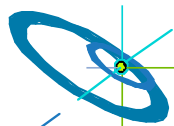


MPI 2.1 at MPI Forum Chicago, March 10-12, 2008 Ballot 4

Rolf Rabenseifner
rabenseifner@hlrs.de
(Chairman of MPI 2.1 Task)

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de



MPI 2.1
Slide 1

Höchstleistungsrechenzentrum Stuttgart



MPI 2.1 – Ballot 4 – Goals

Scope of Effort:

- Clarification to the MPI standards document,
- i.e., defining the outcome of MPI function if the current definition is not clear,
- not changing the current definition if it is “bad” (→ this may be task of MPI 2.2-3.0)

Working plan for Ballot 4:

- This meeting:
 - Monday: Official reading and discussion if necessary (2:00-3:00, 3:15-5:15pm)
 - If additional discussions are needed: 9:00-10:00pm
 - Tuesday: Official reading of such changes
 - Tuesday: Straw votes on each Ballot 4 item (9:00-9:30am).




Not later than Monday evening:

You all must be **sure**, that you will vote on each item with “yes”
only, if it is okay!

If necessary, please discuss it with some members of the Forum.

If there is still a problem, please contact me before Monday 9:00pm.

Ballot 4 - Categories

- Numbering of items: same as in <http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/>
- Sequence in the meeting:
 - **S** Small and simple (Items 1, 2a-g, 3, 5, 6, 10e, 11c, 12, 14, 15a-b, 18, 19, 21, X) 
 - **L** Long, but still simple (Items 4, 8, 13, 17, 20) 
 - **D** Decisions are necessary (Items 7, 10a-d, 11a-b, 16, 22, 9) 
 - **R** May be relevant for implementations that have not implemented this decision (Items 4, 6, 7, 9, 10a-d, 11a-b, 13, 16, 17, 18, 20)

S

Ballot 4, Item 1 – Incorrect use of MPI_IN_PLACE in description of MPI_ALLGATHER and MPI_ALLGATHERV

Question:

Do you accept this entry?

Yes:

All=32

No:

0

Abstain:

0

[Mail discussion](#), proposed by Jeff Squyres, Sep. 15, 2005

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/gatherinplace/>

On MPI-2.0, page 158, lines 25-31, **remove** the text

Specifically, the outcome of a call to MPI_ALLGATHER in the "in place" case is as if all processes executed n calls to

MPI_GATHER(MPI_IN_PLACE, 0, MPI_DATATYPE_NULL, recvbuf, recvcnt, recvtpe, root, comm)

for root = 0, ..., n - 1.

On MPI-2.0, page 159, lines 23-28, **remove** the text

Specifically, the outcome of a call to MPI_ALLGATHER in the "in place" case is as if all processes executed n calls to

MPI_GATHERV(MPI_IN_PLACE, 0, MPI_DATATYPE_NULL, recvbuf, recvcnt, displs, recvtpe, root, comm)

~~for root = 0, ..., n - 1.~~

Reason:

The text is wrong. The text is not needed because the outcome of MPI_ALLGATHER and MPI_ALLGATHERV is already specified and this analogy is not needed.





Ballot 4, Item 2.a – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

All

No:

0

Abstain:

0

[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

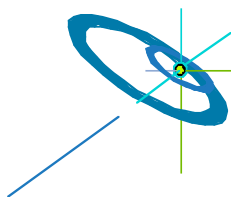
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI-2.0, Sect. 4.1, page 37, **remove** line 44-46:

This is advice to implementors, rather than a required part of MPI-2. It is not suggested that this be the only way to start MPI programs. If an implementation does provide a command called `mpiexec`, however, it must be of the form described here.

Rationale for this remove:

It is largely a repetition of [lines 34-36](#) (except the statement "It is not suggested that this be the only way..."): **Instead, MPI specifies an `mpiexec` startup command and recommends but does not require it, as advice to implementors. However, if an implementation does provide a command called `mpiexec`, it must be of the form described below.**





Ballot 4, Item 2.b – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

All-1

No:

0

Abstain:

1

[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI-2.0, Sect. 4.10, page 43, line 34:

Replace

"It consists of (key, value) pairs"

by

"It stores an unordered set of (key, value) pairs"

MPI-2.0, Sect. 4.10, page 43, line 34:

Replace

"A key may have only one value."

by

"A key can have only one value."

Rationale: To emphasize that the info object is a kind of dictionary (data structure)



S

Ballot 4, Item 2.c – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

All=33

No:

0

Abstain:

0

[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI-2.0, Sect. 4.11, page 49, **add** after line 21:

```
/* no memory is allocated */
```

MPI-2.0, Sect. 4.11, page 49, **add** after line 22:

```
/* memory allocated */
```

Rationale:

To make consistent with Fortran example, add these comments.



S

Ballot 4, Item 2.d – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

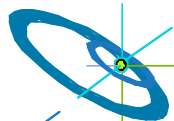
[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI-2.0, Sect. 4.12.2, page 50, line 9: **Remove** first

"in"

Reason: Typo



S

Ballot 4, Item 2.e – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

All-3

No:

0

Abstain:

3

[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI 2.0, Sect. 4.12.6, Exa. 4.12, page 55, line 21-22 read:

```
INTEGER TYPE, IERR  
INTEGER (KIND=MPI_ADDRESS_KIND) ADDR
```

but should read:

```
INTEGER TYPE, IERR, AOBLN(1), AOTYPE(1)  
INTEGER (KIND=MPI_ADDRESS_KIND) AODISP(1)
```

MPI 2.0, Sect. 4.12.6, Exa. 4.12, page 55, line 25-26 read:

```
CALL MPI_GET_ADDRESS( R, ADDR, IERR)  
CALL MPI_TYPE_CREATE_STRUCT(1, 5, ADDR, MPI_REAL, TYPE, IERR)
```

but should read:

```
AOBLN(1) = 5  
CALL MPI_GET_ADDRESS( R, AODISP(1), IERR)  
AOTYPE(1) = MPI_REAL  
CALL MPI_TYPE_CREATE_STRUCT(1, AOBLN, AODISP, AOTYPE, TYPE, IERR)
```

(Text changed in March 2008 meeting)

Reason: It was bad Fortran style and hard to read for C programmer.

S

Ballot 4, Item 2.f – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

All-3

No:

0

Abstain:

3

[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI 2.0, Sect. 4.12.10, Exa. 4.14, page 60, line 31-32 read:

```
INTEGER TYPE, IERR, MYRANK  
INTEGER (KIND=MPI_ADDRESS_KIND) ADDR
```

but should read:

```
INTEGER TYPE, IERR, MYRANK, AOBLN(1), AOTYPE(1)  
INTEGER (KIND=MPI_ADDRESS_KIND) AODISP(1)
```

MPI 2.0, Sect. 4.12.10, Exa. 4.14, page 55, line 35-36 read:

```
CALL MPI_GET_ADDRESS( R, ADDR, IERR)  
CALL MPI_TYPE_CREATE_STRUCT(1, 5, ADDR, MPI_REAL, TYPE, IERR)
```

but should read:

```
AOBLN(1) = 5  
CALL MPI_GET_ADDRESS( R, AODISP(1), IERR)  
AOTYPE(1) = MPI_REAL  
CALL MPI_TYPE_CREATE_STRUCT(1, AOBLN, AODISP, AOTYPE, TYPE, IERR)
```

(Text changed in March 2008 meeting)

Reason: It was bad Fortran style and hard to read for C programmer.



