

Annex A

Language Bindings Summary

In this section we summarize the specific bindings for C and Fortran. First we present the constants, type definitions, info values and keys. Then we present the routine prototypes separately for each binding. Listings are alphabetical within chapter.

A.1 Defined Values and Handles

A.1.1 Defined Constants

The C and Fortran names are listed below. Constants with the type `const int` may also be implemented as literal integer constants substituted by the preprocessor.

Error classes
C type: <code>const int</code> (or unnamed <code>enum</code>)
Fortran type: <code>INTEGER</code>
MPI_SUCCESS
MPI_ERR_BUFFER
MPI_ERR_COUNT
MPI_ERR_TYPE
MPI_ERR_TAG
MPI_ERR_COMM
MPI_ERR_RANK
MPI_ERR_REQUEST
MPI_ERR_ROOT
MPI_ERR_GROUP
MPI_ERR_OP
MPI_ERR_TOPOLOGY
MPI_ERR_DIMS
MPI_ERR_ARG
MPI_ERR_UNKNOWN
MPI_ERR_TRUNCATE
MPI_ERR_OTHER
MPI_ERR_INTERN
MPI_ERR_PENDING

(Continued on next page)

Error classes (continued)

C type: <code>const int</code> (or unnamed <code>enum</code>)
Fortran type: <code>INTEGER</code>
<hr/>
<code>MPI_ERR_IN_STATUS</code>
<code>MPI_ERR_ACCESS</code>
<code>MPI_ERR_AMODE</code>
<code>MPI_ERR_ASSERT</code>
<code>MPI_ERR_BAD_FILE</code>
<code>MPI_ERR_BASE</code>
<code>MPI_ERR_CONVERSION</code>
<code>MPI_ERR_DISP</code>
<code>MPI_ERR_DUP_DATAREP</code>
<code>MPI_ERR_FILE_EXISTS</code>
<code>MPI_ERR_FILE_IN_USE</code>
<code>MPI_ERR_FILE</code>
<code>MPI_ERR_INFO_KEY</code>
<code>MPI_ERR_INFO_NOKEY</code>
<code>MPI_ERR_INFO_VALUE</code>
<code>MPI_ERR_INFO</code>
<code>MPI_ERR_IO</code>
<code>MPI_ERR_KEYVAL</code>
<code>MPI_ERR_LOCKTYPE</code>
<code>MPI_ERR_NAME</code>
<code>MPI_ERR_NO_MEM</code>
<code>MPI_ERR_NOT_SAME</code>
<code>MPI_ERR_NO_SPACE</code>
<code>MPI_ERR_NO_SUCH_FILE</code>
<code>MPI_ERR_PORT</code>
<code>MPI_ERR_QUOTA</code>
<code>MPI_ERR_READ_ONLY</code>
<code>MPI_ERR_RMA_ATTACH</code>
<code>MPI_ERR_RMA_CONFLICT</code>
<code>MPI_ERR_RMA_RANGE</code>
<code>MPI_ERR_RMA_SHARED</code>
<code>MPI_ERR_RMA_SYNC</code>
<code>MPI_ERR_RMA_FLAVOR</code>
<code>MPI_ERR_SERVICE</code>
<code>MPI_ERR_SIZE</code>
<code>MPI_ERR_SPAWN</code>
<code>MPI_ERR_UNSUPPORTED_DATAREP</code>
<code>MPI_ERR_UNSUPPORTED_OPERATION</code>
<code>MPI_ERR_WIN</code>

(Continued on next page)

Error classes (continued)

C type: <code>const int</code> (or unnamed <code>enum</code>)
Fortran type: <code>INTEGER</code>

<code>MPI_T_ERR_CANNOT_INIT</code>
<code>MPI_T_ERR_NOT_INITIALIZED</code>
<code>MPI_T_ERR_MEMORY</code>
<code>MPI_T_ERR_INVALID</code>
<code>MPI_T_ERR_INVALID_INDEX</code>
<code>MPI_T_ERR_INVALID_ITEM</code>
<code>MPI_T_ERR_INVALID_SESSION</code>
<code>MPI_T_ERR_INVALID_HANDLE</code>
<code>MPI_T_ERR_INVALID_NAME</code>
<code>MPI_T_ERR_OUT_OF_HANDLES</code>
<code>MPI_T_ERR_OUT_OF_SESSIONS</code>
<code>MPI_T_ERR_CVAR_SET_NOT_NOW</code>
<code>MPI_T_ERR_CVAR_SET_NEVER</code>
<code>MPI_T_ERR_PVAR_NO_WRITE</code>
<code>MPI_T_ERR_PVAR_NO_STARTSTOP</code>
<code>MPI_T_ERR_PVAR_NO_ATOMIC</code>
<code>MPI_ERR_LASTCODE</code>

Buffer Address Constants

C type: <code>void * const</code>
Fortran type: (predefined memory location) ¹

<code>MPI_BOTTOM</code>
<code>MPI_IN_PLACE</code>

¹ Note that in Fortran these constants are not usable for initialization expressions or assignment. See Section 2.5.4.

Assorted Constants

C type: <code>const int</code> (or unnamed <code>enum</code>)
Fortran type: <code>INTEGER</code>

<code>MPI_PROC_NULL</code>
<code>MPI_ANY_SOURCE</code>
<code>MPI_ANY_TAG</code>
<code>MPI_UNDEFINED</code>
<code>MPI_BSEND_OVERHEAD</code>
<code>MPI_KEYVAL_INVALID</code>
<code>MPI_LOCK_EXCLUSIVE</code>
<code>MPI_LOCK_SHARED</code>
<code>MPI_ROOT</code>

No Process Message Handle

C type: <code>MPI_Message</code>
Fortran type: <code>INTEGER</code> or <code>TYPE(MPI_Message)</code>

<code>MPI_MESSAGE_NO_PROC</code>

Fortran Support Method Specific Constants

Fortran type: LOGICAL

MPI_SUBARRAYS_SUPPORTED (Fortran only)

MPI_ASYNC_PROTECTS_NONBLOCKING (Fortran only)

Status size and reserved index values (Fortran only)

Fortran type: INTEGER

MPI_STATUS_SIZE

MPI_SOURCE

MPI_TAG

MPI_ERROR

Variable Address Size (Fortran only)

Fortran type: INTEGER

MPI_ADDRESS_KIND

MPI_COUNT_KIND

MPI_INTEGER_KIND

MPI_OFFSET_KIND

Error-handling specifiers

C type: MPI_Errhandler

Fortran type: INTEGER or TYPE(MPI_Errhandler)

MPI_ERRORS_ARE_FATAL

MPI_ERRORS_RETURN

Maximum Sizes for Strings

C type: const int (or unnamed enum)

Fortran type: INTEGER

MPI_MAX_DATAREP_STRING

MPI_MAX_ERROR_STRING

MPI_MAX_INFO_KEY

MPI_MAX_INFO_VAL

MPI_MAX_LIBRARY_VERSION_STRING

MPI_MAX_OBJECT_NAME

MPI_MAX_PORT_NAME

MPI_MAX_PROCESSOR_NAME

Named Predefined Datatypes	C types
C type: MPI_Datatype	
Fortran type: INTEGER	
or TYPE(MPI_Datatype)	
MPI_CHAR	char (treated as printable character)
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long
MPI_LONG_LONG_INT	signed long long
MPI_LONG_LONG (as a synonym)	signed long long
MPI_SIGNED_CHAR	signed char (treated as integral value)
MPI_UNSIGNED_CHAR	unsigned char (treated as integral value)
MPI_UNSIGNED_SHORT	unsigned short
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long
MPI_UNSIGNED_LONG_LONG	unsigned long long
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_WCHAR	wchar_t (defined in <stddef.h>) (treated as printable character)
MPI_C_BOOL	_Bool
MPI_INT8_T	int8_t
MPI_INT16_T	int16_t
MPI_INT32_T	int32_t
MPI_INT64_T	int64_t
MPI_UINT8_T	uint8_t
MPI_UINT16_T	uint16_t
MPI_UINT32_T	uint32_t
MPI_UINT64_T	uint64_t
MPI_AINT	MPI_Aint
MPI_COUNT	MPI_Count
MPI_OFFSET	MPI_Offset
MPI_C_COMPLEX	float _Complex
MPI_C_FLOAT_COMPLEX	float _Complex
MPI_C_DOUBLE_COMPLEX	double _Complex
MPI_C_LONG_DOUBLE_COMPLEX	long double _Complex
MPI_BYTE	(any C type)
MPI_PACKED	(any C type)

Named Predefined Datatypes	Fortran types
C type: MPI_Datatype Fortran type: INTEGER or TYPE(MPI_Datatype)	
MPI_INTEGER	INTEGER
MPI_REAL	REAL
MPI_DOUBLE_PRECISION	DOUBLE PRECISION
MPI_COMPLEX	COMPLEX
MPI_LOGICAL	LOGICAL
MPI_CHARACTER	CHARACTER(1)
MPI_AINT	INTEGER (KIND=MPI_ADDRESS_KIND)
MPI_COUNT	INTEGER (KIND=MPI_COUNT_KIND)
MPI_OFFSET	INTEGER (KIND=MPI_OFFSET_KIND)
MPI_BYTE	(any Fortran type)
MPI_PACKED	(any Fortran type)

Named Predefined Datatypes ¹	C++ types
C type: MPI_Datatype Fortran type: INTEGER or TYPE(MPI_Datatype)	
MPI_CXX_BOOL	bool
MPI_CXX_FLOAT_COMPLEX	std::complex<float>
MPI_CXX_DOUBLE_COMPLEX	std::complex<double>
MPI_CXX_LONG_DOUBLE_COMPLEX	std::complex<long double>

¹ If an accompanying C++ compiler is missing, then the MPI datatypes in this table are not defined.

