MPI-3.0 Fortran Reading – Ticket #229

(MPI Forum Meeting September 2011 - Santorini)

The Fortran Working Group

Goals / New Features for New Fortran Interface

- Guaranteed compile-time argument checking
  - Unique MPI handle types
  - Explicit Interfaces
  - Instead of allowing implicit interfaces
- Fix non-blocking issues
  - And still keep useful Fortran optimizations
  - Guaranteed safe methods for nonblocking communication
  - Instead of today's "good luck" programming methods
- Fix subarray issues
  - Strided access through the Fortran subscript triplets, e.g., a(1:100:5) (instead of MPI's derived datatypes as the only method)
  - Now allowed in nonblocking, and split-collective routines
  - Instead of restricting it to blocking routines

Fortran TR29113, June 2011 meeting: WG5/J3 decided to enhance the ASYNCHRONOUS attribute for (MPI) communication.

Based on TR29113 TYPE(*), DIMENSION(..)

New Features / Goals, Continued

- Further enhancements
  - IERROR argument optional
  - MPI_ALLOC_MEM with standardized C_PTR pointers
  - New clear rules for using Fortran derived types as send/recv buffers
  - New solutions for code movement optimization problems:
    - Additional MPI routine: MPI_F_SYNC_REG
      Instead of user defined “DD” routine in MPI-2.0 – MPI-2.2
    - Using the **ASYNCHRONOUS** attribute
- Backward compatibility
  - Within mpif.h and mpi module:
    Only backward compatible enhancements
  - Further enhancements only within the new mpi_f08 module
  - Declaring mpif.h as “use is strongly discouraged”

Does not need TR29113

TR29113 is required

User's Answers (based on the SC09 MPI Forum Questionnaire)

Question 16:

Rank the following in order of importance to your MPI applications (1=most important, .. 6=least important)

- **New Fortran bindings (type safety, etc.)**
  - 838 users answered Question 16
  - 628 users answered the Fortran line of Question 16:
    - **68 (=10.8%)** answered with 1 = **most important**
    - **72 (=11.5%)** answered with 2 = **very important**
    - **78 (=12.4%)** answered with 3 = **medium important**
    - 64 (=10.2%) & 99 (=15.8%) with 4 & 5 = less important
    - 247 (=39.3%) answered with 6 = least important (expected to be mostly C users)

MPI users want to see new **safe** Fortran bindings

i.e., 35% of the 628 Fortran answers and 26% of all 838 Q16 answers are in the range medium ... most important (218 answers)

Question 2 showed that only 303 participants are "MPI application developer". We didn't asked about their programming language.

Review Results = Formal Re-Reading (Saturday)

Acknowledgements

- Thanks to all our 16 reviewers

Reinhold Bader,	Marc-Andre Hermanns,
Purushotham Bangalore,	Torsten Hoefler,
Brian Barrett,	Adam Moody,
George Bosilca,	Craig E Rasmussen,
Darius Buntinas,	Hubert Ritzdorf,
Jim Dinan,	Martin Schulz,
Dave Godell,	Jeff Squyres,
Bill Gropp,	Rajeev Thakur

(My apologies when I forget someone)

- Your work ensured that all the interfaces are now consistent with the language independent specifications.

Major changes through the review process

- The **Cray-pointer-Interface** of MPI_Alloc_mem is kept **un-deprecated**.
- In mpif.h, the linker names must **not** be shortened
  - → mpif.h cannot support BIND(C) for some choice buffer routines
  - → mpif.h cannot support the new TR 29113 quality
- We keep the **count** arguments as **INTEGER**

Only a few changes are needed (1)

- Before p13:27-31:

  Alternative 1:
- After p13:27-31:

  Alternative 2:

  The use of the INTEGER defined handles and the BIND(C) derived type handles is similar: both handles require one numerical storage unit.
- No deprecation of Cray pointers in MPI_Alloc_mem
  - Section 8.2, page 327, lines 36-37 should be removed
  - Section 15.3, page 535, lines 2-6 should be removed
- Add after p586:36:

  In some MPI routines, a buffer dummy argument is defined as ASYNCHRONOUS to guarantee passing by reference, provided that the actual argument is also defined as ASYNCHRONOUS.
- MPI_Alloc_mem and MPI_Win_allocate:
  - Rewording of the overloaded interface description, see review-slides 32-33 and 53-55 for details.
- MPI_File_write_at_all:
  - buf needs INTENT(IN)

Only a few changes are needed (2)

- Only for better readability, in a few routines the type declarations should be put into the sequence of the dummy arguments:
  - MPI_Test: flag ←→ request
  - MPI_Group_incl: n, ranks(n) ←→ group
  - MPI_Win_test: flag ←→ win
  - MPI_File_iwrite_shared: buf ←→ fh
  - MPI_Datarep_extent_function: ierror ←→ extent, extra_state
- sendtype and recvtype on same line:
  - MPI_Gather, MPI_Gatherv, MPI_Scatter, MPI_Scatterv,
  - MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Alltoallv
- The ASYNCHRONOUS attribute is needed in … for the dummy arguments:

– MPI_Igatherv:	recvcounts, displs
– MPI_Iscatterv:	sendcounts, displs
– MPI_Iallgatherv:	recvcounts, displs
– MPI_Ialltoallv:	sendcounts, sdispls, recvcounts, rdispls
– MPI_Ialltoallw:	sendcounts, sdispls, recvcounts, rdispls, sendtypes, recvtypes
– MPI_Ireduce_scatter:	recvcounts

Only a few changes are needed (3)

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 521 lines 19-22 should read (red text should be added):

> provide a mechanism through which all of the MPI defined functions, except those allowed as macros (See Section 2.6.5), may be accessed with a name shift. This requires, in C and Fortran, an alternate entry point name, with the prefix PMPI_ for each MPI function **in each provided language binding and language support method**.

Page 551, lines 7-15 should read (the red text should be added):

> The INTEGER compile-time constant MPI_SUBARRAYS_SUPPORTED is set to … The INTEGER compile-time constant MPI_ASYNCHRONOUS_PROTECTS_NONBL is set to … These constants exist with each Fortran support method, but not in the C/C++ header files. The values may be different for each Fortran support method. **All other constants and the integer values of handles must be the same for each Fortran support method.**

Page 600, lines 18-21 should read (the red text should be added):

> Advice to implementors. Callback functions need to have a language tag. This tag is set when the callback function is passed in by the library function (which is presumably different for each language **and language support method**), and is used to generate the right calling sequence when the callback function is invoked. (End of advice to implementors.)