Ballot 4, Item 2.g – Edits to MPI-2 Chapter 4, Miscellany

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

[Mail discussion](#), proposed by Jesper Larson Traeff, Jan 09, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/mpi2misc/>

MPI-2.0, Sect. 4.12.6, page 56, line 29:

"assciated"

should be

"associati**ed"**

MPI-2.0, Sect. 4.14.5, page 74, line 9: Replace

"it erroneous"

by

"it **is erroneous"**

MPI-2.0, Sect. 5.3.2, page 85, line 25: Replace

"as the** as the"**

by

"as the"

Reason: Typos





Ballot 4, Item 3 – Interlanguage use of Attributes *and* Error in Example 4.13 in MPI-2 (Use of Attributes in C and Fortran)

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

[Mail discussion](#), proposed by Nicholas Nevin (Mar.25, 1999) et al.

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/getattr/>

and [Mail discussion](#), Jeff Squyres (June 24,2005) et al.

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/attrcandf/>

Change MPI-2.0, Sect.4.12, page 58, line 36 reads:

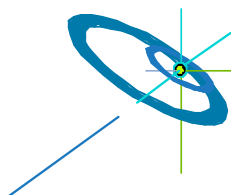
```
IF (val.NE.5) THEN CALL ERROR
```

but should read

```
IF (val.NE.address\_of\_i) THEN CALL ERROR
```

Rationale for this modification:

MPI-2.0, Sect. 4.12, page 58, lines 12-13 and 16-18 clearly state that if an attribute is set by C, retrieving it in Fortran will obtain the address of the attribute.





Ballot 4, Item 4 – What info keys can be set?

Question:

Do you
accept
this
entry?

Yes:

All-2

No:

0

Abstain:

2

[Mail discussion](http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/infoaset/), proposed by Linda Stanberry and Bill Gropp, Jun 17, 1999
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/infoaset/>
MPI 2.0, Sect. 4.10 Info Objects, page 43, line 38-40 read

If a function does not recognize a key, it will ignore it, unless otherwise specified. If an implementation recognizes a key but does not recognize the format of the corresponding value, the result is undefined.

but should read

An implementation must support info objects as caches for arbitrary (key, value) pairs, regardless of whether it recognizes the key. Each function that takes hints in the form of an MPI_Info must be prepared to ignore any key it does not recognize. This description of info objects does not attempt to define how a particular function should react if it recognizes a key but not the associated value. MPI_INFO_GET_NKEYS, MPI_INFO_GET_NTHKEY, MPI_INFO_GET_VALUELEN, and MPI_INFO_GET must retain all (key, value) pairs so that layered functionality can also use the Info object.

Rationale for this clarification: The MPI-2.0 text allowed that also MPI_INFO_DELETE, MPI_INFO_SET, MPI_INFO_GET, and MPI_INFO_DUP could ignore (key,value) pairs that are not recognized in routines in other chapters that take hints with info arguments. The proposed clarification is necessary when we assume, that layered implementation of parts of the MPI-2 standard should be possible and may use the MPI_Info objects for their needs. This was a goal of the MPI-2 Forum and the MPI-2.0 specification.

(Text changed in March 2008 meeting)



S

Ballot 4, Item 5 – Datatypes and MPI_PROBE

Question:

Do you
accept
this
entry?

Yes:

All-1

No:

0

Abstain:

1

[Mail discussion](http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/probedatatype/), proposed by Patrick H. Worley, Nov 24, 1999, and Bill Gropp
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/probedatatype/>

MPI 1.1, page 222, line 48 reads

used after a call to MPI_PROBE. (End of rationale.)

but should read

used after a call to MPI_PROBE or MPI_Iprobe. With a status returned from MPI_PROBE or MPI_Iprobe, the same datatypes are allowed as in a call to MPI_RECV to receive this message. (End of rationale.)

Advice to users. The buffer size required for the receive can be affected by data conversions and by the stride of the receive datatype. In most cases, the safest approach is to use the same datatype with MPI_GET_COUNT and the receive. (End of advice to users.)

Rationale for this clarification: Reason for the first part: The current MPI-1.1 text says "The datatype argument should match the argument provided by the receive call that set the status variable." With MPI_PROBE, there isn't such a receive call.

Reason for the advice to users: It helps to write portable code. Because malloc needs a byte count, users may write wrong programs by using MPI_BYTE.

(Text changed in March 2008 meeting)



Ballot 4, Item 6 – MPI_PROCNUL and RMA Part 2

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

[Mail discussion](#), proposed by Richard Treumann, Mar 8, 2002

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/procnul2/>

MPI-2, page 114, after line 4 (and after the lines added about `MPI_PROC_NULL`), add

After any RMA operation with rank `MPI_PROC_NULL`, it is still necessary to finish the RMA epoch with the synchronization method that started the epoch.

Reason: The behavior of one-sided RMA with target `MPI_PROC_NULL` was not clear.

(Text changed in March 2008 meeting)





Ballot 4, Item 7 – Significance of non-root arguments to MPI_COMM Spawn

Moved into MPI-2.2

[Mail discussion](#), proposed by Jeff Squyres, Jul 30, 1999, et al.

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/spawn/>

Proposal A

Add in MPI-2.0, page 88, after line 24:

Advice to users. If the non-root processes do not use MPI_ERRCODES_IGNORE, then they have to allocate the appropriate number of entries (see maxproc at root) in the array_of_errcodes although the maxproc argument is unused in non-root processes. It is allowed to use an array_of_errcodes at some of the calling processes and MPI_ERRCODES_IGNORE at some others. (End of advice to users.)

Rationale for this clarification:

It was not clear that maxproc is significant as input argument only at root while it is needed at all processes to define the length of array_of_errcodes. It was not explicitly forbidden that MPI_ERRCODES_IGNORE is used only at some processes. And there isn't a general rule that all arguments must be the same.

Question:
Which
proposal
is better?

A:

B:

Abstain:

Question:
Do you
accept
proposal
— ?

Yes:

No:

Abstain:

Ballot 4, Item 7 – Significance of non-root arguments to MPI_COMM_SPAWN

Moved into MPI-2.2

[Mail discussion](#), proposed by Jeff Squyres, Jul 30, 1999, et al.

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/spawn/>

Proposal B

MPI 2.0 page 84 line 45 reads (in MPI_COMM_SPAWN):

OUT array_of_errcodes one code per process (array of integer)

but should read:

**OUT array_of_errcodes one code per process (array of integer,
significant only at root)**

MPI 2.0 page 89 line 42 reads (in MPI_COMM_SPAWN_MULTIPLE):

OUT array_of_errcodes one code per process (array of integer)

but should read:

**OUT array_of_errcodes one code per process (array of integer,
significant only at root)**

Comment: This proposal modifies the MPI interface. User codes may be broken.

Another reason, not to do this modification, is that the non-root processes have no chance (in error-return-mode) to detect an error. And after an error, MPI does not guarantee that MPI communication still works. Only MPI_Abort should be guaranteed to work.





Ballot 4, Item 8 – Thread safety and collective communication

Question:

Do you
accept
this
entry?

Yes:

All-4

No:

0

Abstain:

4

[Mail discussion](#), proposed by Karl Feind, Nov 16, 1998, et al.