Optional datatypes (Fortran)	Fortran types
C type: MPI_Datatype Fortran type: INTEGER or TYPE(MPI_Datatype)	
MPI_DOUBLE_COMPLEX	DOUBLE COMPLEX
MPI_INTEGER1	INTEGER*1
MPI_INTEGER2	INTEGER*2
MPI_INTEGER4	INTEGER*4
MPI_INTEGER8	INTEGER*8
MPI_INTEGER16	INTEGER*16
MPI_REAL2	REAL*2
MPI_REAL4	REAL*4
MPI_REAL8	REAL*8
MPI_REAL16	REAL*16
MPI_COMPLEX4	COMPLEX*4
MPI_COMPLEX8	COMPLEX*8
MPI_COMPLEX16	COMPLEX*16
MPI_COMPLEX32	COMPLEX*32

Datatypes for reduction functions (C)	1
C type: MPI_Datatype	2
Fortran type: INTEGER or TYPE(MPI_Datatype)	3
MPI_FLOAT_INT	4
MPI_DOUBLE_INT	5
MPI_LONG_INT	6
MPI_2INT	7
MPI_SHORT_INT	8
MPI_LONG_DOUBLE_INT	9
	10
Datatypes for reduction functions (Fortran)	11
C type: MPI_Datatype	12
Fortran type: INTEGER or TYPE(MPI_Datatype)	13
MPI_2REAL	14
MPI_2DOUBLE_PRECISION	15
MPI_2INTEGER	16
	17
Reserved communicators	18
C type: MPI_Comm	19
Fortran type: INTEGER or TYPE(MPI_Comm)	20
MPI_COMM_WORLD	21
MPI_COMM_SELF	22
	23
Communicator split type constants	24
C type: const int (or unnamed enum)	25
Fortran type: INTEGER	26
MPI_COMM_TYPE_SHARED	27
	28
Results of communicator and group comparisons	29
C type: const int (or unnamed enum)	30
Fortran type: INTEGER	31
MPI_IDENT	32
MPI_CONGRUENT	33
MPI_SIMILAR	34
MPI_UNEQUAL	35
	36
Environmental inquiry info key	37
C type: MPI_Info	38
Fortran type: INTEGER or TYPE(MPI_Info)	39
MPI_INFO_ENV	40
	41
Environmental inquiry keys	42
C type: const int (or unnamed enum)	43
Fortran type: INTEGER	44
MPI_TAG_UB	45
MPI_IO	46
MPI_HOST	47
MPI_WTIME_IS_GLOBAL	48

Collective Operations

C type: MPI_Op
 Fortran type: INTEGER or TYPE(MPI_Op)

 MPI_MAX
 MPI_MIN
 MPI_SUM
 MPI_PROD
 MPI_MAXLOC
 MPI_MINLOC
 MPI_BAND
 MPI_BOR
 MPI_BXOR
 MPI_LAND
 MPI_LOR
 MPI_LXOR
 MPI_REPLACE
 MPI_NO_OP

Null Handles

C/Fortran name
 C type / Fortran type

 MPI_GROUP_NULL
 MPI_Group / INTEGER or TYPE(MPI_Group)
 MPI_COMM_NULL
 MPI_Comm / INTEGER or TYPE(MPI_Comm)
 MPI_DATATYPE_NULL
 MPI_Datatype / INTEGER or TYPE(MPI_Datatype)
 MPI_REQUEST_NULL
 MPI_Request / INTEGER or TYPE(MPI_Request)
 MPI_OP_NULL
 MPI_Op / INTEGER or TYPE(MPI_Op)
 MPI_ERRHANDLER_NULL
 MPI_Errhandler / INTEGER or TYPE(MPI_Errhandler)
 MPI_FILE_NULL
 MPI_File / INTEGER or TYPE(MPI_File)
 MPI_INFO_NULL
 MPI_Info / INTEGER or TYPE(MPI_Info)
 MPI_WIN_NULL
 MPI_Win / INTEGER or TYPE(MPI_Win)
 MPI_MESSAGE_NULL
 MPI_Message / INTEGER or TYPE(MPI_Message)

Empty group

C type: MPI_Group
 Fortran type: INTEGER or TYPE(MPI_Group)

 MPI_GROUP_EMPTY

Topologies	
C type: <code>const int</code> (or unnamed <code>enum</code>)	
Fortran type: <code>INTEGER</code>	
MPI_GRAPH	
MPI_CART	
MPI_DIST_GRAPH	
Predefined functions	
C/Fortran name	
C type	
/ Fortran type with <code>mpi</code> module	/ Fortran type with <code>mpi_f08</code> module
MPI_COMM_NULL_COPY_FN	
MPI_Comm_copy_attr_function	
/ COMM_COPY_ATTR_FUNCTION	/ PROCEDURE(MPI_Comm_copy_attr_function) ¹⁾
MPI_COMM_DUP_FN	
MPI_Comm_copy_attr_function	
/ COMM_COPY_ATTR_FUNCTION	/ PROCEDURE(MPI_Comm_copy_attr_function) ¹⁾
MPI_COMM_NULL_DELETE_FN	
MPI_Comm_delete_attr_function	
/ COMM_DELETE_ATTR_FUNCTION	/ PROCEDURE(MPI_Comm_delete_attr_function) ¹⁾
MPI_WIN_NULL_COPY_FN	
MPI_Win_copy_attr_function	
/ WIN_COPY_ATTR_FUNCTION	/ PROCEDURE(MPI_Win_copy_attr_function) ¹⁾
MPI_WIN_DUP_FN	
MPI_Win_copy_attr_function	
/ WIN_COPY_ATTR_FUNCTION	/ PROCEDURE(MPI_Win_copy_attr_function) ¹⁾
MPI_WIN_NULL_DELETE_FN	
MPI_Win_delete_attr_function	
/ WIN_DELETE_ATTR_FUNCTION	/ PROCEDURE(MPI_Win_delete_attr_function) ¹⁾
MPI_TYPE_NULL_COPY_FN	
MPI_Type_copy_attr_function	
/ TYPE_COPY_ATTR_FUNCTION	/ PROCEDURE(MPI_Type_copy_attr_function) ¹⁾
MPI_TYPE_DUP_FN	
MPI_Type_copy_attr_function	
/ TYPE_COPY_ATTR_FUNCTION	/ PROCEDURE(MPI_Type_copy_attr_function) ¹⁾
MPI_TYPE_NULL_DELETE_FN	
MPI_Type_delete_attr_function	
/ TYPE_DELETE_ATTR_FUNCTION	/ PROCEDURE(MPI_Type_delete_attr_function) ¹⁾
MPI_CONVERSION_FN_NULL	
MPI_Datarep_conversion_function	
/ DATAREP_CONVERSION_FUNCTION	/ PROCEDURE(MPI_Datarep_conversion_function) ¹⁾
¹ See the advice to implementors (on page 270) and advice to users (on page 270) on the predefined Fortran functions MPI_COMM_NULL_COPY_FN, ... in Section 6.7.2.	

Deprecated predefined functions

C/Fortran nameC type / Fortran type with `mpi` module

MPI_NULL_COPY_FN

MPI_Copy_function / COPY_FUNCTION

MPI_DUP_FN

MPI_Copy_function / COPY_FUNCTION

MPI_NULL_DELETE_FN

MPI_Delete_function / DELETE_FUNCTION

Predefined Attribute Keys

C type: `const int` (or unnamed `enum`)Fortran type: `INTEGER`

MPI_APPNUM

MPI_LASTUSED CODE

MPI_UNIVERSE_SIZE

MPI_WIN_BASE

MPI_WIN_DISP_UNIT

MPI_WIN_SIZE

MPI_WIN_CREATE_FLAVOR

MPI_WIN_MODEL

MPI Window Create Flavors

C type: `const int` (or unnamed `enum`)Fortran type: `INTEGER`

MPI_WIN_FLAVOR_CREATE

MPI_WIN_FLAVOR_ALLOCATE

MPI_WIN_FLAVOR_DYNAMIC

MPI_WIN_FLAVOR_SHARED

MPI Window Models

C type: `const int` (or unnamed `enum`)Fortran type: `INTEGER`

MPI_WIN_SEPARATE

MPI_WIN_UNIFIED

Mode Constants		1
C type: <code>const int</code> (or unnamed <code>enum</code>)		2
Fortran type: <code>INTEGER</code>		3
<hr/> MPI_MODE_APPEND		4
MPI_MODE_CREATE		5
MPI_MODE_DELETE_ON_CLOSE		6
MPI_MODE_EXCL		7
MPI_MODE_NOCHECK		8
MPI_MODE_NOPRECEDE		9
MPI_MODE_NOPUT		10
MPI_MODE_NOSTORE		11
MPI_MODE_NOSUCCEED		12
MPI_MODE_RDONLY		13
MPI_MODE_RDWR		14
MPI_MODE_SEQUENTIAL		15
MPI_MODE_UNIQUE_OPEN		16
MPI_MODE_WRONLY		17
<hr/>		18
Datatype Decoding Constants		19
C type: <code>const int</code> (or unnamed <code>enum</code>)		20
Fortran type: <code>INTEGER</code>		21
<hr/> MPI_COMBINER_CONTIGUOUS		22
MPI_COMBINER_DARRAY		23
MPI_COMBINER_DUP		24
MPI_COMBINER_F90_COMPLEX		25
MPI_COMBINER_F90_INTEGER		26
MPI_COMBINER_F90_REAL		27
MPI_COMBINER_HINDEXED		28
MPI_COMBINER_HVECTOR		29
MPI_COMBINER_INDEXED_BLOCK		30
MPI_COMBINER_HINDEXED_BLOCK		31
MPI_COMBINER_INDEXED		32
MPI_COMBINER_NAMED		33
MPI_COMBINER_RESIZED		34
MPI_COMBINER_STRUCT		35
MPI_COMBINER_SUBARRAY		36
MPI_COMBINER_VECTOR		37
<hr/>		38
Threads Constants		39
C type: <code>const int</code> (or unnamed <code>enum</code>)		40
Fortran type: <code>INTEGER</code>		41
<hr/> MPI_THREAD_FUNNELED		42
MPI_THREAD_MULTIPLE		43
MPI_THREAD_SERIALIZED		44
MPI_THREAD_SINGLE		45
<hr/>		46
		47
		48

File Operation Constants, Part 1

C type: `const MPI_Offset` (or unnamed `enum`)
 Fortran type: `INTEGER (KIND=MPI_OFFSET_KIND)`

