (<i>End of advice to implementors.</i>)</p>			25 26 27 28 29 30 31 32
<ul style="list-style-type: none"> • Page 204, line 30 reads 			33
<pre>bool MPI::Win::Get_attr(const MPI::Win& win, int win_keyval, void* attribute_val) const</pre>			34 35 36
<p>but should read</p>			37
<pre>bool MPI::Win::Get_attr(int win_keyval, void* attribute_val) const</pre>			38 39
<ul style="list-style-type: none"> • Page 221, after line 40, add 			40
<p><code>MPI_DISPLACEMENT_CURRENT</code> is invalid unless the amode for the file has <code>MPI_MODE_SEQUENTIAL</code> set.</p>			41 42 43 44
<ul style="list-style-type: none"> • Page 230, line 17, reads 			45
<p>If <code>MPI_MODE_SEQUENTIAL</code> mode was specified when the file was opened, it is erroneous to call the routines in this section.</p>			46 47 48

1 but should read

2 If `MPI_MODE_SEQUENTIAL` mode was specified when the file was opened, it is erroneous
3 to call the routines in this section, with the exception of `MPI_FILE_GET_BYTE_OFFSET`.
4

- 5 • Page 250, line 8 reads
6 with 15 exponent bits, bias = +10383, 112 fraction bits,
7 but should read
8 with 15 exponent bits, bias = +16383, 112 fraction bits,

- 9 • Page 251, Line 18 reads
10

11 `MPI_LONG_LONG` 8
12

13 but should read
14

15 `MPI_LONG_LONG_INT` 8
16

17 In addition, the type `MPI_LONG_LONG` should be added as an optional type; it is a
18 synonym for `MPI_LONG_LONG_INT`.
19

- 20 • Page 253, line 4 reads
21

22 `typedef MPI::Datarep_extent_function(const MPI::Datatype& datatype,`
23

24 but should read
25

26 `typedef void MPI::Datarep_extent_function(const MPI::Datatype& datatype,`
27

- 28 • Page 253, lines 22-24 read
29

30 `typedef MPI::Datarep_conversion_function(void* userbuf,`
31 `MPI::Datatype& datatype, int count, void* filebuf,`
32 `MPI::Offset position, void* extra_state);`
33

34 but should read
35

36 `typedef void MPI::Datarep_conversion_function(void* userbuf,`
37 `MPI::Datatype& datatype, int count, void* filebuf,`
38 `MPI::Offset position, void* extra_state);`

- 39 • Page 273, line 24 reads
40

41 `void Send(void* buf, int count, const MPI::Datatype& type,`
42
43

44 but should read
45

46 `void Send(const void* buf, int count, const MPI::Datatype& type,`
47
48

- Page 332, lines 23-24 read 1
`MPI::Datatype MPI::Datatype::Resized(const MPI::Aint lb,` 2
`const MPI::Aint extent) const` 3
4
 but should read 5
`MPI::Datatype MPI::Datatype::Create_resized(const MPI::Aint lb,` 6
`const MPI::Aint extent) const` 7
8
- Page 334, line 22 read 9
10
`void MPI::Win::Get(const void *origin_addr, int origin_count, const` 11
12
 but should read 13
14
`void MPI::Win::Get(void *origin_addr, int origin_count, const` 15
16
- Page 341, line 18 reads 17
18
`typedef MPI::Datarep_conversion_function(void* userbuf,` 19
20
 but should read 21
22
`typedef void MPI::Datarep_conversion_function(void* userbuf,` 23
24
- Page 341, line 22 reads 25
26
`typedef MPI::Datarep_extent_function(const MPI::Datatype& Datatype,` 27
28
 but should read 29
30
`typedef void MPI::Datarep_extent_function(const MPI::Datatype& Datatype,` 31
32
- Page 343, line 44 33
 Remove the `const` from `const MPI::Datatype`. 34
35
- Page 344, lines 13, 23, 32, 38, and 47 36
 Remove the `const` from `const MPI::Datatype`. 37
38
- Page 345, lines 5 and 11 39
 Remove the `const` from `const MPI::Datatype`. 40
41
- Page 346, line 16 reads 42
43
`// Type: MPI::Errhandler` 44
45
 but should read 46
47
`// Type: const MPI::Errhandler` 48

- Page 354, line 17 reads

```
void Get_version(int& version, int& subversion);
```

but should read

```
void Get_version(int& version, int& subversion)
```

- Page 354, lines 25-30 read

```
Exception::Exception(int error_code);
```

```
int Exception::Get_error_code() const;
```

```
int Exception::Get_error_class() const;
```

```
const char* Exception::Get_error_string() const;
```

but should read

```
Exception::Exception(int error_code)
```

```
int Exception::Get_error_code() const
```

```
int Exception::Get_error_class() const
```

```
const char* Exception::Get_error_string() const
```

- Page 357, line 24 reads

```
MPI_CART_RANK Cartcomm Get_rank int rank
```

but should read

```
MPI_CART_RANK Cartcomm Get_cart_rank int rank
```

- Page 359, line 27 reads

```
MPI_TOPO_TEST Comm Get_topo int status
```

but should read

```
MPI_TOPO_TEST Comm Get_topology int status
```

- **(Pending Errata (see Ballot 3))**

MPI-2, page 179, lines 4-5 change

Thus, the names of MPI_COMM_WORLD, MPI_COMM_SELF, and MPI_COMM_PARENT will have the default of MPI_COMM_WORLD, MPI_COMM_SELF, and MPI_COMM_PARENT.

to

Thus, the names of MPI_COMM_WORLD, MPI_COMM_SELF, and the communicator returned by MPI_COMM_GET_PARENT (if not MPI_COMM_NULL) will have the default of MPI_COMM_WORLD, MPI_COMM_SELF, and MPI_COMM_PARENT.