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/thread-safety/>

Add new paragraphs after MPI-2, 8.7.2 page 195 line 9 (the end of the clarification on "Collective calls"):

Advice to users. With three concurrent threads in each MPI process of a communicator `comm`, it is allowed that thread A in each MPI process calls a collective operation on `comm`, thread B calls a file operation on an existing filehandle that was formerly opened on `comm`, and thread C invokes one-sided operations on an existing window handle that was also formerly created on `comm`. (End of advice to users.)

Rationale. As already specified in `MPI_FILE_OPEN` and `MPI_WIN_CREATE`, a file handle and a window handle inherit only the group of processes of the underlying communicator, but not the communicator itself. Accesses to communicators, window handles and file handles cannot affect one another. (End of rationale.)

Advice to implementors. If the implementation of file or window operations internally uses MPI communication then a duplicated communicator may be cached on the file or window object. (End of advice to implementors.)

(Text changed in March 2008 meeting)

Rationale for this clarification:

The emails have shown, that the current MPI-2 text can be misunderstood.



Ballot 4, Item 9 – const in C++ specification of predefined MPI objects (was just datatypes)

Item 22.A : **INOUT** → Variant A/D: Same as voted for Bal. 3

[Mail discussion](#), by Richard Treumann and Rolf Rabenseifner, Jun 13 – Jul 26, 2001
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/>

Moved into MPI-2.2

Question:

Do you accept this entry?

Yes:

No:

Abstain:

- MPI-2, page 345, line 37: **Remove** the const from **const MPI::Op**.
 - MPI-2, page 346, line 20: **Remove** the const from **const MPI::Group**.
 - MPI-2, page 346, add after line 34:
Advice to implementors: If an implementation does not change the value of predefined handles while execution of MPI_Init, the implementation is free to define the predefined operation handles as const MPI::Op and the predefined group handle MPI::GROUP_EMPTY as const MPI::Group. Other predefined handles must not be "const" because they are allowed as INOUT argument in the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR routines. (End of advice to implementors.)
-
- Reason: MPI_Init may change the predefined handles, because MPI 1.1, page 10, lines 9-10 says: "Opaque objects accessed by constant handles are defined and do not change value between MPI initialization (MPI_INIT() call) and MPI completion (MPI_FINALIZE() call)." Therefore they must not be defined as const in the MPI standard.
-
- Alternative A: Keep INOUT for handles with call by value in C/C++
In this case, the text above should be used (as already proposed for Ballot 3)

Ballot 4, Item 9 – const in C++ specification of predefined MPI objects (was just datatypes)

Item 22.B : **IN/INOUT** → Variant B: Modifications in Advice

[Mail discussion](#), by Richard Treumann and Rolf Rabenseifner, Jun 13 – Jul 26, 2001
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/>

Moved into MPI-2.2

Question:

Do you accept this entry?

Yes:

No:

Abstain:

- MPI-2, page 345, line 37: **Remove** the const from **const MPI::Op**.
- MPI-2, page 346, line 20: **Remove** the const from **const MPI::Group**.
- MPI-2, page 346, add after line 34:

Advice to implementors: If an implementation does not change the value of predefined handles while execution of MPI_Init, the implementation is free to define the predefined operation handles as const MPI::Op and the predefined group handle MPI::GROUP_EMPTY as const MPI::Group. ~~Other predefined handles must not be "const" because they are allowed as INOUT argument in the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR routines.~~
(End of advice to implementors.)

- Alternative B: Change into IN/INOUT (i.e. mark handles as IN and referenced opaque object as INOUT)
In this case, **remove** the last sentence from the proposal:
"Other predefined handles must not be "const" because they are allowed as INOUT argument in the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR routines."

Ballot 4, Item 9 – const in C++ specification of predefined MPI objects (was just datatypes)

Item 22.B : IN → Variant C: Modifications in Advice

[Mail discussion](#), by Richard Treumann and Rolf Rabenseifner, Jun 13 – Jul 26, 2001
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/>

Moved into MPI-2.2

Question:

Do you accept this entry?

Yes:

No:

Abstain:

- MPI-2, page 345, line 37: **Remove** the const from **const MPI::Op**.
- MPI-2, page 346, line 20: **Remove** the const from **const MPI::Group**.
- MPI-2, page 346, add after line 34:

Advice to implementors: If an implementation does not change the value of predefined handles while execution of MPI_Init, the implementation is free to define the predefined operation handles as const MPI::Op and the predefined group handle MPI::GROUP_EMPTY as const MPI::Group. ~~Other predefined handles must not be "const" because they are allowed as INOUT argument in the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR routines.~~
(End of advice to implementors.)

- Alternative C: Change into IN (i.e. mark handles as IN and forget about referenced opaque object as)
In this case, **remove** the last sentence from the proposal:
"Other predefined handles must not be "const" because they are allowed as INOUT argument in the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR routines."





Ballot 4, Item 10.a – Questions about `Cart_create`, and `Cart_coords`

[Mail discussion](#), proposed by Jesper L. Traeff, Mar 4, 2002, and Rolf Rabenseifner
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/topo/>

Here is a proposal about handling zero-dimensional Cartesian communicators that are produced with `MPI_Cart_sub` if all `remain_dims` are false.

MPI-1.1 Sect.6.5.4, page 187, line 42 (end of definition of `MPI_Cart_sub`) reads
(This function is closely related to `MPI_COMM_SPLIT`.)

but should read

If all entries in `remain_dims` are false or `comm` is already associated with a zero-dimensional Cartesian topology then `newcomm` is associated with a zero-dimensional Cartesian topology. (This function is closely related to `MPI_COMM_SPLIT`.)





Ballot 4, Item 10.b – Questions about `Cart_create`, and `Cart_coords`

MPI-1.1 Sect.6.5.4, page 183, **add** at end of lines 30 (definition of `MPI_Cartdim_get` and `MPI_Cart_get`):

If `comm` is associated with a zero-dimensional Cartesian topology, `MPI_Cartdim_get` returns `ndims=0` and `MPI_Cart_get` will keep all output arguments unchanged.

MPI-1.1 Sect.6.5.4, page 184, **add** a new paragraph after line 23 (definition of `MPI_Cart_rank`):

If `comm` is associated with a zero-dimensional Cartesian topology, `coord` is not significant and 0 is returned in `rank`.

MPI-1.1 Sect.6.5.4, page 184, **add** a new paragraph after lines 39 (definition of `MPI_Cart_coords`):

If `comm` is associated with a zero-dimensional Cartesian topology, `coords` will be unchanged.





Ballot 4, Item 10.c – Questions about Cart_create, and Cart_coords

Question:

Which
alternat.
is better?

A:

all

B:

0

Abstain:

0

Alternative A:

MPI-1.1 Sect.6.5.5, page 186, after line 47 (end of definition of `MPI_Cart_shift`), the following paragraph is **added**:

It is erroneous to call `MPI_CART_SHIFT` with a direction that is either negative or greater than or equal to the number of dimensions in the Cartesian communicator. This implies that it is erroneous to call `MPI_CART_SHIFT` with a `comm` that is associated with a zero-dimensional Cartesian topology.

Alternative B:

MPI-1.1 Sect.6.5.5, page 186, after line 47 (end of definition of `MPI_Cart_shift`), the following paragraph is **added**:

If `comm` is associated with a zero-dimensional Cartesian topology, then the input arguments `direction` and `disp` are ignored and always `MPI_PROC_NULL` is returned in `rank_source` and `rank_dest`.