`MPI_DISPLACEMENT_CURRENT`

File Operation Constants, Part 2

C type: `const int` (or unnamed `enum`)
 Fortran type: `INTEGER`

`MPI_DISTRIBUTE_BLOCK`
`MPI_DISTRIBUTE_CYCLIC`
`MPI_DISTRIBUTE_DFLT_DARG`
`MPI_DISTRIBUTE_NONE`
`MPI_ORDER_C`
`MPI_ORDER_FORTRAN`
`MPI_SEEK_CUR`
`MPI_SEEK_END`
`MPI_SEEK_SET`

F90 Datatype Matching Constants

C type: `const int` (or unnamed `enum`)
 Fortran type: `INTEGER`

`MPI_TYPECLASS_COMPLEX`
`MPI_TYPECLASS_INTEGER`
`MPI_TYPECLASS_REAL`

Constants Specifying Empty or Ignored Input

C/Fortran name
 C type / Fortran type¹

`MPI_ARGVS_NULL`
`char***` / 2-dim. array of `CHARACTER*(*)`
`MPI_ARGV_NULL`
`char**` / array of `CHARACTER*(*)`

`MPI_ERRCODES_IGNORE`
`int*` / `INTEGER` array
`MPI_STATUSES_IGNORE`
`MPI_Status*` / `INTEGER, DIMENSION(MPI_STATUS_SIZE,*)`
or `TYPE(MPI_Status), DIMENSION(*)`
`MPI_STATUS_IGNORE`
`MPI_Status*` / `INTEGER, DIMENSION(MPI_STATUS_SIZE)`
or `TYPE(MPI_Status)`
`MPI_UNWEIGHTED`
`int*` / `INTEGER` array
`MPI_WEIGHTS_EMPTY`
`int*` / `INTEGER` array

¹ Note that in Fortran these constants are not usable for initialization expressions or assignment. See Section [2.5.4](#).

C Constants Specifying Ignored Input (no Fortran)

C type: MPI_Fint*	equivalent to Fortran
MPI_F_STATUSES_IGNORE	MPI_STATUSES_IGNORE in <code>mpi / mpif.h</code>
MPI_F_STATUS_IGNORE	MPI_STATUS_IGNORE in <code>mpi / mpif.h</code>
C type: MPI_F08_status*	equivalent to Fortran
MPI_F08_STATUSES_IGNORE	MPI_STATUSES_IGNORE in <code>mpi_f08</code>
MPI_F08_STATUS_IGNORE	MPI_STATUS_IGNORE in <code>mpi_f08</code>

C preprocessor Constants and Fortran ParametersC type: C-preprocessor macro that expands to an `int` valueFortran type: `INTEGER`

MPI_SUBVERSION

MPI_VERSION

Null handles used in the MPI tool information interface

MPI_T_ENUM_NULL

MPI_T_enum

MPI_T_CVAR_HANDLE_NULL

MPI_T_cvar_handle

MPI_T_PVAR_HANDLE_NULL

MPI_T_pvar_handle

MPI_T_PVAR_SESSION_NULL

MPI_T_pvar_session

Verbosity Levels in the MPI tool information interfaceC type: `const int` (or unnamed `enum`)

MPI_T_VERBOSITY_USER_BASIC

MPI_T_VERBOSITY_USER_DETAIL

MPI_T_VERBOSITY_USER_ALL

MPI_T_VERBOSITY_TUNER_BASIC

MPI_T_VERBOSITY_TUNER_DETAIL

MPI_T_VERBOSITY_TUNER_ALL

MPI_T_VERBOSITY_MPIDEV_BASIC

MPI_T_VERBOSITY_MPIDEV_DETAIL

MPI_T_VERBOSITY_MPIDEV_ALL

Constants to identify associations of variables in the MPI tool information interface

C type: `const int` (or unnamed `enum`)

`MPI_T_BIND_NO_OBJECT`
`MPI_T_BIND_MPI_COMM`
`MPI_T_BIND_MPI_DATATYPE`
`MPI_T_BIND_MPI_ERRHANDLER`
`MPI_T_BIND_MPI_FILE`
`MPI_T_BIND_MPI_GROUP`
`MPI_T_BIND_MPI_OP`
`MPI_T_BIND_MPI_REQUEST`
`MPI_T_BIND_MPI_WIN`
`MPI_T_BIND_MPI_MESSAGE`
`MPI_T_BIND_MPI_INFO`

Constants describing the scope of a control variable in the MPI tool information interface

C type: `const int` (or unnamed `enum`)

`MPI_T_SCOPE_CONSTANT`
`MPI_T_SCOPE_READONLY`
`MPI_T_SCOPE_LOCAL`
`MPI_T_SCOPE_GROUP`
`MPI_T_SCOPE_GROUP_EQ`
`MPI_T_SCOPE_ALL`
`MPI_T_SCOPE_ALL_EQ`

Additional constants used by the MPI tool information interface

C type: `MPI_T_pvar_handle`

`MPI_T_PVAR_ALL_HANDLES`

Performance variables classes used by the MPI tool information interface

C type: `const int` (or unnamed `enum`)

`MPI_T_PVAR_CLASS_STATE`
`MPI_T_PVAR_CLASS_LEVEL`
`MPI_T_PVAR_CLASS_SIZE`
`MPI_T_PVAR_CLASS_PERCENTAGE`
`MPI_T_PVAR_CLASS_HIGHWATERMARK`
`MPI_T_PVAR_CLASS_LOWWATERMARK`
`MPI_T_PVAR_CLASS_COUNTER`
`MPI_T_PVAR_CLASS_AGGREGATE`
`MPI_T_PVAR_CLASS_TIMER`
`MPI_T_PVAR_CLASS_GENERIC`

A.1.2 Types

The following are defined C type definitions, included in the file `mpi.h`.

```

/* C opaque types */
MPI_Aint
MPI_Count
MPI_Fint
MPI_Offset
MPI_Status
MPI_F08_status

/* C handles to assorted structures */
MPI_Comm
MPI_Datatype
MPI_Errhandler
MPI_File
MPI_Group
MPI_Info
MPI_Message
MPI_Op
MPI_Request
MPI_Win

/* Types for the MPI_T interface */
MPI_T_enum
MPI_T_cvar_handle
MPI_T_pvar_handle
MPI_T_pvar_session

The following are defined Fortran type definitions, included in the mpi_f08 and mpi
modules.

! Fortran opaque types in the mpi_f08 and mpi modules
TYPE(MPI_Status)

! Fortran handles in the mpi_f08 and mpi modules
TYPE(MPI_Comm)
TYPE(MPI_Datatype)
TYPE(MPI_Errhandler)
TYPE(MPI_File)
TYPE(MPI_Group)
TYPE(MPI_Info)
TYPE(MPI_Message)
TYPE(MPI_Op)
TYPE(MPI_Request)
TYPE(MPI_Win)

```

A.1.3 Prototype Definitions

C Bindings

The following are defined C typedefs for user-defined functions, also included in the file `mpi.h`.

```
/* prototypes for user-defined functions */
typedef void MPI_User_function(void *invec, void *inoutvec, int *len,
                               MPI_Datatype *datatype);

typedef int MPI_Comm_copy_attr_function(MPI_Comm oldcomm,
                                         int comm_keyval, void *extra_state, void *attribute_val_in,
                                         void *attribute_val_out, int *flag);
typedef int MPI_Comm_delete_attr_function(MPI_Comm comm,
                                         int comm_keyval, void *attribute_val, void *extra_state);

typedef int MPI_Win_copy_attr_function(MPI_Win oldwin, int win_keyval,
                                         void *extra_state, void *attribute_val_in,
                                         void *attribute_val_out, int *flag);
typedef int MPI_Win_delete_attr_function(MPI_Win win, int win_keyval,
                                         void *attribute_val, void *extra_state);

typedef int MPI_Type_copy_attr_function(MPI_Datatype oldtype,
                                         int type_keyval, void *extra_state,
                                         void *attribute_val_in, void *attribute_val_out, int *flag);
typedef int MPI_Type_delete_attr_function(MPI_Datatype datatype,
                                         int type_keyval, void *attribute_val, void *extra_state);

typedef void MPI_Comm_errhandler_function(MPI_Comm *, int *, ...);
typedef void MPI_Win_errhandler_function(MPI_Win *, int *, ...);
typedef void MPI_File_errhandler_function(MPI_File *, int *, ...);

typedef int MPI_Grequest_query_function(void *extra_state,
                                         MPI_Status *status);
typedef int MPI_Grequest_free_function(void *extra_state);
typedef int MPI_Grequest_cancel_function(void *extra_state, int complete);

typedef int MPI_Datarep_extent_function(MPI_Datatype datatype,
                                         MPI_Aint *file_extent, void *extra_state);
typedef int MPI_Datarep_conversion_function(void *userbuf,
                                         MPI_Datatype datatype, int count, void *filebuf,
                                         MPI_Offset position, void *extra_state);
```

Fortran 2008 Bindings with the `mpi_f08` Module

The callback prototypes when using the Fortran `mpi_f08` module are shown below:

The user-function argument to `MPI_Op_create` should be declared according to:

ABSTRACT INTERFACE


```

SUBROUTINE MPI_User_function(invec, inoutvec, len, datatype)
  USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
  TYPE(C_PTR), VALUE :: invec, inoutvec
  INTEGER :: len
  TYPE(MPI_Datatype) :: datatype

```

The copy and delete function arguments to MPI_Comm_create_keyval should be declared according to:

ABSTRACT INTERFACE

```

SUBROUTINE MPI_Comm_copy_attr_function(oldcomm, comm_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
  TYPE(MPI_Comm) :: oldcomm
  INTEGER :: comm_keyval, ierror
  INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in,
  attribute_val_out
  LOGICAL :: flag

```

ABSTRACT INTERFACE

```

SUBROUTINE MPI_Comm_delete_attr_function(comm, comm_keyval,
  attribute_val, extra_state, ierror)
  TYPE(MPI_Comm) :: comm
  INTEGER :: comm_keyval, ierror
  INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state

```

The copy and delete function arguments to MPI_Win_create_keyval should be declared according to:

ABSTRACT INTERFACE

```

SUBROUTINE MPI_Win_copy_attr_function(oldwin, win_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
  TYPE(MPI_Win) :: oldwin
  INTEGER :: win_keyval, ierror
  INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in,
  attribute_val_out
  LOGICAL :: flag

