Annex B

Change-Log

This annex summarizes changes from the previous version of the MPI standard to the version presented by this document. [Only changes (i.e., clarifications and new features) are presented that may cause implementation effort in the MPI libraries.] Only significant changes (i.e., clarifications and new features) that might either require implementation effort in the MPI libraries or change the understanding of MPI from a user's perspective are presented. Editorial modifications, formatting, typo corrections and minor clarifications are not shown.

B.1 Changes from Version 2.1 to Version 2.2

1. Section 2.5.4 on page 14.
   It is now guaranteed that predefined named constant handles (as other constants) can be used in initialization expressions or assignments, i.e., also before the call to MPI_INIT.

2. Section 2.6 on page 16, Section 2.6.4 on page 19, and Section 16.1 on page 465.
   The C++ language bindings have been deprecated and will be removed in a future version of the MPI specification.

3. Section 3.2.2 on page 29.
   MPI_CHAR for printable characters is now defined for C type char (instead of signed char). This change should not have any impact on applications nor on MPI libraries (except some comment lines), because printable characters could and can be stored in any of the C types char, signed char, and unsigned char, and MPI_CHAR is not allowed for predefined reduction operations.

4. Section 3.2.2 on page 29.
   MPI_(U)INT{8,16,32,64}_T, MPI_AINT, MPI_OFFSET, MPI_C_BOOL, MPI_C_COMPLEX, MPI_C_FLOAT_COMPLEX, MPI_C_DOUBLE_COMPLEX, and MPI_C_LONG_DOUBLE_COMPLEX are now valid predefined MPI datatypes.

5. Section 3.4 on page 40, Section 3.7.2 on page 51, Section 3.9 on page 71, and Section 5.1 on page 131.
   The read access restriction on the send buffer for blocking, non blocking and collective API has been lifted. It is permitted to access for read the send buffer while the operation is in progress.

6. Section 3.7 on page 50.
   The Advice to users for IBSEND and IRSEND was slightly changed. ticket143.

7. Section 3.7.3 on page 54.
   The advice to free an active request was removed in the Advice to users for MPI_REQUEST_FREE. ticket137.

8. Section 3.7.6 on page 66.
   MPI_REQUEST_GET_STATUS changed to permit inactive or null requests as input. ticket31.

9. Section 5.8 on page 157.
   "In place" option is added to MPI_ALLTOALL, MPI_ALLTOALLV, and MPI_ALLTOALLW for intracommunicators. ticket64.

10. Section 5.9.2 on page 165.
    Predefined parameterized datatypes (e.g., returned by MPI_TYPE_CREATE_F90_REAL) and optional named predefined datatypes (e.g. MPI_REAL8) have been added to the list of valid datatypes in reduction operations. ticket18.

11. Section 5.9.2 on page 165.
    MPI_(U)INT{8,16,32,64}_T are all considered C integer types for the purposes of the predefined reduction operators. MPI_AINT and MPI_OFFSET are considered Fortran integer types. MPI_C_BOOL is considered a Logical type.
    MPI_C_COMPLEX, MPI_C_FLOAT_COMPLEX, MPI_C_DOUBLE_COMPLEX, and MPI_C_LONG_DOUBLE_COMPLEX are considered Complex types. ticket24.

12. Section ?? on page ??.
    The local routines MPI_REDUCE_LOCAL and MPI_OP_COMMUTATIVE have been added. ticket27.

13. Section ?? on page ??.
    The collective function MPI_REDUCE_SCATTER_BLOCK is added to the MPI standard. ticket94.

14. Section 5.11.2 on page 181.
    Added in place argument to MPI_EXSCAN. ticket19.

15. Section 6.4.2 on page 200, and Section 6.6 on page 219.
    Implementations that did not implement MPI_COMM_CREATE on intercommunicators will need to add that functionality. As the standard described the behavior of this operation on intercommunicators, it is believed that most implementations already provide this functionality. Note also that the C++ binding for both MPI_COMM_CREATE and MPI_COMM_SPLIT explicitly allow Intercomms. ticket66.

16. Section 6.4.2 on page 200.
    MPI_COMM_CREATE is extended to allow several disjoint subgroups as input if comm is an intracommunicator. If comm is an intercommunicator it was clarified that all processes in the same local group of comm must specify the same value for group. ticket33.

17. Section ?? on page ??.
    New functions for a scalable distributed graph topology interface has been added. In this section, the functions MPI_DIST_GRAPH_CREATE_ADJACENT and

MPI_DIST_GRAPH_CREATE, the constants MPI_UNWEIGHTED, and the derived C++ class Distgraphcomm were added. ticket33.

18. Section 7.5.4 on page 262.
For the scalable distributed graph topology interface, the functions MPI_DIST_NEIGHBORS_COUNT and MPI_DIST_NEIGHBORS and the constant MPI_DIST_GRAPH were added. ticket3.

19. Section 7.5.4 on page 262.
Remove ambiguity regarding duplicated neighbors with MPI_GRAPH_NEIGHBORS and MPI_GRAPH_NEIGHBORS_COUNT. ticket101.