(End of Alternative A and B)

Rationale: A is directly in the spirit of the current definition.



Ballot 4, Item 10.d – Questions about Cart_create, and Cart_coords

Question:

Do you
accept
total
Item
10.a-d ?

Yes:

All-1

No:

0

Abstain:

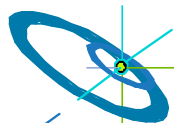
1

MPI-1.1 Sect.6.5.1, page 179, lines 29-30 (end of definition of `MPI_Cart_create`) reads

The call is erroneous if it specifies a grid that is larger than the group size.
but should read

If ndims is zero then a zero-dimensional Cartesian topology is created. The call is erroneous if it specifies a grid that is larger than the group size or if ndims is negative.

(Text changed in March 2008 meeting)



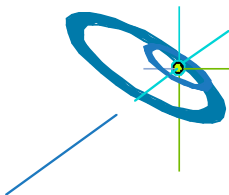
Ballot 4, Item 10.a-d – Background information (1)

This report shows significant differences of MPI implementations in the (extreme) case of zero-dimensional topologies!

Therefore clarifications may be necessary.

(Although hopefully never an application may produce zero-dimensional Cartesian topologies!)

If there are no objections, I would produce clarifications for MPI-2.1 according to the behavior of mpich2 and MPI from IBM (except the rank -767705708, see below).



Ballot 4, Item 10.a-d – Background information (2)

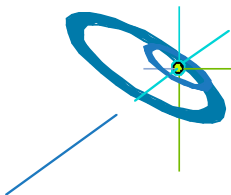
MPI-1.1 Sect. 6.5.6, page 187, routine MPI_CART_SUB defines on lines 38-42:

If a cartesian topology has been created with MPI_CART_CREATE, the function MPI_CART_SUB can be used to partition the communicator group into subgroups that form lower-dimensional cartesian subgrids, and to build for each subgroup a communicator with the associated subgrid cartesian topology. (This function is closely related to MPI_COMM_SPLIT.)

The text clearly says, that the new communicator must

- (1) have a Cartesian topology associated
- (2) be lower-dimensional in the case of a subgrid.

There is no restriction on the input array remain_dims. Therefore all MPI implementations (that I tested) allow that all entries in remain_dims are "false".



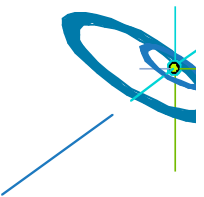
Ballot 4, Item 10.a-d – Background information (3)

I tested several MPI libraries with creating a subgrid with `MPI_Cart_sub(remain_dims=0)` from a 1-dim Cartesian topology.

I tested the subgrid communicator with `MPI_Topo_test`.

In the case of `MPI_CART`, I used `MPI_Cartdim_get` to retrieve `ndims`, and `MPI_Cart_get` for further details:

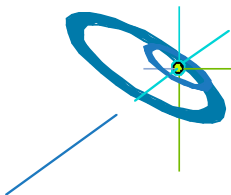
- mpich2 (1.0.3):
 `MPI_CART`, `ndims=0`,
 `MPI_Cart_get` works, but keeps all OUT arguments unchanged
- IBM (on SP):
 `MPI_CART`, `ndims=0`,
 `MPI_Cart_get` works, but keeps all OUT arguments unchanged
- OpenMPI 1.2.4:
 `MPI_CART`, `ndims=1`,
 `MPI_Cart_get` works and returns `dims=1`, `periods=0`, `coords=0`
 independent from process or periods in the original comm
 (may be wrong because (2) is not fulfilled).
- NEC MP/EX: not `MPI_CART` (may be wrong because (1) is not fulfilled)
- Voltaire: not `MPI_CART` (mpich1) (may be wrong because (1) is not fulfilled)



Ballot 4, Item 10.a-d – Background information (4)

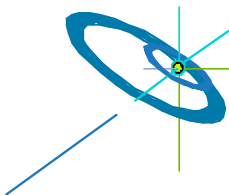
With the implementations that return a correct zero-dim Cartesian topology, I tested further usage of this zero-dim communicator:

- `MPI_Comm_size` returns 1 and `MPI_Comm_rank` returns 0 because this communicator is like `MPI_COMM_SELF`, but with Cartesian topology associated.
- `MPI_Cart_rank(IN ZeroDimComm, IN coords=0, OUT rank)`
Rationale. A zero-dim communicator has zero coords, i.e., this routine should not examine the coords input argument.
 - mpich2:
rank = 0 is returned (may be because this is the only existing rank in this communicator, this value may make sense, independent of the coord, that should not be analyzed)
 - IBM:
rank = -767705708 is returned (strange value, not `MPI_PROC_NULL`, not `MPI_UNDEFINED`)



Ballot 4, Item 10.a-d – Background information (5)

- MPI_Cart_coords(IN ZeroDimComm, IN rank=0, OUT coords)
Rationale. A zero-dim communicator has zero coords, i.e., this routine should not return anything in the coords output argument.
 - mpich2 and IBM: coords is not modified (as expected)
 - MPI_Cart_sub(IN ZeroDimComm, IN remain_dims=0, OUT subsubcomm)
Rationale. A zero-dim communicator has zero dimensions, i.e., this routine should not examine remain_dims and the returned communicator should be again a zero-dim Cartesian communicator.
 - mpich2 and IBM: subsubcomm is a zero-dim Cartesian communicator (as expected)
 - MPI_Cart_shift(IN ZeroDimComm, IN direction=0, IN disp=1, OUT src, OUT dest)
Rationale. This call is erroneous because in a zero-dim communicator, the direction=0 does not exist.
 - mpich2 and IBM: They detect the error and abort.
- (In OpenMPI, all these calls work as expected on a 1-dimensional topology on a "MPI_COMM_SELF")



Ballot 4, Item 10.a-d – Background information (6)

The last test is not addressed by MPI-1.1:

Is it possible to build a zero-dim Cartesian topology directly by calling MPI_Cart_create:

- MPI_Cart_create(IN MPI_COMM_SELF, IN ndims=0, IN dims=1, IN Periods, IN reorder, OUT ZeroDimComm)
 - Results: All tested MPI implementations return an error and abort.
(Same on MPI_COMM_WORLD)





Ballot 4, Item 10.e – Questions about Cart_create, and Cart_coords

[Mail discussion](#), proposed by Jesper L. Traeff, Mar 4, 2002, and Rolf Rabenseifner
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/topo/>

Question:

Do you
accept
this typo
correction
Item
10.e?

Yes:

all

No:

0

Abstain:

0

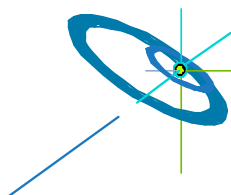
Typo correction:

MPI-1.1 Sect.6.5.4, page 184, lines 30 (definition of `MPI_Cart_coords`) reads

IN maxdims length of vector coord in the calling program (integer)

but should read (missing "s" at coords)

IN maxdims length of vector coords in the calling program (integer)





Ballot 4, Item 11.a – Questions about Graph_create

Question:

Do you
accept
this
entry?