```

ABSTRACT INTERFACE

```

SUBROUTINE MPI_Win_delete_attr_function(win, win_keyval, attribute_val,
  extra_state, ierror)
  TYPE(MPI_Win) :: win
  INTEGER :: win_keyval, ierror
  INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state

```

The copy and delete function arguments to MPI_Type_create_keyval should be declared according to:

ABSTRACT INTERFACE

```

SUBROUTINE MPI_Type_copy_attr_function(oldtype, type_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
  TYPE(MPI_Datatype) :: oldtype
  INTEGER :: type_keyval, ierror
  INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in,

```

```

1      attribute_val_out
2      LOGICAL :: flag

```

ABSTRACT INTERFACE

```

5      SUBROUTINE MPI_Type_delete_attr_function(datatype, type_keyval,
6      attribute_val, extra_state, ierror)
7          TYPE(MPI_Datatype) :: datatype
8          INTEGER :: type_keyval, ierror
9          INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state

```

The handler-function argument to MPI_Comm_create_errhandler should be declared like this:

ABSTRACT INTERFACE

```

13     SUBROUTINE MPI_Comm_errhandler_function(comm, error_code)
14         TYPE(MPI_Comm) :: comm
15         INTEGER :: error_code

```

The handler-function argument to MPI_Win_create_errhandler should be declared like this:

ABSTRACT INTERFACE

```

20     SUBROUTINE MPI_Win_errhandler_function(win, error_code)
21         TYPE(MPI_Win) :: win
22         INTEGER :: error_code

```

The handler-function argument to MPI_File_create_errhandler should be declared like this:

ABSTRACT INTERFACE

```

26     SUBROUTINE MPI_File_errhandler_function(file, error_code)
27         TYPE(MPI_File) :: file
28         INTEGER :: error_code

```

The query, free, and cancel function arguments to MPI_Grequest_start should be declared according to:

ABSTRACT INTERFACE

```

33     SUBROUTINE MPI_Grequest_query_function(extra_state, status, ierror)
34         TYPE(MPI_Status) :: status
35         INTEGER :: ierror
36         INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state

```

ABSTRACT INTERFACE

```

38     SUBROUTINE MPI_Grequest_free_function(extra_state, ierror)
39         INTEGER :: ierror
40         INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state

```

ABSTRACT INTERFACE

```

43     SUBROUTINE MPI_Grequest_cancel_function(extra_state, complete, ierror)
44         INTEGER :: ierror
45         INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state
46         LOGICAL :: complete

```

The extent and conversion function arguments to MPI_Register_datarep should be de-

clared according to:

ABSTRACT INTERFACE

```
SUBROUTINE MPI_Datarep_extent_function(datatype, extent, extra_state,
ierror)
```

```
    TYPE(MPI_Datatype) :: datatype
```

```
    INTEGER(KIND=MPI_ADDRESS_KIND) :: extent, extra_state
```

```
    INTEGER :: ierror
```

ABSTRACT INTERFACE

```
SUBROUTINE MPI_Datarep_conversion_function(userbuf, datatype, count,
filebuf, position, extra_state, ierror)
```

```
    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
```

```
    TYPE(C_PTR), VALUE :: userbuf, filebuf
```

```
    TYPE(MPI_Datatype) :: datatype
```

```
    INTEGER :: count, ierror
```

```
    INTEGER(KIND=MPI_OFFSET_KIND) :: position
```

```
    INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state
```

Fortran Bindings with mpif.h or the mpi Module

With the Fortran `mpi` module or `mpif.h`, here are examples of how each of the user-defined subroutines should be declared.

The user-function argument to `MPI_OP_CREATE` should be declared like this:

```
SUBROUTINE USER_FUNCTION(INVEC, INOUTVEC, LEN, DATATYPE)
```

```
    <type> INVEC(LEN), INOUTVEC(LEN)
```

```
    INTEGER LEN, DATATYPE
```

The copy and delete function arguments to `MPI_COMM_CREATE_KEYVAL` should be declared like these:

```
SUBROUTINE COMM_COPY_ATTR_FUNCTION(OLDCOMM, COMM_KEYVAL, EXTRA_STATE,
```

```
    ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
```

```
    INTEGER OLDCOMM, COMM_KEYVAL, IERROR
```

```
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
```

```
    ATTRIBUTE_VAL_OUT
```

```
    LOGICAL FLAG
```

```
SUBROUTINE COMM_DELETE_ATTR_FUNCTION(COMM, COMM_KEYVAL, ATTRIBUTE_VAL,
```

```
    EXTRA_STATE, IERROR)
```

```
    INTEGER COMM, COMM_KEYVAL, IERROR
```

```
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE
```

The copy and delete function arguments to `MPI_WIN_CREATE_KEYVAL` should be declared like these:

```
SUBROUTINE WIN_COPY_ATTR_FUNCTION(OLDWIN, WIN_KEYVAL, EXTRA_STATE,
```

```
    ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
```

```
    INTEGER OLDWIN, WIN_KEYVAL, IERROR
```

```

1      INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
2          ATTRIBUTE_VAL_OUT
3      LOGICAL FLAG

```

```

5  SUBROUTINE WIN_DELETE_ATTR_FUNCTION(WIN, WIN_KEYVAL, ATTRIBUTE_VAL,
6      EXTRA_STATE, IERROR)
7      INTEGER WIN, WIN_KEYVAL, IERROR
8      INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE

```

The copy and delete function arguments to MPI_TYPE_CREATE_KEYVAL should be declared like these:

```

12
13 SUBROUTINE TYPE_COPY_ATTR_FUNCTION(OLDTYPE, TYPE_KEYVAL, EXTRA_STATE,
14     ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
15     INTEGER OLDTYPE, TYPE_KEYVAL, IERROR
16     INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE,
17         ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT
18     LOGICAL FLAG
19
20 SUBROUTINE TYPE_DELETE_ATTR_FUNCTION(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL,
21     EXTRA_STATE, IERROR)
22     INTEGER DATATYPE, TYPE_KEYVAL, IERROR
23     INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE

```

The handler-function argument to MPI_COMM_CREATE_ERRHANDLER should be declared like this:

```

27 SUBROUTINE COMM_ERRHANDLER_FUNCTION(COMM, ERROR_CODE)
28     INTEGER COMM, ERROR_CODE

```

The handler-function argument to MPI_WIN_CREATE_ERRHANDLER should be declared like this:

```

33 SUBROUTINE WIN_ERRHANDLER_FUNCTION(WIN, ERROR_CODE)
34     INTEGER WIN, ERROR_CODE

```

The handler-function argument to MPI_FILE_CREATE_ERRHANDLER should be declared like this:

```

39 SUBROUTINE FILE_ERRHANDLER_FUNCTION(FILE, ERROR_CODE)
40     INTEGER FILE, ERROR_CODE

```

The query, free, and cancel function arguments to MPI_GREQUEST_START should be declared like these:

```

45 SUBROUTINE GREQUEST_QUERY_FUNCTION(EXTRA_STATE, STATUS, IERROR)
46     INTEGER STATUS(MPI_STATUS_SIZE), IERROR
47     INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

```

```

SUBROUTINE GREQUEST_FREE_FUNCTION(EXTRA_STATE, IERROR)
  INTEGER IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

```

```

SUBROUTINE GREQUEST_CANCEL_FUNCTION(EXTRA_STATE, COMPLETE, IERROR)
  INTEGER IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
  LOGICAL COMPLETE

```

The extent and conversion function arguments to MPI_REGISTER_DATAREP should be declared like these:

```

SUBROUTINE DATAREP_EXTENT_FUNCTION(DATATYPE, EXTENT, EXTRA_STATE, IERROR)
  INTEGER DATATYPE, IERROR
  INTEGER(KIND=MPI_ADDRESS_KIND) EXTENT, EXTRA_STATE

```

```

SUBROUTINE DATAREP_CONVERSION_FUNCTION(USERBUF, DATATYPE, COUNT, FILEBUF,
  POSITION, EXTRA_STATE, IERROR)
  <TYPE> USERBUF(*), FILEBUF(*)
  INTEGER COUNT, DATATYPE, IERROR
  INTEGER(KIND=MPI_OFFSET_KIND) POSITION
  INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

```

A.1.4 Deprecated Prototype Definitions

The following are defined C typedefs for deprecated user-defined functions, also included in the file `mpi.h`.

```

/* prototypes for user-defined functions */
typedef int MPI_Copy_function(MPI_Comm oldcomm, int keyval,
  void *extra_state, void *attribute_val_in,
  void *attribute_val_out, int *flag);
typedef int MPI_Delete_function(MPI_Comm comm, int keyval,
  void *attribute_val, void *extra_state);

```

The following are deprecated Fortran user-defined callback subroutine prototypes. The deprecated copy and delete function arguments to MPI_KEYVAL_CREATE should be declared like these:

```

SUBROUTINE COPY_FUNCTION(OLDCOMM, KEYVAL, EXTRA_STATE,
  ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERR)
  INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
  ATTRIBUTE_VAL_OUT, IERR
  LOGICAL FLAG

SUBROUTINE DELETE_FUNCTION(COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR)
  INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR

```

A.1.5 Info Keys

The following info keys are reserved. They are strings.

access_style
accumulate_ops
accumulate_ordering
alloc_shared_noncontig
appnum
arch
cb_block_size
cb_buffer_size
cb_nodes
chunked_item
chunked_size
chunked
collective_buffering
file_perm
filename
file
host
io_node_list
ip_address
ip_port
nb_proc
no_locks
num_io_nodes
path
same_disp_unit
same_size
soft
striping_factor
striping_unit
wdir

A.1.6 Info Values

The following info values are reserved. They are strings.

false
random
rar
raw
read_mostly
read_once
reverse_sequential
same_op
same_op_no_op
sequential

true	1
war	2
waw	3
write_mostly	4
write_once	5
	6
	7
	8
	9
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
	20
	21
	22
	23
	24
	25
	26
	27
	28
	29
	30
	31
	32
	33
	34
	35
	36
	37
	38
	39
	40
	41
	42
	43
	44
	45
	46
	47
	48