Only a few changes are needed (4)

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 527 line 34 and in all other wrong mpi_f08 Fortran interfaces:

> Such lines will be substituted by
> For this routine, an interface within the {\tt mpi_f08} module was never defined.

Page 551 line 46 must be indented one more level

Page 553, line 14: Quotes around TR name

Page 553 line 20: .[35] → [35]

Page 553 line 26: missing **“.**

Page 554 line 36+37, and page 563 line 31: Always ‘contiguous’ and ‘simply contiguous’

> (single quotes and no italic, as on page 578)

Page 557, lines 10+13:

> Use **P**MPI_DIST_... and 6**6** characters.

Page 753, line 48, the red text should be added:

> The provided interfaces or changes will be part of the appropriate chapter when the cited ticket are voted into this chapter of the MPI standard.

Only a few changes are needed (5)

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 766 line 23, MPI_WIN_ATTACH mpi binding: **MPI_ADDRESS_SIZE → MPI_ADDRESS_KIND**

Page 770 line 38: mpi_f08 interface for Ticket #284 – shared memory alloc

→ see review-slide 56 on B.4.8 for details

Page 371 line 34 (= MPI-2.2, p314:41), the following text should be added:

In Fortran at non-root processes, the count argument must be set to a value that is consistent with the provided array_of_argv although the content of these arguments has no meaning for this operation.

Page 600 lines 24-29 should be substituted by

Advice to users. If a subroutine written in one language or Fortran support method wants to pass a callback routine including the predefined Fortran functions (e.g., MPI_COMM_NULL_COPY_FN) to another application routine written in another language or Fortran support method, then it must be guaranteed that both routines use the callback interface definition that is defined for the argument when passing the callback to an MPI routine (e.g., MPI_COMM_CREATE_KEYVAL); see also the advice to users on page 274. (End of advice to users.)

Only a few changes are needed (6)

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 745, line 47, the following sentence should be added:

> In mpi.h, the new type MPI_F08_status, and the external variables MPI_F08_STATUS_IGNORE and MPI_F08_STATUSES_IGNORE are added.

Page 563 lines 25-30 should read:

> The language features assumed-type and assumed-rank from Fortran 2008 TR 29113 [36] are available. This is required only for mpi_f08. As long as this requirement is not supported by the compiler, it is valid to build a**n** ~~preliminary~~ MPI~~-3.0 (and not later)~~ library that implements the mpi_f08 module with MPI_SUBARRAYS_SUPPORTED set to .FALSE..

Page 557 lines 14-16 should be deleted.

Page 557 lines 17-30 are reworded **to prohibit the need of shortened linker names in mpif.h**

> See review-slide 69 for details.

Everywhere (because of 31 → 30 character name length):

`MPI_ASYNCHRONOUS_PROTECTS_NONBL` (old)

→ `MPI_ASYNC_PROTECTS_NONBLOCKING` (new)

Variants

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

The proposed text still contains some variants. Both alternatives are formally read, but before voting, the technically incorrect variant will be withdrawn.

Variants are on

- Page 13
  - Newly defined within these change-slides
  - About the size of an mpi_f08 handle counted in “Fortran numerical storage units”
- Page 581
  - About the extend rule in combination with user-defined Fortran
    - SEQUENCE derived types, and
    - BIND(C) derived types.

Important hints for other tickets of the MPI Forum

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 557, lines 1-13 contains the hint that (if a Fortran interface exists)

- **(Routine name length) + 2 * (number of arguments including ierror) ≤ 53**
- because 72 – 6 – 12 (for “SUBROUTINE P”) – 1 (for ending “)”) = 53.
- **Otherwise, the new routine is inconsistent with MPI-1.1 and later.**
- Therefore, this rule must be controlled by all ticket authors and reviewers!!!!!

- Additionally, the routine name length must be **≤ 30.**

Next steps

- As part of the formal reading and re-reading, these slides are provided on #229.
- The variants must be decided → technical analysis.
- The final pdf must be provided.
- Chicago meeting, Oct. 2011
  - On the first day, again a tutorial to inform all non-Fortran members about the goals and how they are achieved with ticket #229
  - One day for discussions on the floor to get all questions answered
  - 1st vote on the last day

Thank you for your interest and all the reviews!

The Reviewing Process (Thursday & Friday)

Major Changes since July 2011

- Callback problem with choice buffers is solved, see callback MPI_User_function and MPI_Datarep_conversion_function
- Character-Output-Strings with fixed size where appropriate
- MPI_Alloc_mem's Cray-Pointer interface is now deprecated
  → based on the reviews, we will not deprecate
- MPI_STATUS(ES)_IGNORE, MPI_ERRCODES_IGNORE and MPI_UNWEIGHTED are kept in mpi_f08

Reviews #229

(Green = review done, blue = traveled to Santorini, reviewers underlined)

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Special Fortran reviewers:
----- **Jeff Squyres** **(pre-reviews)**
----- **Craig E Rasmussen** **(pre-reviews)**
All **Hubert Ritzdorf**
16. **Purushotham Bangalore**
----- Bill Gropp
----- **Reinhold Bader** **(pre-reviews)**

Chapter authors from
http://meetings.mpi-forum.org/mpi3.0_chapter_wgs.php

--- Front matter: **Bill Gropp**
1. Introduction: **Bill Gropp**
2. Terms & Definitions: **Rolf Rabenseifner, Bill Gropp**
3. Point-to-Point: **Rich Graham** → **Brian Barrett**
4. Datatypes: George Bosilca
5. Collective Communication:
Adam Moody, **Torsten Hoefler**,
Yutaka Ishikawa, Hideyuki Jitsumoto
6. Process Topologies: **Torsten Hoefler**
7. Groups, Contexts, and Communicators: **Bill Gropp**
8. Environmental Management: **George Bosilca**
MPI_Alloc_mem → **Dave Godell** & **Brian Barrett**
9. Info-Object: **Darius Buntinas**
10. Process Creation and Management: David Solt
→ **Adam Moody**