Yes:

All-1

No:

0

Abstain:

1

[Mail discussion](#), proposed by Jesper L. Traeff, Mar 4, 2002, and Rolf Rabenseifner
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/topo/MPI-1.1>, Sect. 6.5.3, page 181, line 1-3 read:

If the size, `nnodes`, of the graph is smaller than the size of the group of `comm`, then some processes are returned `MPI_COMM_NULL`, in analogy to `MPI_CART_CREATE` and `MPI_COMM_SPLIT`.

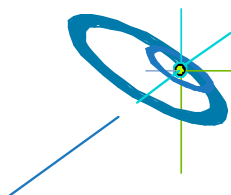
but should read

If the size, `nnodes`, of the graph is smaller than the size of the group of `comm`, then some processes are returned `MPI_COMM_NULL`, in analogy to `MPI_CART_CREATE` and `MPI_COMM_SPLIT`. *If the graph is empty, i.e., `nnodes == 0`, then `MPI_COMM_NULL` is returned in all processes.*

Rationale for this clarification:

As in `MPI_COMM_CREATE`, empty groups are allowed, but empty groups are described here in a different way, and should be mentioned explicitly therefore.

Empty graphs have been allowed in the past





Ballot 4, Item 11.b – Questions about Graph_create

Question:

Do you
accept
this
entry?

Yes:

All-2=30

No:

0

Abstain:

2

After MPI-1.1, Sect. 6.5.3, page 181, line 35, the following paragraph should be **added**:

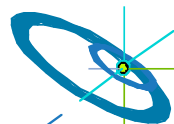
A single process is allowed to be defined multiple times in the list of neighbors of a process (i.e., there may be multiple edges between two processes). A process is also allowed to be a neighbor to itself (i.e., a self loop in the graph). The adjacency matrix is allowed to be non-symmetric.

Advice to users. Performance implications of using multiple edges or a non-symmetric adjacency matrix are not defined. The definition of a node-neighbor edge does not imply a direction of the communication. (End of advice to users.)

Rationale for this clarification:

The Example 6.3, MPI-1.1, page 15, line 29 - page 186, line 13, clearly shows multiple edges between nodes and self loops: the two (multiple) self-loops of node 0 and of node 7. It is nowhere forbidden, that the graph has edges only in one direction.

(Text modified at March 2008 meeting)



S

Ballot 4, Item 11.c – Questions about Graph_create

Question:

Do you
accept
this
entry?

Yes:

All=32

No:

0

Abstain:

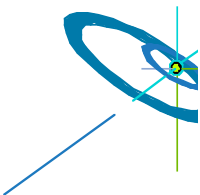
0

After MPI-1.1, Sect. 6.4, page 178, end of the sentence on lines 6-7, the following sentence should be **added**:

All input arguments must have identical values on all processes of the group of `comm_old`.

Rationale for this clarification:

This statement is missing.





Ballot 4, Item 12 – Why no MPI_INPLACE for MPI_EXSCAN?

Question:

Do you
accept
this
entry?

Yes:

All-1=30

No:

1

Abstain:

0

[Mail discussion](http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/exscan/), proposed by Falk Zimmermann and Tony Skjellum, Mar 19, 1999
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/exscan/>

MPI-2, Sect. 7.3.6, page 167, lines 6-8 read:

The reason that MPI-1 chose the inclusive scan is that the definition of behavior on processes zero and one was thought to offer too many complexities in definition, particularly for user-defined operations. (End of rationale.)

but should read:

No in-place version is specified for MPI_EXSCAN because it is not clear what this means for the process of rank zero.

The reason that MPI-1 chose the inclusive scan is that the definition of behavior on processes zero and one was thought to offer too many complexities in definition, particularly for user-defined operations. (End of rationale.)

(Text modified at March 2008 meeting)

MPI 2.1
Slide 36

Rolf Rabenseifner
Hochleistungsrechenzentrum Stuttgart

H L R I S





Ballot 4, Item 13 – MPI_GET_PROCESSOR_NAME and Fortran

Question:

Do you
accept
this
entry?

Yes:

All-1=32

No:

0

Abstain:

1

[Mail discussion](#), proposed by William Gropp et al., Dec 09, 1999

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/procname/>

Add the following clarification to the current interface definitions of `MPI_GET_PROCESSOR_NAME` and `MPI_COMM_GET_NAME`.

MPI 1.1, Sect. 7.1, routine `MPI_GET_PROCESSOR_NAME`, page 193, **add** after line 20:

In C, a null character is additionally stored at `name[resultlen]`. `resultlen` cannot be larger than `MPI_MAX_PROCESSOR_NAME-1`. In Fortran, `name` is padded on the right with blank characters. `resultlen` cannot be larger than `MPI_MAX_PROCESSOR_NAME`.

MPI-1.1, Sect. 7.1, page 193, beginning of line 29 reads

examine the ouput argument

but should read (additional t in output)

(Text modified at March 2008 meeting)

examine the output argument

MPI 2.0, Sect. 8.4, routine `MPI_COMM_GET_NAME`, page 178, **add** after line 48:

In C, a null character is additionally stored at `name[resultlen]`. `resultlen` cannot be larger than `MPI_MAX_OBJECT_NAME-1`. In Fortran, `name` is padded on the right with blank characters. `resultlen` cannot be larger than `MPI_MAX_OBJECT_NAME`.

There is additional information, including results of checking the behavior of many MPI implementations, in the mail discussion. The test programs and some results are also available here:

- [mpi_comm_get_name_test_protocol.txt](#), [mpi_comm_get_name_test.c](#), [mpi_comm_get_name_test.f](#)
- [mpi_get_processor_name_test.c](#), [mpi_get_processor_name_test.f](#), [mpi_get_processor_name_test_protocol.txt](#)



S

Ballot 4, Item 14 – Interpretation of user defined datarep

Question:

Do you
accept
this
entry?

Yes:

All=33

No:

0

Abstain:

0

[Mail discussion](#), proposed by Hubert Ritzdorf et al., Feb 02, 1999

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/datarep/>

MPI-2.0 Sect. 9.5.3 User-defined Data Representations, page 254, lines 13-15 read:

Then in subsequent calls to the conversion function, MPI will increment the value in `position` by the count of items converted in the previous call.

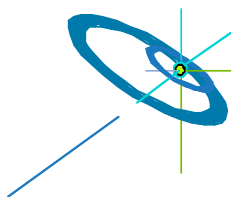
but should read:

Then in subsequent calls to the conversion function, MPI will increment the value in `position` by the count of items converted in the previous call, and the `userbuf` pointer will be unchanged.

Rationale for this clarification:

It was not clear, whether the `userbuf` pointer must also be moved in the subsequent calls. This clarification was already done in 1999 and should already be implemented in existing implementations of user-defined data representations.

(Text changed in March 2008 meeting)



S

Ballot 4, Item 15.a – MPI_Abort

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

[Mail discussion](#), proposed by William Gropp et al., Jan 31, 2001

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/abort/>

MPI-1.1 Sect. 7.5, MPI_Abort, page 200, lines 23-26 read:

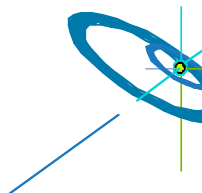
This routine makes a "best attempt" to abort all tasks in the group of `comm`. This function does not require that the invoking environment take any action with the error code. However, a Unix or POSIX environment should handle this as a `return errorcode` from the main program **or an `abort(errorcode)`.**

but should read (" or an `abort(errorcode)`" removed):

This routine makes a "best attempt" to abort all tasks in the group of `comm`. This function does not require that the invoking environment take any action with the error code. However, a Unix or POSIX environment should handle this as a `return errorcode` from the main program.