MPI-2, page 94, line 3-5, change

* The manager is represented as the process with rank 0 in (the remote
 * group of) MPI_COMM_PARENT. If the workers need to communicate among
 * themselves, they can use MPI_COMM_WORLD.

to

* The manager is represented as the process with rank 0 in (the remote
 * group of) the parent communicator. If the workers need to communicate
 * among themselves, they can use MPI_COMM_WORLD.

- **(Pending Errata (see Ballot 3))**

MPI2, page 79, Line 11 is

```
MPI_UNPACK_EXTERNAL (datarep, inbuf, incount, datatype, outbuf, outsize,
position)
```

but should be

```
MPI_UNPACK_EXTERNAL (datarep, inbuf, insize, position, outbuf, outcount,
datatype)
```

- **(Pending Errata (see Ballot 3))**

MPI-2, page 337, line 31-32 reads

```
bool MPI::Win::Get_attr(const MPI::Win&win, int win_keyval,
void* attribute_val) const
```

but should read

```
bool MPI::Win::Get_attr(int win_keyval, void* attribute_val) const
```

- **(Pending Errata (see Ballot 3))**

On page 172, line 37 in section 8.2, change MPI_REQUEST_CANCEL to MPI_CANCEL.

- **(Pending Errata (see Ballot 3))**

MPI-2, page 163, line 22 reads

Within each group, all processes provide the same `recvcounts` argument,
 and the sum of the `recvcounts` entries should be the same for the two
 groups.

but should read

Within each group, all processes provide the same `recvcounts` argument,
 and the sum of the `recvcounts` entries and `datatype` should specify the
 same type signature for the two groups.

1 • **(Pending Errata (see Ballot 3))**

2 MPI-2, page 345, line 37: Remove the const from const MPI::Op.

3 MPI-2, page 346, line 20: Remove the const from const MPI::Group.

4 MPI-2, page 346, add after line 34:

5
6 *Advice to implementors.* If an implementation does not change the value of
7 predefined handles while execution of MPI_Init, the implementation is free to
8 define the predefined operation handles as const MPI::Op and the predefined
9 group handle MPI::GROUP_EMPTY as const MPI::Group. Other predefined
10 handles must not be "const" because they are allowed as INOUT argument in
11 the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR rou-
12 tines. (*End of advice to implementors.*)
13

14 • **(Pending Errata (see Ballot 3))**

15 MPI-1, page 128, line 11, in MPI-1.1 has an extraneous root argument. That line
16 should be

17
18 `MPI_Scan(a, answer, 1, sspair, myOp, comm);`
19

20 • **(Pending Errata (see Ballot 3))** MPI-2, page 223, line 19. Change

21 `MPI_FILE_GET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, IERROR)`
22 `INTEGER FH, ETYPE, FILETYPE, IERROR`
23 `CHARACTER*(*) DATAREP, INTEGER(KIND=MPI_OFFSET_KIND) DISP`
24

25 to

26
27 `MPI_FILE_GET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, IERROR)`
28 `INTEGER FH, ETYPE, FILETYPE, IERROR`
29 `CHARACTER*(*) DATAREP`
30 `INTEGER(KIND=MPI_OFFSET_KIND) DISP`
31

32 in io-2.tex. (Replace the comma after the declaration of datarep)

33 • **(Pending Errata (see Ballot 3))**

34 MPI-2, page 66, line 26, change

35
36 `MPI_TYPE_CREATE_HVECTOR(COUNT, BLOCKLENGTH, STIDE, OLDTYPE, NEWTYPE,`
37 `IERROR)`
38 `INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR`
39 `INTEGER(KIND=MPI_ADDRESS_KIND) STRIDE`
40

41 to

42 `MPI_TYPE_CREATE_HVECTOR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE,`
43 `IERROR)`
44 `INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR`
45 `INTEGER(KIND=MPI_ADDRESS_KIND) STRIDE`
46

47 in misc-2.tex (Replace STIDE with STRIDE).
48

- **(Pending Errata (see Ballot 3))**

Examples in Chapter 3 of MPI 1.1 require several fixes.

MPI 1.1, Example 3.12, page 43, line 47 and page 44, lines 1, 5, 8, 10, and 13, the communicator argument `comm` must be added before the `req` argument.

The `ierr` argument must be added at the end of the argument list in the calls to `MPI_COMM_RANK` and `MPI_WAIT` in MPI 1.1, page 43, line 43, and page 44, lines 6 and 14.

The `ierr` argument must be added at the end of the argument list in the calls to `MPI_WAIT` in MPI 1.1, page 44, lines 35 and 36.

The lines in MPI 1.1, page 52, line 45, and page 53, line 17

```
IF (status(MPI_SOURCE) = 0) THEN
```

should be

```
IF (status(MPI_SOURCE) .EQ. 0) THEN
```

- **(Pending Errata (see Ballot 3))**

MPI 1.1, page 80, line 2, The variable `base` should be declared as `MPI_Aint`, not `int`, in Example 3.34.

- **(Pending Errata (see Ballot 3))**Change MPI-2, page 343, lines 22-23

```
// Type: const void *
MPI::BOTTOM
```

to

```
// Type: void * const
MPI::BOTTOM
```

- **(Pending Errata (see Ballot 3))**In MPI 1.1, page 16, line 23, use `strlen(message) + 1` instead of `strlen(message)` in the `MPI_Send` call.

- **(Pending Errata (see Ballot 3))**A LaTeX line break is needed in MPI 1.1, page 58, line 44, in Section 3.9. The text should read

be invoked in a sequence of the form,

```
Create (Start Complete)* Free
```

where `*` indicates zero or more repetitions. If the same communication ...