A.2 C Bindings

A.2.1 Point-to-Point Communication C Bindings

```

1  int MPI_Bsend(const void* buf, int count, MPI_Datatype datatype, int dest,
2      int tag, MPI_Comm comm)
3
4  int MPI_Bsend_init(const void* buf, int count, MPI_Datatype datatype,
5      int dest, int tag, MPI_Comm comm, MPI_Request *request)
6
7  int MPI_Buffer_attach(void* buffer, int size)
8
9  int MPI_Buffer_detach(void* buffer_addr, int* size)
10
11 int MPI_Cancel(MPI_Request *request)
12
13 int MPI_Get_count(const MPI_Status *status, MPI_Datatype datatype,
14     int *count)
15
16 int MPI_Ibsend(const void* buf, int count, MPI_Datatype datatype, int dest,
17     int tag, MPI_Comm comm, MPI_Request *request)
18
19 int MPI_Improbe(int source, int tag, MPI_Comm comm, int *flag,
20     MPI_Message *message, MPI_Status *status)
21
22 int MPI_Imrecv(void* buf, int count, MPI_Datatype datatype,
23     MPI_Message *message, MPI_Request *request)
24
25 int MPI_Iprobe(int source, int tag, MPI_Comm comm, int *flag,
26     MPI_Status *status)
27
28 int MPI_Irecv(void* buf, int count, MPI_Datatype datatype, int source,
29     int tag, MPI_Comm comm, MPI_Request *request)
30
31 int MPI_Irsend(const void* buf, int count, MPI_Datatype datatype, int dest,
32     int tag, MPI_Comm comm, MPI_Request *request)
33
34 int MPI_Isend(const void* buf, int count, MPI_Datatype datatype, int dest,
35     int tag, MPI_Comm comm, MPI_Request *request)
36
37 int MPI_Issend(const void* buf, int count, MPI_Datatype datatype, int dest,
38     int tag, MPI_Comm comm, MPI_Request *request)
39
40 int MPI_Mprobe(int source, int tag, MPI_Comm comm, MPI_Message *message,
41     MPI_Status *status)
42
43 int MPI_Mrecv(void* buf, int count, MPI_Datatype datatype,
44     MPI_Message *message, MPI_Status *status)
45
46 int MPI_Probe(int source, int tag, MPI_Comm comm, MPI_Status *status)
47
48 int MPI_Recv_init(void* buf, int count, MPI_Datatype datatype, int source,
49     int tag, MPI_Comm comm, MPI_Request *request)
50
51 int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source,
52     int tag, MPI_Comm comm, MPI_Status *status)

```



```

int MPI_Request_free(MPI_Request *request) 1
int MPI_Request_get_status(MPI_Request request, int *flag, 2
    MPI_Status *status) 3
int MPI_Rsend(const void* buf, int count, MPI_Datatype datatype, int dest, 4
    int tag, MPI_Comm comm) 5
int MPI_Rsend_init(const void* buf, int count, MPI_Datatype datatype, 6
    int dest, int tag, MPI_Comm comm, MPI_Request *request) 7
int MPI_Send(const void* buf, int count, MPI_Datatype datatype, int dest, 8
    int tag, MPI_Comm comm) 9
int MPI_Send_init(const void* buf, int count, MPI_Datatype datatype, 10
    int dest, int tag, MPI_Comm comm, MPI_Request *request) 11
int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype, 12
    int dest, int sendtag, void *recvbuf, int recvcount, 13
    MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm, 14
    MPI_Status *status) 15
int MPI_Sendrecv_replace(void* buf, int count, MPI_Datatype datatype, 16
    int dest, int sendtag, int source, int recvtag, MPI_Comm comm, 17
    MPI_Status *status) 18
int MPI_Ssend(const void* buf, int count, MPI_Datatype datatype, int dest, 19
    int tag, MPI_Comm comm) 20
int MPI_Ssend_init(const void* buf, int count, MPI_Datatype datatype, 21
    int dest, int tag, MPI_Comm comm, MPI_Request *request) 22
int MPI_Startall(int count, MPI_Request array_of_requests[]) 23
int MPI_Start(MPI_Request *request) 24
int MPI_Testall(int count, MPI_Request array_of_requests[], int *flag, 25
    MPI_Status array_of_statuses[]) 26
int MPI_Testany(int count, MPI_Request array_of_requests[], int *index, 27
    int *flag, MPI_Status *status) 28
int MPI_Test_cancelled(const MPI_Status *status, int *flag) 29
int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status) 30
int MPI_Testsome(int incount, MPI_Request array_of_requests[], 31
    int *outcount, int array_of_indices[], 32
    MPI_Status array_of_statuses[]) 33
int MPI_Waitall(int count, MPI_Request array_of_requests[], 34
    MPI_Status array_of_statuses[]) 35
int MPI_Waitany(int count, MPI_Request array_of_requests[], int *index, 36
    MPI_Status *status) 37

```

```
1  int MPI_Wait(MPI_Request *request, MPI_Status *status)
```

```
2
3  int MPI_Waitsome(int incout, MPI_Request array_of_requests[],
4                  int *outcount, int array_of_indices[],
5                  MPI_Status array_of_statuses[])
```

7 A.2.2 Datatypes C Bindings

```
8
9  int MPI_Get_address(const void *location, MPI_Aint *address)
```

```
10 int MPI_Get_elements(const MPI_Status *status, MPI_Datatype datatype,
11                     int *count)
```

```
12
13 int MPI_Get_elements_x(const MPI_Status *status, MPI_Datatype datatype,
14                       MPI_Count *count)
```

```
15 int MPI_Pack(const void* inbuf, int incout, MPI_Datatype datatype,
16             void *outbuf, int outsize, int *position, MPI_Comm comm)
```

```
17
18 int MPI_Pack_external(const char datarep[], const void *inbuf, int incout,
19                     MPI_Datatype datatype, void *outbuf, MPI_Aint outsize,
20                     MPI_Aint *position)
```

```
21 int MPI_Pack_external_size(const char datarep[], int incout,
22                          MPI_Datatype datatype, MPI_Aint *size)
```

```
23
24 int MPI_Pack_size(int incout, MPI_Datatype datatype, MPI_Comm comm,
25                  int *size)
```

```
26
27 int MPI_Type_commit(MPI_Datatype *datatype)
```

```
28 int MPI_Type_contiguous(int count, MPI_Datatype oldtype,
29                        MPI_Datatype *newtype)
```

```
30
31 int MPI_Type_create_darray(int size, int rank, int ndims, const
32                          int array_of_gsizes[], const int array_of_distrib[], const
33                          int array_of_dargs[], const int array_of_psize[], int order,
34                          MPI_Datatype oldtype, MPI_Datatype *newtype)
```

```
35 int MPI_Type_create_hindexed_block(int count, int blocklength, const
36                                  MPI_Aint array_of_displacements[], MPI_Datatype oldtype,
37                                  MPI_Datatype *newtype)
```

```
38
39 int MPI_Type_create_hindexed(int count, const int array_of_blocklengths[],
40                             const MPI_Aint array_of_displacements[], MPI_Datatype oldtype,
41                             MPI_Datatype *newtype)
```

```
42 int MPI_Type_create_hvector(int count, int blocklength, MPI_Aint stride,
43                             MPI_Datatype oldtype, MPI_Datatype *newtype)
```

```
44
45 int MPI_Type_create_indexed_block(int count, int blocklength, const
46                                  int array_of_displacements[], MPI_Datatype oldtype,
47                                  MPI_Datatype *newtype)
```

```
48
```

```

int MPI_Type_create_resized(MPI_Datatype oldtype, MPI_Aint lb, MPI_Aint 1
    extent, MPI_Datatype *newtype) 2
3
int MPI_Type_create_struct(int count, const int array_of_blocklengths[], 4
    const MPI_Aint array_of_displacements[], const 5
    MPI_Datatype array_of_types[], MPI_Datatype *newtype) 6
7
int MPI_Type_create_subarray(int ndims, const int array_of_sizes[], const 8
    int array_of_subsizes[], const int array_of_starts[], int 9
    order, MPI_Datatype oldtype, MPI_Datatype *newtype) 10
11
int MPI_Type_dup(MPI_Datatype oldtype, MPI_Datatype *newtype) 12
13
int MPI_Type_free(MPI_Datatype *datatype) 14
15
int MPI_Type_get_contents(MPI_Datatype datatype, int max_integers, 16
    int max_addresses, int max_datatypes, int array_of_integers[], 17
    MPI_Aint array_of_addresses[], 18
    MPI_Datatype array_of_datatypes[]) 19
20
int MPI_Type_get_envelope(MPI_Datatype datatype, int *num_integers, 21
    int *num_addresses, int *num_datatypes, int *combiner) 22
23
int MPI_Type_get_extent(MPI_Datatype datatype, MPI_Aint *lb, 24
    MPI_Aint *extent) 25
26
int MPI_Type_get_extent_x(MPI_Datatype datatype, MPI_Count *lb, 27
    MPI_Count *extent) 28
29
int MPI_Type_get_true_extent(MPI_Datatype datatype, MPI_Aint *true_lb, 30
    MPI_Aint *true_extent) 31
32
int MPI_Type_get_true_extent_x(MPI_Datatype datatype, MPI_Count *true_lb, 33
    MPI_Count *true_extent) 34
35
int MPI_Type_indexed(int count, const int array_of_blocklengths[], const 36
    int array_of_displacements[], MPI_Datatype oldtype, 37
    MPI_Datatype *newtype) 38
39
int MPI_Type_size(MPI_Datatype datatype, int *size) 40
41
int MPI_Type_size_x(MPI_Datatype datatype, MPI_Count *size) 42
43
int MPI_Type_vector(int count, int blocklength, int stride, 44
    MPI_Datatype oldtype, MPI_Datatype *newtype) 45
46
int MPI_Unpack(const void* inbuf, int insize, int *position, void *outbuf, 47
    int outcount, MPI_Datatype datatype, MPI_Comm comm) 48
49
int MPI_Unpack_external(const char datarep[], const void *inbuf, 50
    MPI_Aint insize, MPI_Aint *position, void *outbuf, 51
    int outcount, MPI_Datatype datatype) 52
53
MPI_Aint MPI_Aint_add(MPI_Aint base, MPI_Aint disp) 54
55
MPI_Aint MPI_Aint_diff(MPI_Aint addr1, MPI_Aint addr2) 56
57