11. One-Sided Communications: **Bill Gropp** + Rajeev Thakur
12. External Interf.: Bronis R. de Supinski → **Darius Buntinas**
13. I/O: Rajeev Thakur (c)
14. Profiling interface: Bronis R. de Supinski→ **Martin Schulz**
15. Deprecated Functions: **Rolf Rabenseifner** → **Jim Dinan**
MPI_Alloc_mem → **Dave Godell** & **Brian Barrett**
16. Language Bindings (→special reviews): Jeffrey M. Squyres
A. Annex Language Bindings Summary: **Rolf Rabenseifner**
→ **Hubert Ritzdorf** + **George Bosilca**
B. Annex Change-Log: **Rolf Rabenseifner**
→ **Hubert Ritzdorf**
B.4 New MPI-3.0 Interfaces:
B.4.1 #38, #274: Matched probe: (Torsten Hoefler) →
R.Graham → **Marc-Andre Hermanns**, **Rajeev Thakur**
B.4.2 #109: Nonblocking collectives: **Torsten Hoefler**
B.4.3 #258: Neighborhood Collective Comm.: **Torsten Hoefler**
B.4.4 #265: MPI_Count:(Fab Tillier)→**R.Graham**→**Adam Moody**
B.4.5 #266: Tools – there is no Fortran interface →**Martin Schulz**
B.4.6 #270: RMA: **Bill Gropp** + **Torsten Hoefler**
B.4.7 #276: Fault Tolerance: J. Hursey (interfaces are missing)
B.4.8 #284: Hybrid Programming: **Torsten Hoefler** (intrf. missing)
--- Bibliography: **Bill Gropp**

Review by Bill Gropp: Frontmatter & Chapter 1. Intro

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 1 is ok. The front matter is ok (as regards the Fortran material).

Review by Bill Gropp: Chapter 2. Terms & Conventions

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 2 is ok.

Review by Rolf Rabenseifner: Chapter 2. Terms & Conventions

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

p13:27-31

> "The use of the INTEGER defined handles and the BIND(C) derived type handles is different: Fortran 2003 (and later) define that BIND(C) derived types can be used within user defined common blocks, but it is up to the rules of the companion C compiler how many numerical storage unit are used for these BIND(C) derived type handles."

This sentence seems to be wrong:

> If it would be true, then we would have a serious problem with the definition of epsilon on p86:45-47 (= MPI-2.2 p78:45-47) and the rationale on the next page line 7-9.
>
> We currently try to check, whether a C struct containing one integer of the size of a Fortran INTEGER must have the size and alignment (when used in an array of struct) that is identical to the size and alignment of this integer.

I propose the alternative sentence:

> "The use of the INTEGER defined handles and the BIND(C) derived type handles is similar: both handles require one numerical storage unit."

(Both alternatives are formally read; decision will be done before voting.)

Review by Brian Barrett: Chapter 3. Point-to-point

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

- request should be before flag in arg list of MPI_Test → **OKAY**
- Shouldn't it be array_of_statuses(count) for MPI_Waitall & MPI_Testall?
  → NO, due to MPI_STATUSES_IGNORE
- Shouldn't array_of_statuses have size incount for MPI_Testsome and MPI_Waitsome?
  → NO, due to MPI_STATUSES_IGNORE
- Shouldn't array_of_indices have size incount for MPI_Testsome and MPI_Waitsome?
  → We come back to this in Hubert's review

All the rest of Chapter 3 is fine as far as I oversee.

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by George Bosilca: Chapter 4. Datatypes

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

All okay in Pre-review.

He is still re-checking

- All routines with CHARACTER(LEN=MPI_...MAX…)
  - Non in this chapter
- Is Section 16.2.15 okay?

All fine now.

Review by Torsten Hoefler: Chapter 5. Collectives

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

The collectives and process topology chapters are reviewed and ok.

Review by Adam Moody: Chapter 5. Collectives

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Why not list sendtype and recvtype on same line?
MPI_Gather, MPI_Gatherv, MPI_Scatter, MPI_Scatterv,
MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Alltoallv → **OKAY**

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by Adam Moody: Chapter 5. Collectives

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

At least the count, displacement, and datatype arrays on icollectives need to be marked as ASYNCHRONOUS due to page 201, line 2-5:

> "Once initiated, all associated send buffers and buffers associated with input arguments (such as arrays of counts, displacements, or datatypes in the vector versions of the collectives) should not be modifed, and all associated receive buffers should not be accessed, until the collective operation completes."

MPI_Igatherv:	recvcounts, displs
MPI_Iscatterv:	sendcounts, displs
MPI_Iallgatherv:	recvcounts, displs
MPI_Ialltoallv:	sendcounts, sdispls, recvcounts, rdispls
MPI_Ialltoallw:	sendcounts, sdispls, recvcounts, rdispls, sendtypes, recvtypes
MPI_Ireduce_scatter:	recvcounts

→ OKAY

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by Torsten Hoefler: Chapter 6. Topologies

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

The collectives and process topology chapters are reviewed and ok.

Review by Bill Gropp: Chapter 7. Groups, context, …

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 238, lines 40-41 put the second argument ahead of the first in MPI_Group_incl. Like MPI_Win_test, not incorrect but odd, and different than MPI_Group_excl on the next page. → **OKAY**

(Same on p672 lines 2-3 in Annex A.3)

Page 270, lines 47. This is probably ok, but with support for what is equivalent to a void *, shouldn't the extra state parameter use that instead of the pre-Fortran2008 "use an address-sized integer as the value of a pointer or something similar" hack? At least a rationale for sticking with the older Fortran style? → **NO CHANGES**

Comment: There isn‘t any information about extra state, i.e., it is void * in C, and INTEGER(KIND=MPI_ADDRESS_KIND) in Fortran and that‘s it.

I'm not familiar enough with F 2008 to judge the attribute interface, but it looks ok.

Otherwise, this chapter is ok (and there are no required changes here).

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by George Bosilca: Chapter 8. Environment

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

All okay in Pre-review.

He is still re-checking

- All routines with CHARACTER(LEN=MPI_...MAX…)
- MPI_Alloc_mem → slides about Alloc_mem

All other is okay.

