Chapter 8

MPI Environmental Management

This chapter discusses routines for getting and, where appropriate, setting various parameters that relate to the MPI implementation and the execution environment (such as error handling). The procedures for entering and leaving the MPI execution environment are also described here.

8.1 Implementation Information

8.1.1 Version Inquiries

In order to cope with changes to the MPI Standard, there are both compile-time and run-time ways to determine which version of the standard is in use in the environment one is using.

The "version" will be represented by two separate integers, for the version and subversion: In C and C++,

```
#define MPI_VERSION    2
#define MPI_SUBVERSION [ticket101.][1]2
```

in Fortran,

```
INTEGER MPI_VERSION, MPI_SUBVERSION
PARAMETER (MPI_VERSION    = 2)
PARAMETER (MPI_SUBVERSION = [ticket101.][1]2)
```

For runtime determination,

MPI_GET_VERSION(version, subversion)

OUT	version	version number (integer)
OUT	subversion	subversion number (integer)

```
int MPI_Get_version(int *version, int *subversion)
```

```
MPI_GET_VERSION(VERSION, SUBVERSION, IERROR)
    INTEGER VERSION, SUBVERSION, IERROR
```

ticket150.

{void MPI::Get_version(int& version, int& subversion) *(binding deprecated, see* ticket150.

Section 15.2) }

MPI_GET_VERSION is one of the few functions that can be called before MPI_INIT and after MPI_FINALIZE. Valid (MPI_VERSION, MPI_SUBVERSION) pairs in this and previous versions of the MPI standard are (2,2), (2,1), (2,0), and (1,2). ticket101.

8.1.2 Environmental Inquiries

A set of attributes that describe the execution environment are attached to the communicator MPI_COMM_WORLD when MPI is initialized. The value of these attributes can be inquired by using the function [MPI_ATTR_GET]MPI_COMM_GET_ATTR described in Chapter 6. It is erroneous to delete these attributes, free their keys, or change their values. ticket149.

The list of predefined attribute keys include

MPI_TAG_UB Upper bound for tag value.

MPI_HOST Host process rank, if such exists, MPI_PROC_NULL, otherwise.

MPI_IO rank of a node that has regular I/O facilities (possibly myrank). Nodes in the same communicator may return different values for this parameter.

MPI_WTIME_IS_GLOBAL Boolean variable that indicates whether clocks are synchronized.

Vendors may add implementation specific parameters (such as node number, real memory size, virtual memory size, etc.)

These predefined attributes do not change value between MPI initialization (MPI_INIT and MPI completion (MPI_FINALIZE), and cannot be updated or deleted by users.

Advice to users. Note that in the C binding, the value returned by these attributes is a *pointer* to an `int` containing the requested value. (*End of advice to users.*)

The required parameter values are discussed in more detail below:

Tag Values

Tag values range from `0` to the value returned for MPI_TAG_UB inclusive. These values are guaranteed to be unchanging during the execution of an MPI program. In addition, the tag upper bound value must be *at least* 32767. An MPI implementation is free to make the value of MPI_TAG_UB larger than this; for example, the value $2^{30} - 1$ is also a legal value for MPI_TAG_UB.

The attribute MPI_TAG_UB has the same value on all processes of MPI_COMM_WORLD.

Host Rank

The value returned for MPI_HOST gets the rank of the `HOST` process in the group associated with communicator MPI_COMM_WORLD, if there is such. MPI_PROC_NULL is returned if there is no host. MPI does not specify what it means for a process to be a `HOST`, nor does it requires that a `HOST` exists.

The attribute MPI_HOST has the same value on all processes of MPI_COMM_WORLD.

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

18. Section 7.5.5 on page 273.
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 7.5.5 on page 273.
Remove ambiguity regarding duplicated neighbors with MPI_GRAPH_NEIGHBORS and MPI_GRAPH_NEIGHBORS_COUNT. ticket101.

20. Section 8.1.1 on page 287.
The subversion number changed from 1 to 2. ticket7.

21. Section 8.3 on page 292, Section 15.2 on page 484, and Annex A.1.3 on page 539.
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 311.
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 361.
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 391.
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 449, and Table 13.2 on page 451.
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 522.
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 527.
Removed type MPI::Fint (compare MPI_Fint in Section A.1.2 on page 538). ticket18.