```

A.2.3 Collective Communication C Bindings

```

1  int MPI_Allgather(const void* sendbuf, int sendcount,
2                      MPI_Datatype sendtype, void* recvbuf, int recvcount,
3                      MPI_Datatype recvtype, MPI_Comm comm)
4
5
6  int MPI_Allgatherv(const void* sendbuf, int sendcount,
7                      MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
8                      const int displs[], MPI_Datatype recvtype, MPI_Comm comm)
9
10 int MPI_Allreduce(const void* sendbuf, void* recvbuf, int count,
11                   MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
12
13 int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
14                  void* recvbuf, int recvcount, MPI_Datatype recvtype,
15                  MPI_Comm comm)
16
17 int MPI_Alltoallv(const void* sendbuf, const int sendcounts[], const
18                   int sdispls[], MPI_Datatype sendtype, void* recvbuf, const
19                   int recvcounts[], const int rdispls[], MPI_Datatype recvtype,
20                   MPI_Comm comm)
21
22 int MPI_Alltoallw(const void* sendbuf, const int sendcounts[], const
23                   int sdispls[], const MPI_Datatype sendtypes[], void* recvbuf,
24                   const int recvcounts[], const int rdispls[], const
25                   MPI_Datatype recvtypes[], MPI_Comm comm)
26
27 int MPI_Barrier(MPI_Comm comm)
28
29 int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root,
30               MPI_Comm comm)
31
32 int MPI_Exscan(const void* sendbuf, void* recvbuf, int count,
33                MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
34
35 int MPI_Gather(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
36                void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
37                MPI_Comm comm)
38
39 int MPI_Gatherv(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
40                 void* recvbuf, const int recvcounts[], const int displs[],
41                 MPI_Datatype recvtype, int root, MPI_Comm comm)
42
43 int MPI_Iallgather(const void* sendbuf, int sendcount,
44                   MPI_Datatype sendtype, void* recvbuf, int recvcount,
45                   MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)
46
47 int MPI_Iallgatherv(const void* sendbuf, int sendcount,
48                     MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
49                     const int displs[], MPI_Datatype recvtype, MPI_Comm comm,
50                     MPI_Request* request)
51
52 int MPI_Iallreduce(const void* sendbuf, void* recvbuf, int count,
53                   MPI_Datatype datatype, MPI_Op op, MPI_Comm comm,
54                   MPI_Request* request)

```

```

        MPI_Request *request)
1
2
int MPI_Ialltoall(const void* sendbuf, int sendcount,
3
4
5
        MPI_Datatype sendtype, void* recvbuf, int recvcount,
        MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)
6
7
8
9
10
int MPI_Ialltoallv(const void* sendbuf, const int sendcounts[], const
11
12
13
        int sdispls[], MPI_Datatype sendtype, void* recvbuf, const
        int recvcounts[], const int rdispls[], MPI_Datatype recvtype,
        MPI_Comm comm, MPI_Request *request)
14
15
16
int MPI_Ialltoallw(const void* sendbuf, const int sendcounts[], const
17
18
19
        int sdispls[], const MPI_Datatype sendtypes[], void* recvbuf,
        const int recvcounts[], const int rdispls[], const
        MPI_Datatype recvtypes[], MPI_Comm comm, MPI_Request *request)
20
21
22
int MPI_Ibcast(void* buffer, int count, MPI_Datatype datatype, int root,
23
24
25
        MPI_Comm comm, MPI_Request *request)
26
27
28
int MPI_Iexscan(const void* sendbuf, void* recvbuf, int count,
29
30
31
        MPI_Datatype datatype, MPI_Op op, MPI_Comm comm,
        MPI_Request *request)
32
33
34
int MPI_Igather(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
35
36
37
        void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
        MPI_Comm comm, MPI_Request *request)
38
39
40
int MPI_Igatherv(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
41
42
43
        void* recvbuf, const int recvcounts[], const int displs[],
        MPI_Datatype recvtype, int root, MPI_Comm comm,
        MPI_Request *request)
44
45
46
int MPI_Ireduce(const void* sendbuf, void* recvbuf, int count,
47
48
        MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm,
        MPI_Request *request)
int MPI_Ireduce_scatter_block(const void* sendbuf, void* recvbuf,
        int recvcount, MPI_Datatype datatype, MPI_Op op,
        MPI_Comm comm, MPI_Request *request)
int MPI_Ireduce_scatter(const void* sendbuf, void* recvbuf, const
        int recvcounts[], MPI_Datatype datatype, MPI_Op op,
        MPI_Comm comm, MPI_Request *request)
int MPI_Iscan(const void* sendbuf, void* recvbuf, int count,
        MPI_Datatype datatype, MPI_Op op, MPI_Comm comm,
        MPI_Request *request)
int MPI_Iscatter(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
        void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
        MPI_Comm comm, MPI_Request *request)

```

```

1  int MPI_Iscatterv(const void* sendbuf, const int sendcounts[], const
2      int displs[], MPI_Datatype sendtype, void* recvbuf,
3      int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm,
4      MPI_Request *request)
5
6  int MPI_Op_commutative(MPI_Op op, int *commute)
7
8  int MPI_Op_create(MPI_User_function* user_fn, int commute, MPI_Op* op)
9
10 int MPI_Op_free(MPI_Op *op)
11
12 int MPI_Reduce(const void* sendbuf, void* recvbuf, int count,
13     MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
14
15 int MPI_Reduce_local(const void* inbuf, void* inoutbuf, int count,
16     MPI_Datatype datatype, MPI_Op op)
17
18 int MPI_Reduce_scatter_block(const void* sendbuf, void* recvbuf,
19     int recvcount, MPI_Datatype datatype, MPI_Op op,
20     MPI_Comm comm)
21
22 int MPI_Reduce_scatter(const void* sendbuf, void* recvbuf, const
23     int recvcounts[], MPI_Datatype datatype, MPI_Op op,
24     MPI_Comm comm)
25
26 int MPI_Scan(const void* sendbuf, void* recvbuf, int count,
27     MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
28
29 int MPI_Scatter(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
30     void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
31     MPI_Comm comm)
32
33 int MPI_Scatterv(const void* sendbuf, const int sendcounts[], const
34     int displs[], MPI_Datatype sendtype, void* recvbuf,
35     int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

```

A.2.4 Groups, Contexts, Communicators, and Caching C Bindings

```

34 int MPI_Comm_compare(MPI_Comm comm1, MPI_Comm comm2, int *result)
35
36 int MPI_Comm_create_group(MPI_Comm comm, MPI_Group group, int tag,
37     MPI_Comm *newcomm)
38
39 int MPI_Comm_create_keyval(MPI_Comm_copy_attr_function *comm_copy_attr_fn,
40     MPI_Comm_delete_attr_function *comm_delete_attr_fn,
41     int *comm_keyval, void *extra_state)
42
43 int MPI_Comm_create(MPI_Comm comm, MPI_Group group, MPI_Comm *newcomm)
44
45 int MPI_Comm_delete_attr(MPI_Comm comm, int comm_keyval)
46
47 int MPI_COMM_DUP_FN(MPI_Comm oldcomm, int comm_keyval, void *extra_state,
48     void *attribute_val_in, void *attribute_val_out, int *flag)
49
50 int MPI_Comm_dup(MPI_Comm comm, MPI_Comm *newcomm)

```

```

int MPI_Comm_dup_with_info(MPI_Comm comm, MPI_Info info, MPI_Comm *newcomm)
int MPI_Comm_free_keyval(int *comm_keyval)
int MPI_Comm_free(MPI_Comm *comm)
int MPI_Comm_get_attr(MPI_Comm comm, int comm_keyval, void *attribute_val,
                      int *flag)
int MPI_Comm_get_info(MPI_Comm comm, MPI_Info *info_used)
int MPI_Comm_get_name(MPI_Comm comm, char *comm_name, int *resultlen)
int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)
int MPI_Comm_idup(MPI_Comm comm, MPI_Comm *newcomm, MPI_Request *request)
int MPI_COMM_NULL_COPY_FN(MPI_Comm oldcomm, int comm_keyval,
                          void *extra_state, void *attribute_val_in,
                          void *attribute_val_out, int *flag)
int MPI_COMM_NULL_DELETE_FN(MPI_Comm comm, int comm_keyval, void
                             *attribute_val, void *extra_state)
int MPI_Comm_rank(MPI_Comm comm, int *rank)
int MPI_Comm_remote_group(MPI_Comm comm, MPI_Group *group)
int MPI_Comm_remote_size(MPI_Comm comm, int *size)
int MPI_Comm_set_attr(MPI_Comm comm, int comm_keyval, void *attribute_val)
int MPI_Comm_set_info(MPI_Comm comm, MPI_Info info)
int MPI_Comm_set_name(MPI_Comm comm, const char *comm_name)
int MPI_Comm_size(MPI_Comm comm, int *size)
int MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)
int MPI_Comm_split_type(MPI_Comm comm, int split_type, int key,
                        MPI_Info info, MPI_Comm *newcomm)
int MPI_Comm_test_inter(MPI_Comm comm, int *flag)
int MPI_Group_compare(MPI_Group group1, MPI_Group group2, int *result)
int MPI_Group_difference(MPI_Group group1, MPI_Group group2,
                        MPI_Group *newgroup)
int MPI_Group_excl(MPI_Group group, int n, const int ranks[],
                  MPI_Group *newgroup)
int MPI_Group_free(MPI_Group *group)
int MPI_Group_incl(MPI_Group group, int n, const int ranks[],
                  MPI_Group *newgroup)