Review by Brian Barrett: MPI_Alloc_mem

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

- Why "::" in MPI_ALLOC_MEM version? Not anywhere else in standard for old style declarations. → **NO CHANGES**
  Comment: This routine and MPI_Win_allocate are the only routines where Fortran 2003 elements are used within the specification. If tese elements are not available, the fallback solution (F90) is given at p327:33-35. Here, "::" is not used.
- Overloading is really not clear in definition.
  → next slide
- No deprecation of Cray pointers... → **OKAY**
  - Section 8.2, page 327, lines 36-37 should be removed
  - Section 15.3, page 535, lines 2-6 should be removed
- Why is base async in Free_mem?
  Add after p586:36 (see also B.4.1.6: #270 RMA review):

  In some MPI routines, a buffer dummy argument is defined as ASYNCHRONOUS to guarantee passing by reference, provided that the actual argument is also defined as ASYNCHRONOUS.

Review by Brian Barrett: MPI_Alloc_mem

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 327 lines 16-38 read:

```
MPI_ALLOC_MEM(SIZE, INFO, BASEPTR, IERROR)
  USE, INTRINSIC :: ISO_C_BINDING
  INTEGER :: INFO, IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
  TYPE(C_PTR) :: BASEPTR !overloaded with following...
  INTEGER(KIND=MPI_ADDRESS_KIND) BASEPTR ! ...type
```

Will be interchanged

{void* MPI::Alloc_mem(MPI::Aint size, const MPI::Info& info)(binding deprecated, see Section 15.2) }

With the Fortran mpi modules, MPI_ALLOC_MEM is an INTERFACE with two routines through function overloading: One routine defines baseptr as an INTEGER(KIND=MPI_ADDRESS_KIND), and the second one as TYPE(C_PTR). The first one is without a linker suffix, the second one has _CPTR as linker suffix, see Section 16.2.5 on page 557.

With Fortran mpif.h or if the compiler does not provide the TYPE(C_PTR) interface, only the INTEGER(KIND=MPI_ADDRESS_KIND) BASEPTR is required:

Must be reworded

```
MPI_ALLOC_MEM(SIZE, INFO, BASEPTR, IERROR)
  INTEGER INFO, IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

The Fortran interfaces with INTEGER(KIND=MPI_ADDRESS_KIND) BASEPTR in the mpi module and the mpif.h include le are deprecated since MPI-3.0.

Will be removed

Review by Brian Barrett: MPI_Alloc_mem

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

but should read:

Same will be done for Ticket RMA: MPI_Win_allocate

```
MPI_ALLOC_MEM(SIZE, INFO, BASEPTR, IERROR)
  INTEGER INFO, IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

{void* MPI::Alloc_mem(MPI::Aint size, const MPI::Info& info)(binding deprecated, see Section 15.2) }

If the Fortran compiler provides TYPE(C_PTR), then the following interface must be provided in the mpi module and should be provided in mpif.h through overloading, i.e., with the same routine name as the routine with INTEGER(KIND=MPI_ADDRESS_KIND) BASEPTR, but with a different linker name:

```
INTERFACE MPI_ALLOC_MEM
  SUBROUTINE MPI_ALLOC_MEM_CPTR(SIZE, INFO, BASEPTR, IERROR)
    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
    INTEGER :: INFO, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
    TYPE(C_PTR) :: BASEPTR
  END SUBROUTINE
END INTERFACE
```

The linker name base of this overloaded function is MPI_ALLOC_MEM_CPTR. The implied linker names are described in Section 16.2.5 on page 557.

Review by Darius Buntinas: Chapter 9. Info-Object

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

I've reviewed the Info-object chapter and it looks fine to me.

Review by Adam Moody: Chapter 10. Process Creation & Mngmt

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 10 is fine as far as I oversee.

Review by Bill Gropp: Chapter 11. One-sided

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 415, lines 8-9, it is odd that "LOGICAL, INTENT(OUT) :: flag
comes before the declaration of the win argument in MPI_Win_test.
It isn't incorrect, but it doesn't follow the usual style. → **OKAY**
(Same on p685 lines 14-15 in Annex A.3) → **OKAY**

Otherwise, this section is ok.

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by Darius Buntinas: Chapter 12. External Interfaces

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 434, lines 26-28

```
MPI_Grequest_start(query_fn, free_fn, cancel_fn, extra_state, request, ierror) BIND(C)
  PROCEDURE(MPI_Grequest_query_function) :: query_fn
  PROCEDURE(MPI_Grequest_free_function) :: free_fn
  PROCEDURE(MPI_Grequest_cancel_function) :: cancel_fn
```