20. Section 8.1.1 on page 273.
The subversion number changed from 1 to 2. ticket7.

21. Section 8.3 on page 278, Section ?? on page ??, and Annex A.1.3 on page 519.
Changed function pointer typedef names MPI_{Comm,File,Win}_errhandler_fn to MPI_{Comm,File,Win}_errhandler_function. Deprecated old "_fn" names. ticket71.

22. Section 8.7.1 on page 297.
Attribute deletion callbacks on MPI_COMM_SELF are now called in LIFO order. Implementors must now also register all implementation-internal attribute deletion callbacks on MPI_COMM_SELF before returning from MPI_INIT/MPI_INIT_THREAD. ticket43.

23. Section 11.3.4 on page 347.
The restriction added in MPI 2.1 that the operation MPI_REPLACE in MPI_ACCUMULATE can be used only with predefined datatypes has been removed. MPI_REPLACE can now be used even with derived datatypes, as it was in MPI 2.0. Also, a clarification has been made that MPI_REPLACE can be used only in MPI_ACCUMULATE, not in collective operations that do reductions, such as MPI_REDUCE and others. ticket6.

24. Section 12.2 on page 373.
Add "*" to the query_fn, free_fn, and cancel_fn arguments to the C++ binding for MPI::Grequest::Start() for consistency with the rest of MPI functions that take function pointer arguments. ticket18.

25. Section 13.5.2 on page 430, and Table 13.2 on page 432.
MPI_(U)INT{8,16,32,64}_T, MPI_AINT, MPI_OFFSET, MPI_C_COMPLEX, MPI_C_FLOAT_COMPLEX, MPI_C_DOUBLE_COMPLEX, MPI_C_LONG_DOUBLE_COMPLEX, and MPI_C_BOOL are added as predefined datatypes in the external32 representation. ticket55.

26. Section 16.3.7 on page 502.
The description was modified that it only describes how an MPI implementation behaves, but not how MPI stores attributes internally. The erroneous MPI-2.1 Example 16.17 was replaced with three new examples ??, ??, and ?? on pages ??-?? explicitly detailing cross-language attribute behavior. Implementations that matched the behavior of the old example will need to be updated. ticket4.

27. Annex A.1.1 on page 507.
Removed type MPI::Fint (compare MPI_Fint in Section A.1.2 on page 518). ticket18.

28. Annex A.1.1 on page 507. Table *Named Predefined Datatypes*.
Added MPI_(U)INT{8,16,32,64}_T, MPI_AINT, MPI_OFFSET, MPI_C_BOOL, MPI_C_FLOAT_COMPLEX, MPI_C_COMPLEX, MPI_C_DOUBLE_COMPLEX, and MPI_C_LONG_DOUBLE_COMPLEX are added as predefined datatypes.

B.2 Changes from Version 2.0 to Version 2.1

1. Section 3.2.2 on page 29, Section 16.1.6 on page 469, and Annex A.1 on page 507.
In addition, the MPI_LONG_LONG should be added as an optional type; it is a synonym for MPI_LONG_LONG_INT.

2. Section 3.2.2 on page 29, Section 16.1.6 on page 469, and Annex A.1 on page 507.
MPI_LONG_LONG_INT, MPI_LONG_LONG (as synonym), MPI_UNSIGNED_LONG_LONG, MPI_SIGNED_CHAR, and MPI_WCHAR are moved from optional to official and they are therefore defined for all three language bindings.

3. Section 3.2.5 on page 33.
MPI_GET_COUNT with zero-length datatypes: The value returned as the count argument of MPI_GET_COUNT for a datatype of length zero where zero bytes have been transferred is zero. If the number of bytes transferred is greater than zero, MPI_UNDEFINED is returned.

4. Section 4.1 on page 79.
General rule about derived datatypes: Most datatype constructors have replication count or block length arguments. Allowed values are [nonnegative]non-negative integers. If the value is zero, no elements are generated in the type map and there is no effect on datatype bounds or extent. ticket74.

5. Section 4.3 on page 129.
MPI_BYTE should be used to send and receive data that is packed using MPI_PACK_EXTERNAL.

6. Section 5.9.6 on page 175.
If comm is an intercommunicator in MPI_ALLREDUCE, then both groups should provide count and datatype arguments that specify the same type signature (i.e., it is not necessary that both groups provide the same count value).

7. Section 6.3.1 on page 192.
MPI_GROUP_TRANSLATE_RANKS and MPI_PROC_NULL: MPI_PROC_NULL is a valid rank for input to MPI_GROUP_TRANSLATE_RANKS, which returns MPI_PROC_NULL as the translated rank.

8. Section 6.7 on page 234.
About the attribute caching functions:

> *Advice to implementors.* High-quality implementations should raise an error when a keyval that was created by a call to MPI_XXX_CREATE_KEYVAL is used with an object of the wrong type with a call to MPI_YYY_GET_ATTR, MPI_YYY_SET_ATTR, MPI_YYY_DELETE_ATTR, or