```

```

1  int MPI_Group_intersection(MPI_Group group1, MPI_Group group2,
2      MPI_Group *newgroup)
3
4  int MPI_Group_range_excl(MPI_Group group, int n, int ranges[][3],
5      MPI_Group *newgroup)
6
7  int MPI_Group_range_incl(MPI_Group group, int n, int ranges[][3],
8      MPI_Group *newgroup)
9
10 int MPI_Group_rank(MPI_Group group, int *rank)
11
12 int MPI_Group_size(MPI_Group group, int *size)
13
14 int MPI_Group_translate_ranks(MPI_Group group1, int n, const int ranks1[],
15     MPI_Group group2, int ranks2[])
16
17 int MPI_Group_union(MPI_Group group1, MPI_Group group2,
18     MPI_Group *newgroup)
19
20 int MPI_Intercomm_create(MPI_Comm local_comm, int local_leader,
21     MPI_Comm peer_comm, int remote_leader, int tag,
22     MPI_Comm *newintercomm)
23
24 int MPI_Intercomm_merge(MPI_Comm intercomm, int high,
25     MPI_Comm *newintracomm)
26
27 int MPI_Type_create_keyval(MPI_Type_copy_attr_function *type_copy_attr_fn,
28     MPI_Type_delete_attr_function *type_delete_attr_fn,
29     int *type_keyval, void *extra_state)
30
31 int MPI_Type_delete_attr(MPI_Datatype datatype, int type_keyval)
32
33 int MPI_TYPE_DUP_FN(MPI_Datatype oldtype, int type_keyval,
34     void *extra_state, void *attribute_val_in,
35     void *attribute_val_out, int *flag)
36
37 int MPI_Type_free_keyval(int *type_keyval)
38
39 int MPI_Type_get_attr(MPI_Datatype datatype, int type_keyval, void
40     *attribute_val, int *flag)
41
42 int MPI_Type_get_name(MPI_Datatype datatype, char *type_name, int
43     *resultlen)
44
45 int MPI_TYPE_NULL_COPY_FN(MPI_Datatype oldtype, int type_keyval,
46     void *extra_state, void *attribute_val_in,
47     void *attribute_val_out, int *flag)
48
49 int MPI_TYPE_NULL_DELETE_FN(MPI_Datatype datatype, int type_keyval, void
50     *attribute_val, void *extra_state)
51
52 int MPI_Type_set_attr(MPI_Datatype datatype, int type_keyval,
53     void *attribute_val)
54
55 int MPI_Type_set_name(MPI_Datatype datatype, const char *type_name)

```



```

int MPI_Win_create_keyval(MPI_Win_copy_attr_function *win_copy_attr_fn,
                          MPI_Win_delete_attr_function *win_delete_attr_fn,
                          int *win_keyval, void *extra_state)
int MPI_Win_delete_attr(MPI_Win win, int win_keyval)
int MPI_WIN_DUP_FN(MPI_Win oldwin, int win_keyval, void *extra_state,
                  void *attribute_val_in, void *attribute_val_out, int *flag)
int MPI_Win_free_keyval(int *win_keyval)
int MPI_Win_get_attr(MPI_Win win, int win_keyval, void *attribute_val,
                    int *flag)
int MPI_Win_get_name(MPI_Win win, char *win_name, int *resultlen)
int MPI_WIN_NULL_COPY_FN(MPI_Win oldwin, int win_keyval, void *extra_state,
                        void *attribute_val_in, void *attribute_val_out, int *flag)
int MPI_WIN_NULL_DELETE_FN(MPI_Win win, int win_keyval, void
                          *attribute_val, void *extra_state)
int MPI_Win_set_attr(MPI_Win win, int win_keyval, void *attribute_val)
int MPI_Win_set_name(MPI_Win win, const char *win_name)

```

A.2.5 Process Topologies C Bindings

```

int MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int coords[])
int MPI_Cart_create(MPI_Comm comm_old, int ndims, const int dims[], const
                  int periods[], int reorder, MPI_Comm *comm_cart)
int MPI_Cartdim_get(MPI_Comm comm, int *ndims)
int MPI_Cart_get(MPI_Comm comm, int maxdims, int dims[], int periods[],
                int coords[])
int MPI_Cart_map(MPI_Comm comm, int ndims, const int dims[], const
                int periods[], int *newrank)
int MPI_Cart_rank(MPI_Comm comm, const int coords[], int *rank)
int MPI_Cart_shift(MPI_Comm comm, int direction, int disp,
                  int *rank_source, int *rank_dest)
int MPI_Cart_sub(MPI_Comm comm, const int remain_dims[], MPI_Comm *newcomm)
int MPI_Dims_create(int nnodes, int ndims, int dims[])
int MPI_Dist_graph_create_adjacent(MPI_Comm comm_old, int indegree, const
                                  int sources[], const int sourceweights[], int outdegree, const
                                  int destinations[], const int destweights[], MPI_Info info,
                                  int reorder, MPI_Comm *comm_dist_graph)
int MPI_Dist_graph_create(MPI_Comm comm_old, int n, const int sources[],

```

```

1         const int degrees[], const int destinations[], const
2         int weights[], MPI_Info info, int reorder,
3         MPI_Comm *comm_dist_graph)
4
5     int MPI_Dist_graph_neighbors_count(MPI_Comm comm, int *indegree,
6         int *outdegree, int *weighted)
7
8     int MPI_Dist_graph_neighbors(MPI_Comm comm, int maxindegree, int sources[],
9         int sourceweights[], int maxoutdegree, int destinations[],
10        int destweights[])
11
12     int MPI_Graph_create(MPI_Comm comm_old, int nnodes, const int index[],
13        const int edges[], int reorder, MPI_Comm *comm_graph)
14
15     int MPI_Graphdims_get(MPI_Comm comm, int *nnodes, int *nedges)
16
17     int MPI_Graph_get(MPI_Comm comm, int maxindex, int maxedges, int index[],
18        int edges[])
19
20     int MPI_Graph_map(MPI_Comm comm, int nnodes, const int index[], const
21        int edges[], int *newrank)
22
23     int MPI_Graph_neighbors_count(MPI_Comm comm, int rank, int *nneighbors)
24
25     int MPI_Graph_neighbors(MPI_Comm comm, int rank, int maxneighbors,
26        int neighbors[])
27
28     int MPI_Ineighbor_allgather(const void* sendbuf, int sendcount,
29        MPI_Datatype sendtype, void* recvbuf, int recvcount,
30        MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)
31
32     int MPI_Ineighbor_allgatherv(const void* sendbuf, int sendcount,
33        MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
34        const int displs[], MPI_Datatype recvtype, MPI_Comm comm,
35        MPI_Request *request)
36
37     int MPI_Ineighbor_alltoall(const void* sendbuf, int sendcount, MPI_Datatype
38        sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype,
39        MPI_Comm comm, MPI_Request *request)
40
41     int MPI_Ineighbor_alltoallv(const void* sendbuf, const int sendcounts[],
42        const int sdispls[], MPI_Datatype sendtype, void* recvbuf,
43        const int recvcounts[], const int rdispls[], MPI_Datatype
44        recvtype, MPI_Comm comm, MPI_Request *request)
45
46     int MPI_Ineighbor_alltoallw(const void* sendbuf, const int sendcounts[],
47        const MPI_Aint sdispls[], const MPI_Datatype sendtypes[],
48        void* recvbuf, const int recvcounts[], const MPI_Aint
49        rdispls[], const MPI_Datatype recvtypes[], MPI_Comm comm,
50        MPI_Request *request)
51
52     int MPI_Neighbor_allgather(const void* sendbuf, int sendcount, MPI_Datatype
53        sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype,
54        MPI_Comm comm)

```

```

int MPI_Neighbor_allgatherv(const void* sendbuf, int sendcount,
    MPI_Datatype sendtype, void* recvbuf, const int recvcnts[],
    const int displs[], MPI_Datatype recvtpe, MPI_Comm comm)
int MPI_Neighbor_alltoall(const void* sendbuf, int sendcount, MPI_Datatype
    sendtype, void* recvbuf, int recvcnt, MPI_Datatype recvtpe,
    MPI_Comm comm)
int MPI_Neighbor_alltoallv(const void* sendbuf, const int sendcounts[],
    const int sdispls[], MPI_Datatype sendtype, void* recvbuf,
    const int recvcnts[], const int rdispls[], MPI_Datatype
    recvtpe, MPI_Comm comm)
int MPI_Neighbor_alltoallw(const void* sendbuf, const int sendcounts[],
    const MPI_Aint sdispls[], const MPI_Datatype sendtypes[],
    void* recvbuf, const int recvcnts[], const MPI_Aint
    rdispls[], const MPI_Datatype recvtypes[], MPI_Comm comm)
int MPI_Topo_test(MPI_Comm comm, int *status)

```

A.2.6 MPI Environmental Management C Bindings

```

int MPI_Abort(MPI_Comm comm, int errorcode)
int MPI_Add_error_class(int *errorclass)
int MPI_Add_error_code(int errorclass, int *errorcode)
int MPI_Add_error_string(int errorcode, const char *string)
int MPI_Alloc_mem(MPI_Aint size, MPI_Info info, void *baseptr)
int MPI_Comm_call_errhandler(MPI_Comm comm, int errorcode)
int MPI_Comm_create_errhandler(MPI_Comm_errhandler_function
    *comm_errhandler_fn, MPI_Errhandler *errhandler)
int MPI_Comm_get_errhandler(MPI_Comm comm, MPI_Errhandler *errhandler)
int MPI_Comm_set_errhandler(MPI_Comm comm, MPI_Errhandler errhandler)
int MPI_Errhandler_free(MPI_Errhandler *errhandler)
int MPI_Error_class(int errorcode, int *errorclass)
int MPI_Error_string(int errorcode, char *string, int *resultlen)
int MPI_File_call_errhandler(MPI_File fh, int errorcode)
int MPI_File_create_errhandler(MPI_File_errhandler_function
    *file_errhandler_fn, MPI_Errhandler *errhandler)
int MPI_File_get_errhandler(MPI_File file, MPI_Errhandler *errhandler)
int MPI_File_set_errhandler(MPI_File file, MPI_Errhandler errhandler)
int MPI_Finalized(int *flag)