Page 435, lines 14-16

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_query_function(extra_state, status, ierror) BIND(C)
    TYPE(MPI_Status) :: status
    INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state
    INTEGER :: ierror
```

etc.

Do you need INTENT() for these arguments? → **NO CHANGES**

Comment: (a) Callbacks need no INTENT. (b) Arguments of the callbacks should not have an INTENT to keep the needed changes small when moving from the mpi to the mpi_f08 module.

Review by Rajeev Thakur: Chapter 13. I/O

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

p471:6 & p692:15: In MPI_File_write_at_all, buf needs an INTENT(IN). → **OKAY**

p483:13-14 & p688:24-25: In MPI_File_iwrite_shared, the definition of the 2nd argument is before the 1st. → **OKAY**

p501:37-38 & p630:30-31: In MPI_Datarep_extent_function, the definition of ierror is before extent and extra_state → **OKAY**

P502:15 & p630:39: In MPI_Datarep_conversion_function, the definition of ierror is before position and extra_state → **NO CHANGES**

Comment: count and ierror are defined together on one statement.

Rest of the I/O chapter looks ok.

BTW, in MPI_Probe and MPI_Iprobe why is status declared as INTENT(OUT)?
→ **NO CHANGES**
Comment: MPI_STATUS_IGNORE is not allowed, see Sect. 3.2.6.
Only allowed in MPI_Recv, MPI_Test…, MPI_Wait…, and MPI_Request_get_status.

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by Martin Schulz: Chapter 14. Profiling Interface

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 521 lines 19-22 read

> provide a mechanism through which all of the MPI defined functions, except those allowed as macros (See Section 2.6.5), may be accessed with a name shift. This requires, in C and Fortran, an alternate entry point name, with the prefix PMPI_ for each MPI function.

but should read

> provide a mechanism through which all of the MPI defined functions, except those allowed as macros (See Section 2.6.5), may be accessed with a name shift. This requires, in C and Fortran, an alternate entry point name, with the prefix PMPI_ for each MPI function **in each provided language binding and language support method**.

Review by Martin Schulz: Chapter 14+16. Profiling Interface

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

The issue that we talked about that tools need to find all available bindings and hence would need some kind of query function (to find which of the 7 options are available).

The 7 options are

- C binding
- Fortran with mpif.h or mpi module and **without** TR 29113
- Fortran with mpi_f08 module and **without** TR 29113
- Fortran with mpif.h or mpi module and **with** TR 29113
- Fortran with mpi_f08 module and **with** TR 29113
- Some routines where we need to shorten the routine names for mpif.h with TR 29113
- C++ binding

This problem is not only a Fortran problem.

→ NO CHANGES

Comment: A proposal should be done outside of this Fortran ticket #229.

Review by Martin Schulz: Chapter 14+16. Profiling Interface

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

The proposal contains three Fortran support methods. There should be some language in there (unless I missed it) that ensure that the options are compatible (e.g., have the same constants). → **OKAY**

On page 551, line 15 the bold text should be added:

> The INTEGER compile-time constant MPI_SUBARRAYS_SUPPORTED is set to … The INTEGER compile-time constant MPI_ASYNCHRONOUS_PROTECTS_NONBL is set to … These constants exist with each Fortran support method, but not in the C/C++ header files. The values may be different for each Fortran support method. **All other constants and the integer values of handles must be the same for each Fortran support method.**

Review by Martin Schulz: Chapter 14+16. Profiling Interface

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 551, line 8:
The "all" is strange here - sounds like that you can also do a subset and then the flag would be .FALSE. - similar for the second flag. I would assume that "half" implementations are not allowed, correct?

→ NO CHANGES
Comment: It is allowed to implement only a few routines with TR 29113.
From the users viewpoint, it is like none of the routines has TR 29113 quality.

Page 553, line 14:
Quotes around TR name **→ OKAY**

Review by Martin Schulz: Chapter 14+16. Profiling Interface

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 557, line 10+13:
Use PMPI_DIST_... and 66 characters. **→ OKAY**

Page 557, line 26:
Yes, the shortened names must be standardized. When we do this, we also should retain the pre and post fixes and only change the name in the middle. Ideal would be something automatic that can be computed, but I am not sure if this is feasible.

→ See TODO slides

Page 557/558:
This text is not compatible anymore with the need to shorten names in some instances.

→ Yes, we then need wording on which base-names apply for which support method.

Page 558, line 31:
We now have “four or five (in the case of shortened names for mpif.h)” instead of “four” interfaces. ~~**→ OKAY**~~ **→ NO, because we decided at the meeting to forbid this shortening**

Review by Martin Schulz: Chapter 14+16. Profiling Interface

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 559, line 26-34:
That part is not clear to me - does this mean there are _f and _f08 routines for all overloaded functions? Also, it seems hard to deduce the names (in the example the type is C_PTR, but the suffix is CPTR). Why do we need overloading? If we do need it, it would be preferable to actually list all cases and names.

→ NO CHANGES

Comment: Overloading is only with MPI_Alloc_mem and MPI_Win_allocate.

The additional suffix is defined with each routine, see, p327:27-29 and p765:44-46.

Review by Puri Bangalore: Chapter 16. Language Bindings

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 551: Section 16.2.2, Line 46 must be indented one more level **→ OKAY**

Page 553: Line 20: According to .[35] page iv, last paragraph, "it is
=> There is an extra period before [35] and also there is no ending quotes at the end of the paragraph. **→ OKAY**

Page 554 Line 37: simply contiguous is in italics where as on Page 563 Line 31 it is in double quotes and on Page 578 it is in single quotes. I am fine with keeping it all consistent (say, single quotes). **→ OKAY**

Page 557: Lines 14-16: I am not sure who will be checking all the new names in MPI-3.0 to make sure that they are less than 66 characters.
→ This is a TODO and should be discussed → see "TODO" slides

OKAY = Proposal will be integrated in the version for re-reading in Santorini 2011

Review by Christian Siebert: Chapter 16.3. Interoperability

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 600: About "Callback Functions"

The **bold** text should be added to line 18-21:

> Advice to implementors. Callback functions need to have a language tag. This tag is set when the callback function is passed in by the library function (which is presumably different for each language **and language support method**), and is used to generate the right calling sequence when the callback function is invoked. (End of advice to implementors.)

Review by Jim Dinan: Chapter 15. Deprecated Routines

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 527 line 34 and in all other Fortran interfaces:

MPI_TYPE_HVECTOR() must be removed.

→ OKAY

It will be substituted by

> For this routine, an interface within the {\tt mpi_f08} module was never defined.

Page 535, line 5: Suggest changing "overloaded with an interface" to "overloaded with an additional interface" to make it a little more clear that both interfaces are present.
→ OKAY

Comment: This paragraph will be totally deleted when the Forum decides, not to deprecate the Cray-Pointer interfaces → see review from Dave Goodell and Brian Barrett

Review by Dave Goodell and Brian Barrett: MPI_Alloc_mem

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Review: The Cray-pointer interfaces of MPI_Alloc_mem should not be deprecated.
→ OKAY

Section 15.3, page 535, lines 2-6 should be removed

Section 8.2, page 327, lines 36-37 should be removed

Review by George Bosilca: Annex A. Language Summary

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

(Not yet finished)

But we have the review from Hubert Ritzdorf. → **OKAY**

Review by Rolf Rabenseifner: Chapter B.4 at all

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 753, line 45-48, the bold text will be added:

> This section is not part of the MPI standard. It is only part of this draft document. The Fortran bindings with the mpi module or the mpif.h header file should be identical to those provided in the appropriate tickets. The mpi_f08 bindings are added here. They are provided as part of this draft.
>
> **The provided interfaces or changes will be part of the appropriate chapter when the cited ticket are voted into this chapter of the MPI standard.**

Reviews: Chapter B.4.x

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

B.4.1 #38, #274: Matched probe: (Torsten Hoefler) → **R.Graham** → **Marc-Andre Hermanns**

Torsten Hoefler: This is voted in and under the responsibility of the chapter author.

Rajeev Thakur: MPI_MProbe and MPI_Improbe also have status declared as INTENT(OUT), which I am not sure is right. → **NO CHANGES**

Comment: MPI_STATUS_IGNORE is not allowed, see Sect. 3.2.6.
Only allowed in MPI_Recv, MPI_Test…, MPI_Wait…, and MPI_Request_get_status.

Marc-Andre Hermanns: All okay

B.4.2 #109: Nonblocking collectives: **Torsten Hoefler**

Torsten: I checked those. Well done.

Adam Moody: See review on Coll. Chapter

B.4.3 #258: Neighborhood Collective Comm.: **Torsten Hoefler**

Torsten: I checked those too. Well done.

B.4.4 #265: MPI_Count: (Fab Tillier) → **R.Graham** → **Adam Moody**

Adam: All okay.

B.4.5 #266: Tools – there is no Fortran interface

Martin Schulz: Really no Fortran

Reviews: Chapter B.4.6: #270 – RMA

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Bill Gropp: B.4.6 looks ok, but I haven't been able to go over it line by line.

Torsten Hoefler:

MPI_Win_allocate: whatever we decide for MPI_Alloc_mem needs to be propagated to here (I have a weird feeling about the current overloading interface) → **OKAY**

MPI_WIN_ATTACH mpi binding: MPI_ADDRESS_SIZE → MPI_ADDRESS_KIND (fixed in latest proposal) → **OKAY**

MPI_Win_attach: I think base doesn't need to be ASYNCHRONOUS (there is no "wait" that can be moved around) → **no direct changes**

MPI_Win_detach: I think base doesn't need to be ASYNCHRONOUS → **no direct ch.**

Everything else looks fine!

Comment: ASYNCHRONOUS is used for two reasons:

- To prohibit some optimizations while asynchronous communication
- To guarantee call by reference (this is the reason here), therefore:

Add after p586:36:

In some MPI routines, a buffer dummy argument is defined as ASYNCHRONOUS to guarantee passing by reference, provided that the actual argument is also defined as ASYNCHRONOUS.

Review according to Brian Barrett: MPI_Win_allocate

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 765 line 7 – page 766 line 3 should read:

The following interface in ticket #270