Rationale for this clarification:

POSIX defines `void abort(void)`. The routine `void exit(int status)` may be used to implement "handle this as a return errorcode from the main program". `abort(errorcode)` was not substituted by `exit(errorcode)` because this is technically not enough, if the MPI implementation wants to return it also from `mpiexec`, see next proposal.





Ballot 4, Item 15.b – MPI_Abort

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

[Mail discussion](#), proposed by William Gropp et al., Jan 31, 2001

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/abort/>

MPI-1.1 Sect. 7.5, MPI_Abort, page 200, **add** after line 34 (end of rationale):

Advice to users. Whether the errorcode is returned from the executable or from the MPI process startup mechanism (e.g., mpiexec), is an aspect of quality of the MPI library but not mandatory. (End of advice to users.)

Advice to implementors. Where possible, a high quality implementation will try to return the errorcode from the MPI process startup mechanism (e.g. mpiexec or singleton init). (End of advice to implementors.)

Rationale for this clarification:

The intent of word "should" in "should handle this as a return errorcode from the main program" is only a quality of implementation aspect and not a must. This was not clear and could be misinterpreted.

(Text modified at March 2008 meeting)

MPI 2.1
Slide 40

Rolf Rabenseifner
Höchstleistungsrechenzentrum Stuttgart

H L R I S





Ballot 4, Item 16 – MPI_Type_create_f90_real etc.

Question:

Do you
accept
this
entry?

Yes:

All-5

No:

1

Abstain:

4

[Mail discussion](#), proposed by Nicholas Nevin et al., Feb 09, 2001

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/typef90real/>

Problem:

An application may repeatedly call (probably with same (p,r) combination) the MPI_TYPE_CREATE_F90_xxxx routines.

Proposal: **Add** after MPI-2.0 Sect. 10.2.5, MPI_TYPE_CREATE_F90_xxxx, page 295, line 47 (End of advice to users.):

Advice to implementors. An application may often repeat a call to MPI_TYPE_CREATE_F90_xxxx with the same combination of (xxxx, p, r). The application is not allowed to free the returned predefined, unnamed datatype handles. To prevent the creation of a potentially huge amount of handles, the MPI implementation should return the same datatype handle for the same (REAL/COMPLEX/INTEGER, p, r) combination. Checking for the combination (p, r) in the preceding call to MPI_TYPE_CREATE_F90_xxxx and using a hash-table to find formerly generated handles should limit the overhead of finding a previously generated datatype with same combination of (xxxx, p, r). (End of advice to implementors.)

Rationale for this clarification: (Text changed in March 2008 meeting)

Currently most MPI implementations are handling the MPI_TYPE_CREATE_F90_xxxx functions wrong or not with the requested quality.





Ballot 4, Item 17 – MPI_File_get_info

Question:

Do you
accept
this
entry?

Yes:

All-1=31

No:

0

Abstain:

1

[Mail discussion](#), proposed by Rolf Rabenseifner, Jan. 31, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/filegetinfoNULL/>

Add text stating that the return value can be null:

MPI-2.0 Sect. 9.2.8, File Info, page 219, lines 11-13 read:

MPI_FILE_GET_INFO returns a new info object containing the hints of the file associated with `fh`. The current setting of all hints actually used by the system related to this open file is returned in `info_used`.

The user is responsible for freeing `info_used` via `MPI_INFO_FREE`.

but should read:

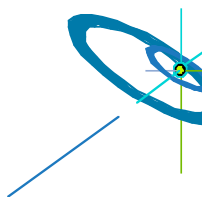
MPI_FILE_GET_INFO returns a new info object containing the hints of the file associated with `fh`. The current setting of all hints actually used by the system related to this open file is returned in `info_used`.

***If no such hints exist, a handle to a newly created info object is returned that contains no key/value pair.* The user is responsible for freeing `info_used` via `MPI_INFO_FREE`.**

Rationale for this clarification:

(Text modified at March 2008 meeting)

This text was missing. It was not clear, whether a MPI_Info handle would be returned that would return `nkeys=0` from `MPI_INFO_GET_NKEYS`. From user's point of view, this behavior might have been expected without this clarification. For most implementations, this clarification is irrelevant because they always return several default hints. e.g.. the filename.





Ballot 4, Item 18 – MPI_File_set_view

Question:

Do you
accept
this
entry?

Yes:

All-1=32

No:

1

Abstain:

0

[Mail discussion](#), proposed by Rolf Rabenseifner, Jan. 31, 2008

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/filesetview/>

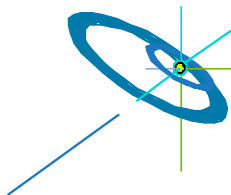
Does the info passed to MPI_File_set_view replace all of the previous info keys?
(The answer given in this clarification is "no".)

Proposal: **Add** in MPI-2.0 Sect. 9.2.8, File Info, page 218, after line 18 the following sentences:

When an info object that specifies a subset of valid hints is passed to MPI_FILE_SET_VIEW or MPI_FILE_SET_INFO, there will be no effect on previously set or defaulted hints that the info does not specify.

Rationale for this clarification:

This text was missing. It was not clear, whether an info object in MPI_FILE_SET_VIEW and MPI_FILE_SET_INFO was intended to replace only the mentioned hints or was intended to substitute a complete new set of hints for the prior set.





Ballot 4, Item 19 – MPI_IN_PLACE for MPI_Reduce_scatter

Question:

Do you
accept
this
entry?

Yes:

All=33

No:

0

Abstain:

0

[Mail discussion](#), proposed by Rajeev Thakur, Oct. 10, 2002

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/redscat/>

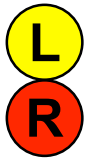
MPI-2.0, Sect. 7.3.3, routine MPI_REDUCE_SCATTER, page 163, **delete** the sentence on line 19-20:

Note that the area occupied by the input data may be either longer or shorter than the data filled by the output data.

Rationale for this clarification:

The sentence makes no sense because the input data can never be shorter than the output data. The output, determined by `recvcounts[i]`, is a subset of the input.





Ballot 4, Item 20 – Blocklengths of zero in MPI_TYPE_STRUCT and in MPI Datatypes

[Mail discussion](#), proposed by Jesper Larsson Traeff, April 23, 2007

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/blklenzero/>

And [Mail discussion](#), proposed by Peter Ganster and Bill Gropp, Jul. 25 2001

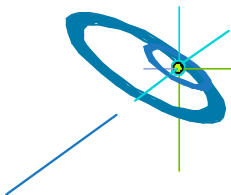
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/structblocklen/>

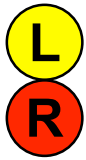
Problem:

Blocklengths of zero are allowed. Do we need to add a statement to this effect? The mail threads contain a deeper discussion of the implications of zero-length blocks in MPI datatypes.

Rationale for the clarification and modification on the following slides:

The outcome of zero-count entries in the type map was not defined. For this, a clarification was needed. The interfaces of `MPI_TYPE_CREATE_HINDEXED` and `MPI_TYPE_CREATE_STRUCT` was inconsistent to the rest derived datatype routines. This was probably due to editing errors. A meaning of negative values was never defined not intended. Therefore, portable applications could not use negative values. These editing errors are fixed by this proposal.