```
MPI_Win_allocate(size, disp_unit, info, comm, baseptr, win, ierror) BIND(C)
   USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
   INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
   INTEGER, INTENT(IN) :: disp_unit
   TYPE(MPI_Info), INTENT(IN) :: info
   TYPE(MPI_Comm), INTENT(IN) :: comm
   TYPE(C_PTR), INTENT(OUT) :: baseptr
   TYPE(MPI_Win), INTENT(OUT) :: win
   INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_WIN_ALLOCATE(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
   INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

→ **continued on next slide**

Review according to Brian Barrett: MPI_Win_allocate

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

should be substituted by

```
MPI_Win_allocate(size, disp_unit, info, comm, baseptr, win, ierror) BIND(C)
  USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
  INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
  INTEGER, INTENT(IN) :: disp_unit
  TYPE(MPI_Info), INTENT(IN) :: info
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(C_PTR), INTENT(OUT) :: baseptr
  TYPE(MPI_Win), INTENT(OUT) :: win
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_WIN_ALLOCATE(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
  INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

→ **continued on next slide**

Review according to Brian Barrett: MPI_Win_allocate

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

If the Fortran compiler provides TYPE(C_PTR), then the following interface must be provided in the mpi module and should be provided in mpif.h through overloading, i.e., with the same routine name as the routine with INTEGER(KIND=MPI_ADDRESS_KIND) BASEPTR, but with a different linker name:

```
INTERFACE MPI_WIN_ALLOCATE
  SUBROUTINE MPI_WIN_ALLOCATE_CPTR(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
    INTEGER :: DISP_UNIT, INFO, COMM, WIN, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
    TYPE(C_PTR) :: BASEPTR
  END SUBROUTINE
END INTERFACE
```

The linker name base of this overloaded function is MPI_WIN_ALLOCATE_CPTR. The implied linker names are described in Section 16.2.5 on page 557.

Reviews: Chapter B.4.8: #284 – shared memory alloc

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

```
MPI_Comm_split_type(comm, comm_type, key, info, newcomm, ierror) BIND(C)
   TYPE(MPI_Comm), INTENT(IN) :: comm
   INTEGER, INTENT(IN) :: comm_type, key
   TYPE(MPI_Info), INTENT(IN) :: info
   TYPE(MPI_Comm), INTENT(OUT) :: newcomm
   INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_COMM_SPLIT_TYPE(COMM, COMM_TYPE, KEY, INFO, NEWCOMM, IERROR)
   INTEGER COMM, COMM_TYPE, KEY, INFO, NEWCOMM, IERROR
```

Reviews: Chapter B.4.x

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

B.4.7 #276: Fault Tolerance: J. Hursey (interfaces are missing)

B.4.8 #284: Hybrid Programming: **Torsten Hoefler** (on previous slide)

Which tickets are missing ???

- **Tickets that are in the state**
  - **formally read, or**
  - **in the voting process**
- **List:**
  #264 (DV collectives – scalable vector), #204 (MPI_Get_library_version),

Which tickets must do the new bindings within there tickets ???

- **Tickets that are not yet formally read at the Sep. 2011 Santorini meeting**
- **List: #288 (End point A), #289 (End point B), #Adam (info query), #Adam (set/get info on comm.), #286 (MPI_Comm_create_group), #Quincy (nonblocking IO)**

Review by Bill Gropp: Bibliography

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

The Bib is ok.

Reviews by Hubert Ritzdorf: Whole standard

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 3. Point-to-point, routine MPI_Waitsome, page 67, Line 2:

array_of_indices(*) -> array_of_indices(incount)

This is the (maximal) dimension required for MPI_Waitsome() (and this argument dimension would correspond to sources or destinations in MPI_Dist_graph_neighbors and other output argument dimensions in other routines)

routine MPI_Testsome, page 68, Line 21:

array_of_indices(*) -> array_of_indices(incount)

→ NO CHANGES
Comment: array_of_indices in MPI_WAITSOME and MPI_TESTSOME can be defined shorter than incount as long as the application knows that outcount cannot reach incount. Therefore I used only array_of_indices(*) and not array_of_indices(incount).

We can define array_of_indices(incount) but this would be a tiny change of the interface.

Should we do this change?

Straw	Yes	No	Abstain
Votes:	0	**4**	14

Reviews by Hubert Ritzdorf: Whole standard

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 3, Point-to-point, routines MPI_Iprobe and MPI_Probe, page 71, line 32, and page 72, Line 29:

Annex B.4, Ticket Page 754, Line 9 and 20 (definitions of MPI_MPROBE and MPI_IMPROBE):

TYPE(MPI_Status), INTENT(OUT) :: status → TYPE(MPI_Status) :: status

because status might be MPI_STATUS_IGNORE.

→ NO CHANGES
Comment: MPI_STATUS_IGNORE is not allowed, see Sect. 3.2.6.
Only allowed in MPI_Recv, MPI_Test…, MPI_Wait…, and MPI_Request_get_status.

Reviews by Hubert Ritzdorf: Whole standard

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 186:46 – 187:4 Definition of MPI_User_function

```
ABSTRACT INTERFACE
    SUBROUTINE MPI_User_function(invec, inoutvec, len, datatype) BIND(C)
       USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
       TYPE(C_PTR), VALUE :: invec, inoutvec
       INTEGER :: len
       TYPE(MPI_Datatype) :: datatype
```

I think that this definition violates the Rationale in Page 600, Lines 37-41.

> Reduce Operations
>
> Advice to users. Reduce operations receive as one of their arguments the datatype of the operands. Thus, one can define “polymorphic” reduce operations that work for C, C++, and Fortran datatypes. (End of advice to users.)

This rationale claims that the user functions are "polymorphic". The usage of the Fortran derived types C_PTR and MPI_Types disables this Rationale and would require that the MPI library takes care on the type of this user function (defined by Fortran 2008, Fortran, C).

→ NO CHANGES

Comment: TYPE(C_PTR) is a C pointer identical to void * and guarantees this polymorphism.

Disadvantage: The user has to change the source code when using mpi_f08, see p190:19-32.

Reviews by Hubert Ritzdorf: Whole standard

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 10. Proces creation, routine MPI_Comm_spawn_multiple, page 370, line 35 and Line 46 (mpi_f08 and mpi module):

array_of_argv(count, *) → array_of_argv(*,*)

array_of_argv is only significant on root. Also the dimension of array_of_maxprocs, array_of_commands and array_of_infos is not not defined with count.

→ (NO CHANGES required)

Comment: (*,*) is forbidden. Only the last dimension may be a *.

This is a problem of the Fortran interface since MPI-2.0, which implies that all non-root processes must provide a sensful combination of a count value and a scratch array for array_of_argv.

Proposal:

Add on page 371, line 34 (= MPI-2.2, p314:41)

In Fortran at non-root processes, the count argument must be set to a value that is consistent with the provided array_of_argv although the content of these arguments has no meaning for this operation.

Reviews by Hubert Ritzdorf: Whole standard

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Chapter 16.3 Language Interoperability, Page 600, Line 24-29, about callbacks:

> Advice to users. Callbacks themselves, including the predefined Fortran functions (e.g., MPI_COMM_NULL_COPY_FN) should not be passed from one application routine written in one language or Fortran support method to another application routine written in another language or Fortran support method, which passes this callback routine to an MPI routine (e.g., MPI_COMM_CREATE_KEYVAL); see also the advice to users on page 274. (End of advice to users.)

I don't know whether I understand this advice correctly. The new advice in Line 24 would be required also for Error Handlers since they use different interfaces for Fortran and Fortran 2008. But this doesn't make any sense for me, since the error handler is used for all languages. Therefore, I think that this advice should be removed since the MPI library must take care on definitions of callback functions.

→ Proposed alternative:

> Advice to users. If a subroutine written in one language or Fortran support method wants to pass a callback routine including the predefined Fortran functions (e.g., MPI_COMM_NULL_COPY_FN) to another application routine written in another language or Fortran support method, then it must be guaranteed that both routines use the callback interface definition that is defined for the argument when passing the callback to an MPI routine (e.g., MPI_COMM_CREATE_KEYVAL); see also the advice to users on page 274. (End of advice to users.)

Reviews by Hubert Ritzdorf: Whole standard

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Annex B.1 Page 745, Line 47: Add something such as

External variables MPI_F08_STATUS_IGNORE and MPI_F08_STATUSES_IGNORE were added to include file mpi.h

→ OKAY

Proposed text (bold is new):

> Section 3.2.5 on page 34, Section 16.2.3 on page 554, Section 16.2.2 on page 551, Section 16.2.7 on page 563, and Section 16.3.5 on page 596.
>
> Within the mpi_08 Fortran module, the status is defined as TYPE(MPI_Status). New conversion routines are added: MPI_STATUS_F2F08, MPI_STATUS_F082F, MPI_Status_c2f08, and MPI_Status_f082c**.**
> **In mpi.h, the new type MPI_F08_status, and the external variables MPI_F08_STATUS_IGNORE and MPI_F08_STATUSES_IGNORE are added.**

Review by Fab Tillier: A detail in Section 16.2.7 Requirements

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Page 563 lines 25-30:

The language features assumed-type and assumed-rank from Fortran 2008 TR 29113 [36] are available. This is required only for mpi_f08. As long as this requirement is not supported by the compiler, it is valid to build a**n** ~~preliminary~~ MPI~~-3.0 (and not later)~~ library that implements the mpi_f08 module with MPI_SUBARRAYS_SUPPORTED set to .FALSE..

Fab:

> Since Microsoft does not ship a Fortran compiler suite, control over standard compliance is effectively out of our hands.

Blue changes above and a comment, which is not part of the standard:

An MPI library that supports Fortran is always supporting Fortran that is provided by a specific compiler or a set of compilers. It is usually necessary to define such a set very precisely, e.g., including compiler version numbers. A compiler with a given version number cannot change its quality with respect to TR 29113, i.e., it stays forever TR 29113 compliant or not compliant. Therefore “as long as …” does not imply that the control over standard compliance is out of an MPI implementor’s hands.

Alternatives (References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

We read two alternatives and take the time until voting to choose, which alternative will be presented for voting.
(Formally is this like reading two different tickets and withdrawing one before voting.)

Page 581 lines 11-29:

TODO: The following text about the extent of derived types should be strongly checked by specialist on derived types!!! The correct variant should be chosen.

To use a derived type in an array requires a correct extent of the datatype handle to take care of the alignment rules applied by the compiler. These alignment rules may imply that there are gaps between the elements of a derived type, and also between the array elements. The alignment rules in Section 4.1 on page 85 and Section 4.1.6 on page 106 apply only to

VARIANT 1: SEQUENCE

VARIANT 2: BIND(C)

derived types. The extent of an iteroperable derived type (i.e., defined with BIND(C)) and a SEQUENCE derived type with the same content may be different because C and Fortran may apply different alignment rules.

VARIANT 1: Using the SEQUENCE attribute instead of BIND(C) in the declaration on mytype, one can directly use newtype to send the fooarr array.

VARIANT 2: In the example, one can directly use newtype to send the fooarr array. The resized newarrtype datatype is only needed, if mytype is a SEQUENCE derived type.

TODO (References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

You can still find “TODO” comments in this version. They will be deleted before re-reading.

Page 557 lines 1-18:

Advice to implementors. To make mpif.h compatible with both fixed- and free-source forms, to allow automatic inclusion by preprocessors, and to allow extended fixedform line length, it is recommended that the requirement of usability in free and fixed source form applications be met by constructing mpif.h without any continuation lines. This should be possible because mpif.h may contain only declarations, and because common block declarations can be split among several lines. The argument names may need to be shortened to keep the SUBROUTINE statement within the allowed 72-6=66 characters, e.g.,