Ballot 4, Item 20 – Blocklengths of zero in MPI_TYPE_STRUCT and in MPI Datatypes

Proposal:

Add the following paragraph in MPI 1.1, Sect. 3.12, page 62, after line 2 (i.e., after ... "of the types defined by Typesig."):

Most datatype constructors have replication count or block length arguments. Allowed values are nonnegative integers. If the value is zero, no elements are generated in the type map and there is no effect on datatype bounds or extent.

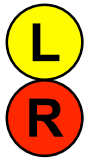
MPI 1.1, Sect 3.12.1, MPI_TYPE_HINDEXED, page 67, line 22-24 read:

IN count	number of blocks - also number of entries in array_of_displacements and array_of_blocklengths (integer)
-----------------	--

but should read:

IN count	number of blocks - also number of entries array_of_displacements and array_of_blocklengths (nonnegative integer)
-----------------	--





Ballot 4, Item 20 – Blocklengths of zero in MPI_TYPE_STRUCT and in MPI Datatypes

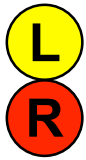
MPI 1.1, Sect 3.12.1, MPI_TYPE_STRUCT, page 68, line 19-22 read:

IN count	number of blocks (integer) - also number of entries in arrays array_of_types, array_of_displacements and array_of_blocklengths
IN array_of_blocklength	number of elements in each block (array of integer)

but should read:

IN count	number of blocks (nonnegative integer) - also number of entries in arrays array_of_types, array_of_displacements and array_of_blocklengths
IN array_of_blocklength	number of elements in each block (array of nonnegative integer)





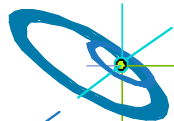
Ballot 4, Item 20 – Blocklengths of zero in MPI_TYPE_STRUCT and in MPI Datatypes

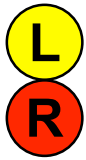
MPI 2.0, Sect 4.14.1, MPI_TYPE_CREATE_HINDEXED, page 66, line 36-38 read:

IN count	number of blocks - also number of entries in array_of_displacements and array_of_blocklengths (integer)
-----------------	--

but should read:

IN count	number of blocks - also number of entries array_of_displacements and array_of_blocklengths (nonnegative integer)
-----------------	--





Ballot 4, Item 20 – Blocklengths of zero in MPI_TYPE_STRUCT and in MPI Datatypes

Question:

Do you accept this Item 20?

Yes:

all

No:

0

Abstain:

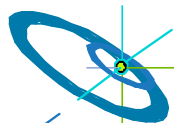
0

MPI 2.0, Sect 4.14.1, MPI_TYPE_CREATE_STRUCT, page 67, line 14-18 read:

IN count	number of blocks (integer) - also number of entries in arrays array_of_types, array_of_displacements and array_of_blocklengths
IN array_of_blocklength	number of elements in each block (array of integer)

but should read:

IN count	number of blocks (nonnegative integer) - also number of entries in arrays array_of_types, array_of_displacements and array_of_blocklengths
IN array_of_blocklength	number of elements in each block (array of nonnegative integer)



S

Ballot 4, Item 21 – Which thread is the funneled thread?

Question:

Do you
accept
this
entry?

Yes:

All=33

No:

0

Abstain:

0

[Mail discussion](#), proposed by Greg Lindahlet al., Nov. 30, 2007

<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/funneled/>

MPI-2.0 Sect. 8.7.3, MPI_Init_thread, page 196, lines 25-26 read:

MPI_THREAD_FUNNELED The process may be multi-threaded,
but **only** the main thread will make MPI calls
(**all MPI calls are "funneled" to the main thread**).

but should read:

MPI_THREAD_FUNNELED The process may be multi-threaded,
but **the application must ensure that only** the main thread makes MPI calls
(**for the definition of main thread, see MPI_IS_THREAD_MAIN**).

Rationale for this clarification:

The existing document doesn't make it clear that the MPI user has to funnel the calls to the main thread; it's not the job of the MPI library. I have seen multiple MPI users confused by this issue, and when I first read this section, I was confused by it, too.



D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (1)

[Mail discussion](http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/inout/), proposed by Jeff Squyres and Rolf Rabenseifner, Jan. 21, 2008
<http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/inout/>

Problem:

There is a inconsistency between the INOUT description for handle arguments, and their usage in the language independent definitions of MPI-1.1 and MPI-2.0. There are 3 possibilities to solve this. The MPI Forum should decide, which possibility is the best.



MPI 2.1
Slide 51

Rolf Rabenseifner
Höchstleistungsrechenzentrum Stuttgart

H L R I S

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (2)

Background:

MPI 2.0 Sect. 2.3 Procedure Specification, page 6 lines 30-34 read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the **object is updated** by the procedure call, then the argument is marked **OUT**. It is marked this way **even though the handle itself is not modified** - we use the OUT attribute to denote that what the handle references is updated. Thus, in C++, **IN arguments are either references or pointers to const objects**.

Example: MPI 1.1, page 171, lines 26-34:

```
MPI_ATTR_PUT(comm, keyval, attribute_val)
    IN      comm      communicator to which attribute will be attached (handle)
    IN      keyval     key value, as returned by MPI_KEYVAL_CREATE (integer)
    IN      attribute_val attribute value

int MPI_Attr_put(MPI_Comm comm, int keyval, void* attribute_val)
```

Example: MPI 2.0, page 44, lines 36-43:

```
MPI_INFO_SET(info, key, value)
    INOUT   info      info object (handle)
    IN      key       key (string)
    IN      value     value (string)

int MPI_Info_set(MPI_Info info, char *key, char *value)
```

- Handle with call by value (IN)
- The object behind is modified (INOUT)

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (3)

List of routines with IN handles and INOUT objects:

MPI 1.1:

MPI_ATTR_PUT, MPI_ATTR_DELETE,
MPI_ERRHANDLER_SET

MPI 2.0:

MPI_INFO_SET, MPI_INFO_DELETE,
MPI_COMM_SET_ERRHANDLER, MPI_TYPE_SET_ERRHANDLER,
MPI_WIN_SET_ERRHANDLER,
MPI_GREQUEST_COMPLETE,
MPI_COMM_SET_NAME, MPI_TYPE_SET_NAME, MPI_WIN_SET_NAME,
MPI_COMM_SET_ATTR, MPI_TYPE_SET_ATTR, MPI_WIN_SET_ATTR,
MPI_COMM_DELETE_ATTR, MPI_TYPE_DELETE_ATTR, MPI_WIN_DELETE_ATTR,
MPI_FILE_SET_SIZE, MPI_FILE_PREALLOCATE,
MPI_FILE_SET_INFO, MPI_FILE_SET_VIEW,
MPI_FILE_WRITE_AT, MPI_FILE_WRITE_AT_ALL, MPI_FILE_IWRITE_AT,
MPI_FILE_READ, MPI_FILE_READ_ALL,
MPI_FILE_WRITE, MPI_FILE_WRITE_ALL,
MPI_FILE_IREAD, MPI_FILE_IWRITE,
MPI_FILE_SEEK,
MPI_FILE_READ_SHARED, MPI_FILE_WRITE_SHARED,
MPI_FILE_IREAD_SHARED, MPI_FILE_IWRITE_SHARED,
MPI_FILE_READ_ORDERED, MPI_FILE_WRITE_ORDERED,
MPI_FILE_SEEK_SHARED,
MPI_FILE_WRITE_AT_ALL_BEGIN, MPI_FILE_WRITE_AT_ALL_END,
MPI_FILE_READ_ALL_BEGIN, MPI_FILE_READ_ALL_END,
MPI_FILE_WRITE_ALL_BEGIN, MPI_FILE_WRITE_ALL_END,
MPI_FILE_READ_ORDERED_BEGIN, MPI_FILE_READ_ORDERED_END,
MPI_FILE_WRITE_ORDERED_BEGIN, MPI_FILE_WRITE_ORDERED_END,
MPI_FILE_SET_ATOMICITY, MPI_FILE_SYNC

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (4)

Question:

Do you
accept
this
entry?