```
INTERFACE
SUBROUTINE PMPI_DIST_GRAPH_CREATE_ADJACENT(a,b,c,d,e,f,g,h,i,j,k)
            ... ! dummy argument declarations
```

This line has ~~65~~66 characters and is the longest in MPI-3.0.

TODO: This is only checked for MPI-2.2. We have to check all new MPI-3.0 interfaces that they stay within these 66 characters. Otherwise the routine name should be shortened before the name is standardized.

This check was done with all routines, including tickets in Annex B.4.

This check must be done for all further tickets. This was already part of MPI-1 !!!!!
Up to now, it was mainly luck that nobody designed a routine name + argument list that was too long. ☺

TODO (References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

You can still find “TODO” comments in this version. They will be deleted before re-reading.

Page 557 lines 17-30:

If mpif.h contains also explicit interfaces with BIND(C,NAME='...') for providing MPI_SUBARRAYS_SUPPORTED and MPI_ASYNCHRONOUS_PROTECTS_NONBL equals .TRUE., the linker routine name may need to be shortened. For example, MPI_FILE_WRITE_AT_ALL_BEGIN with 6 arguments, may be defined:

```
INTERFACE MPI_FILE_WRITE_AT_ALL_BEGIN
SUBROUTINE MPI_X(a,b,c,d,e,f)BIND(C,NAME='MPI_File_write_at_all_begin_f')
    ... ! dummy argument declarations
```

This would need a line length of 73 characters, i.e., the C routine name must be shortened by 7 characters to stay within the available 66 characters. **TODO: Do we want to define these shortened routine names for mpif.h; this would help the tools people.** Note that the name MPI_X has no meaning for the compilation, and that this problem occurs only with routines with choice buffers implemented with the assumed-type and assumed-rank facility of TR 29113.

Question to the Forum: Do we expect that most implementors want to support only one internal Fortran interface and therefore try to provide for the mpi module and mpif.h also the TR quality when doing it for mpi_f08?

Question to the Forum:
Do we want to define these shortened routines? → **RESULT: Prohibit it by new wording**

TODO (References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

This slide contains the changes
to prohibit the need of shortened linker names in mpif.h

Page 557 lines 17-30 should read (blue parts are changed):

As long as the MPI standard contains routines with choice buffers and a name length and argument count that implies that a BIND(C) implementation would need to shorten their linker names in mpif.h, the mpif.h cannot set MPI_SUBARRAYS_SUPPORTED and MPI_ASYNCHRONOUS_PROTECTS_NONBL equals .TRUE., because such shortening is invalid. For example, MPI_FILE_WRITE_AT_ALL_BEGIN with 6 arguments, may be defined:

```
INTERFACE MPI_FILE_WRITE_AT_ALL_BEGIN
SUBROUTINE MPI_X(a,b,c,d,e,f)BIND(C,NAME='MPI_File_write_at_all_begin_f')
    ... ! dummy argument declarations
```

This would need a line length of 73 characters, i.e., the C routine name would need to be shortened by 7 characters to stay within the available 66 characters. Note that the name MPI_X has no meaning for the compilation, and that this problem occurs only with routines with choice buffers implemented with the assumed-type and assumed-rank facility of TR 29113.

Question by Torsten Hoefler: INTEGER(KIND=MPI_COUNT_KIND) count

(References → mpi-report-F2008-2011-09-08-changeonlyplustickets.pdf)

Question to the Forum:

Should we substitute in all locations in the new mpi_f08 module

```
INTEGER count
```

by

```
INTEGER(KIND=MPI_COUNT_KIND) count
```

Pros & cons:

- The user need not to use the workaround through arrays of derived types together with the routines provided by Ticket #265
- The implementor may need also a wrapper in C, i.e., different wrappers for C and the Fortran modules are calling the same backend that has long counts.

Straw	Yes	No	Abstain
Votes:	0	12	6

Next steps

- Receiving missing reviews on Thursday
- Formal reading of these reviews on Friday
- Straw votes
  - At July 2011 meeting
  - At Sep. 2011 meeting
- TODO: Defining the changes coming out of Friday meeting
- Re-reading the needed changes on Saturday
- First vote Oct. 2011, Chicago
- Second vote Jan. 2012, San Jose

Straw Votes:	Yes	No	Abstain
At July 2011 meeting	12	0	8
At Sep. 2011 meeting	6	2	10

5 of them have not been at the May 2011 meeting, i.e., they have not been familiar with the text presented their. Therefore, they have too little information for voting.

2 no: because they are not sure whether the ticket is really in a complete state