Yes:

all

No:

0

Abstain:

0

Alternative A:

Keep the argument definition for handling the opaque objects (INOUT) and add the argument definition for the handles as IN.

Proposal: MPI 2.0 Sect. 2.3 Procedure Specification, page 6 lines 30-34 read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the object is updated by the procedure call, then the argument is marked OUT. It is marked this way even though the handle itself is not modified - we use the OUT attribute to denote that what the handle references is updated. Thus, in C++, IN arguments are either references or pointers to const objects.

but should read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the object is updated by the procedure call, then the argument is marked **INOUT or OUT. It is marked this way even though the handle itself is not modified - we use the **INOUT or** OUT attribute to denote that what the handle references is updated. Thus, in C++, IN arguments are either references or pointers to const objects.**

Change the three inconsistent interface definitions from IN to INOUT in MPI-1.1 - see list of MPI 1.1 routines below. (Text modified at March 2008 meeting)

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (5)

Rationale for this proposal A:

This is the minimal change to remove the existing inconsistency. Only the Fortran interfaces of three deprecated MPI-1.1 routines is modified from IN to INOUT. Due to Fortran call by reference, this has no impact for the applications. In the C interfaces, the handle argument is call by value.



D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (6)

Moved into MPI-2.2

Alternative B:

Keep the argument definition for handling the opaque objects (INOUT) and add the argument definition for the handles as IN.

Proposal: MPI 2.0 Sect. 2.3 Procedure Specification, page 6 lines 30-34 read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the object is updated by the procedure call, then the argument is marked **OUT. It is marked this way even though the handle itself is not modified - we use the OUT attribute to denote that what the handle references is updated.** Thus, in C++, IN arguments are either references or pointers to const objects.

but should read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the object is updated by the procedure call but the handle itself is not modified, then the argument is marked IN/INOUT. We use the first part (IN) to specify the use of the handle and the second part (INOUT) to specify the use of the opaque object. Thus, in C++, IN arguments are either references or pointers to const objects, **IN/INOUT arguments are references to const handles to non-const objects.**

In the listed routines, the **INOUT** handle declaration (in MPI-2.0) and the **IN** handle declaration (in MPI-1.1) is **modified** into a **IN/INOUT** handle declaration.

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (7)

Moved into MPI-2.2

Rationale for this proposal B:

I have checked the total MPI 1.1 and 2.0 standard to find all routines with an argument specification according to the following declaration pattern:

Language independent interface:

INOUT handle

C interface

MPI_handletype handle

All these routines keep the handle itself unchanged, but the opaque object is modified in a way, that with other MPI routines this change can be detected. For example, an attribute is cached or changed, a file pointer is moved, the content of a file was modified.

The current specification with IN (in MPI 1.1) or INOUT (in MPI 2.0) is inadequate and led to misinterpretation in the const declarations of the C++ interface.

It is not explicitly mentioned that IN/IN is abbreviated with IN, and OUT/OUT with OUT. (Therefore no change in all routines with pure IN and pure OUT handles/opaque objects.

This proposal changes the Fortran interface, because the handles itself are now declared as IN. The MPI-2.0 did not decide whether they are IN or INOUT. Only C/C++ interfaces specified call by value for the handles itself. This hasn't any impact for applications. It is not expected that it has any impact on any MPI implementation.

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (8)

Still
TODO!

Moved into MPI-2.2

Alternative C:

Substitute the argument definition for handling the opaque objects (INOUT) and by the argument definition for the handles (IN).

Proposal:

MPI 2.0 Sect. 2.3 Procedure Specification, page 6 lines 30-34 read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the object is updated by the procedure call, then the argument is marked **OUT**. It is marked this way even though the handle itself is not modified - we use the **OUT** attribute to denote that what the handle references is updated. Thus, in C++, IN arguments are either references or pointers to const objects.

but should read:

There is one special case - if an argument is a handle to an opaque object (these terms are defined in Section 2.5.1), and the object is updated by the procedure call but the handle itself is not modified, then the argument is marked IN. Thus, in C++, IN arguments are either references or pointers to const objects, or references to const handles to non-const objects.

In all MPI-2.0 routines from the list above, the **INOUT** handle declaration is modified into a **IN** handle declaration.

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (9)

Moved into MPI-2.2

Rationale for this proposal C:

This proposal is easier, but loses the INOUT information on the opaque object itself. As at Alternative B, this proposal changes the Fortran interface, because the handles itself are now declared as IN. The MPI-2.0 did not decide whether they are IN or INOUT. Only C/C++ interfaces specified call by value for the handles itself. This hasn't any impact for applications. It is not expected that it has any impact on any MPI implementation.

D

Ballot 4, Item 22 – Change "INOUT" to "IN" for MPI Handle Parameters in several routines (10)

Moved into MPI-2.2

Proposal A: Defining only the outcome for the opaque object:

- e.g., INOUT filehandle in write routines.
- Text in "Terms and Conventions" is **not** changed.
- Changes at 3 MPI 1.1 routines (IN → INOUT)

Proposal B: Explicitly defining the outcome for **handle and opaque object**

- e.g., IN/INOUT filehandle in write routines.
- Text in "Terms and Conventions" is changed.
- Changes at 3 MPI 1.1 (IN → IN/INOUT)
and 48 MPI 2.0 routines (OUT → IN/INOUT)

Proposal C: Defining only the outcome for the **handle itself**

- e.g., IN filehandle in write routines.
- Text in "Terms and Conventions" is changed.
- Changes at 48 MPI 2.0 routines (INOUT → IN)

Do you
accept
this
entry?

Yes:

No:

Abstain:

Question 1: Do you (additionally) want the specification of the outcome for handles?
i.e., Proposal B or C instead of A

Yes:

No:

Abstain:

Only if we have a positive decision in the first question:

Question 2: Do you want to keep the specification of outcome for the opaque objects?
i.e., Proposal B instead of C

Yes:

No:

Abstain:

Decision
is
Proposal



S**Ballot 4, Item X.1-6 – A clarification is not needed**

The following Ballot 4 items do not need a clarification.

Therefore, they are closed and will be removed from the MPI-2 Errata web-page when Ballot 4 is finished.

→ They will go to the same history web-page as all Ballot 4 items.

- X.1** Description of the send and receive count arguments to MPI_Alltoallv
- X.2** Shared File Pointers
- X.3** Overlapping buffers in collective communications
- X.4** MPI_MODE_RDONLY, MPI_MODE_RDWR, MPI_MODE_WRONLY
- X.5** MPI_FINALIZE in MPI-2 (with spawn)
- X.6** Reporting invalid handles provided to handle conversion functions

Any reasons for a veto?



MPI 2.1 – Ballot 4

- Thank you very much Ballot 4 is done for now!