```

A.2.7 The Info Object C Bindings

A.2.8 Process Creation and Management C Bindings

```
int MPI_Close_port(const char *port_name)

int MPI_Comm_accept(const char *port_name, MPI_Info info, int root,
                    MPI_Comm comm, MPI_Comm *newcomm)

int MPI_Comm_connect(const char *port_name, MPI_Info info, int root,
                    MPI_Comm comm, MPI_Comm *newcomm)
```

```

int MPI_Comm_disconnect(MPI_Comm *comm)
int MPI_Comm_get_parent(MPI_Comm *parent)
int MPI_Comm_join(int fd, MPI_Comm *intercomm)
int MPI_Comm_spawn(const char *command, char *argv[], int maxprocs,
                   MPI_Info info, int root, MPI_Comm comm, MPI_Comm *intercomm,
                   int array_of_errcodes[])
int MPI_Comm_spawn_multiple(int count, char *array_of_commands[],
                           char **array_of_argv[], const int array_of_maxprocs[], const
                           MPI_Info array_of_info[], int root, MPI_Comm comm,
                           MPI_Comm *intercomm, int array_of_errcodes[])
int MPI_Lookup_name(const char *service_name, MPI_Info info,
                   char *port_name)
int MPI_Open_port(MPI_Info info, char *port_name)
int MPI_Publish_name(const char *service_name, MPI_Info info, const
                    char *port_name)
int MPI_Unpublish_name(const char *service_name, MPI_Info info, const
                      char *port_name)

```

A.2.9 One-Sided Communications C Bindings

```

int MPI_Accumulate(const void *origin_addr, int origin_count,
                  MPI_Datatype origin_datatype, int target_rank,
                  MPI_Aint target_disp, int target_count,
                  MPI_Datatype target_datatype, MPI_Op op, MPI_Win win)
int MPI_Compare_and_swap(const void *origin_addr, const void *compare_addr,
                        void *result_addr, MPI_Datatype datatype, int target_rank,
                        MPI_Aint target_disp, MPI_Win win)
int MPI_Fetch_and_op(const void *origin_addr, void *result_addr,
                    MPI_Datatype datatype, int target_rank, MPI_Aint target_disp,
                    MPI_Op op, MPI_Win win)
int MPI_Get_accumulate(const void *origin_addr, int origin_count,
                      MPI_Datatype origin_datatype, void *result_addr,
                      int result_count, MPI_Datatype result_datatype,
                      int target_rank, MPI_Aint target_disp, int target_count,
                      MPI_Datatype target_datatype, MPI_Op op, MPI_Win win)
int MPI_Get(void *origin_addr, int origin_count,
            MPI_Datatype origin_datatype, int target_rank,
            MPI_Aint target_disp, int target_count,
            MPI_Datatype target_datatype, MPI_Win win)
int MPI_Put(const void *origin_addr, int origin_count, MPI_Datatype

```

```

1      origin_datatype, int target_rank, MPI_Aint target_disp, int
2      target_count, MPI_Datatype target_datatype, MPI_Win win)
3
4  int MPI_Raccumulate(const void *origin_addr, int origin_count,
5      MPI_Datatype origin_datatype, int target_rank,
6      MPI_Aint target_disp, int target_count,
7      MPI_Datatype target_datatype, MPI_Op op, MPI_Win win,
8      MPI_Request *request)
9
10 int MPI_Rget_accumulate(const void *origin_addr, int origin_count,
11     MPI_Datatype origin_datatype, void *result_addr,
12     int result_count, MPI_Datatype result_datatype,
13     int target_rank, MPI_Aint target_disp, int target_count,
14     MPI_Datatype target_datatype, MPI_Op op, MPI_Win win,
15     MPI_Request *request)
16
17 int MPI_Rget(void *origin_addr, int origin_count,
18     MPI_Datatype origin_datatype, int target_rank,
19     MPI_Aint target_disp, int target_count,
20     MPI_Datatype target_datatype, MPI_Win win,
21     MPI_Request *request)
22
23 int MPI_Rput(const void *origin_addr, int origin_count,
24     MPI_Datatype origin_datatype, int target_rank,
25     MPI_Aint target_disp, int target_count,
26     MPI_Datatype target_datatype, MPI_Win win,
27     MPI_Request *request)
28
29 int MPI_Win_allocate(MPI_Aint size, int disp_unit, MPI_Info info,
30     MPI_Comm comm, void *baseptr, MPI_Win *win)
31
32 int MPI_Win_allocate_shared(MPI_Aint size, int disp_unit, MPI_Info info,
33     MPI_Comm comm, void *baseptr, MPI_Win *win)
34
35 int MPI_Win_attach(MPI_Win win, void *base, MPI_Aint size)
36
37 int MPI_Win_complete(MPI_Win win)
38
39 int MPI_Win_create_dynamic(MPI_Info info, MPI_Comm comm, MPI_Win *win)
40
41 int MPI_Win_create(void *base, MPI_Aint size, int disp_unit, MPI_Info info,
42     MPI_Comm comm, MPI_Win *win)
43
44 int MPI_Win_detach(MPI_Win win, const void *base)
45
46 int MPI_Win_fence(int assert, MPI_Win win)
47
48 int MPI_Win_flush_all(MPI_Win win)
49
50 int MPI_Win_flush(int rank, MPI_Win win)
51
52 int MPI_Win_flush_local_all(MPI_Win win)
53
54 int MPI_Win_flush_local(int rank, MPI_Win win)

```

```

int MPI_Win_free(MPI_Win *win)
int MPI_Win_get_group(MPI_Win win, MPI_Group *group)
int MPI_Win_get_info(MPI_Win win, MPI_Info *info_used)
int MPI_Win_lock_all(int assert, MPI_Win win)
int MPI_Win_lock(int lock_type, int rank, int assert, MPI_Win win)
int MPI_Win_post(MPI_Group group, int assert, MPI_Win win)
int MPI_Win_set_info(MPI_Win win, MPI_Info info)
int MPI_Win_shared_query(MPI_Win win, int rank, MPI_Aint *size,
    int *disp_unit, void *baseptr)
int MPI_Win_start(MPI_Group group, int assert, MPI_Win win)
int MPI_Win_sync(MPI_Win win)
int MPI_Win_test(MPI_Win win, int *flag)
int MPI_Win_unlock_all(MPI_Win win)
int MPI_Win_unlock(int rank, MPI_Win win)
int MPI_Win_wait(MPI_Win win)

```

A.2.10 External Interfaces C Bindings

```

int MPI_Grequest_complete(MPI_Request request)
int MPI_Grequest_start(MPI_Grequest_query_function *query_fn,
    MPI_Grequest_free_function *free_fn,
    MPI_Grequest_cancel_function *cancel_fn, void *extra_state,
    MPI_Request *request)
int MPI_Init_thread(int *argc, char ***argv, int required, int *provided)
int MPI_Is_thread_main(int *flag)
int MPI_Query_thread(int *provided)
int MPI_Status_set_cancelled(MPI_Status *status, int flag)
int MPI_Status_set_elements(MPI_Status *status, MPI_Datatype datatype,
    int count)
int MPI_Status_set_elements_x(MPI_Status *status, MPI_Datatype datatype,
    MPI_Count count)

```

A.2.11 I/O C Bindings

```

int MPI_CONVERSION_FN_NULL(void *userbuf, MPI_Datatype datatype, int count,
    void *filebuf, MPI_Offset position, void *extra_state)

```

```
1  int MPI_File_close(MPI_File *fh)
2
3  int MPI_File_delete(const char *filename, MPI_Info info)
4
5  int MPI_File_get_amode(MPI_File fh, int *amode)
6
7  int MPI_File_get_atomicity(MPI_File fh, int *flag)
8
9  int MPI_File_get_byte_offset(MPI_File fh, MPI_Offset offset,
10                               MPI_Offset *disp)
11
12 int MPI_File_get_group(MPI_File fh, MPI_Group *group)
13
14 int MPI_File_get_info(MPI_File fh, MPI_Info *info_used)
15
16 int MPI_File_get_position(MPI_File fh, MPI_Offset *offset)
17
18 int MPI_File_get_position_shared(MPI_File fh, MPI_Offset *offset)
19
20 int MPI_File_get_size(MPI_File fh, MPI_Offset *size)
21
22 int MPI_File_get_type_extent(MPI_File fh, MPI_Datatype datatype,
23                               MPI_Aint *extent)
24
25 int MPI_File_get_view(MPI_File fh, MPI_Offset *disp, MPI_Datatype *etype,
26                       MPI_Datatype *filetype, char *datarep)
27
28 int MPI_File_iread_all(MPI_File fh, void *buf, int count,
29                       MPI_Datatype datatype, MPI_Request *request)
30
31 int MPI_File_iread_at_all(MPI_File fh, MPI_Offset offset, void *buf,
32                           int count, MPI_Datatype datatype, MPI_Request *request)
33
34 int MPI_File_iread_at(MPI_File fh, MPI_Offset offset, void *buf, int count,
35                       MPI_Datatype datatype, MPI_Request *request)
36
37 int MPI_File_iread(MPI_File fh, void *buf, int count,
38                   MPI_Datatype datatype, MPI_Request *request)
39
40 int MPI_File_iread_shared(MPI_File fh, void *buf, int count,
41                           MPI_Datatype datatype, MPI_Request *request)
42
43 int MPI_File_iwrite_all(MPI_File fh, const void *buf, int count,
44                         MPI_Datatype datatype, MPI_Request *request)
45
46 int MPI_File_iwrite_at_all(MPI_File fh, MPI_Offset offset, const void *buf,
47                             int count, MPI_Datatype datatype, MPI_Request *request)
48
49 int MPI_File_iwrite_at(MPI_File fh, MPI_Offset offset, const void *buf,
50                         int count, MPI_Datatype datatype, MPI_Request *request)
51
52 int MPI_File_iwrite(MPI_File fh, const void *buf, int count,
53                    MPI_Datatype datatype, MPI_Request *request)
54
55 int MPI_File_iwrite_shared(MPI_File fh, const void *buf, int count,
56                            MPI_Datatype datatype, MPI_Request *request)
```



```

int MPI_File_open(MPI_Comm comm, const char *filename, int amode,
                  MPI_Info info, MPI_File *fh)
int MPI_File_preallocate(MPI_File fh, MPI_Offset size)
int MPI_File_read_all_begin(MPI_File fh, void *buf, int count,
                             MPI_Datatype datatype)
int MPI_File_read_all_end(MPI_File fh, void *buf, MPI_Status *status)
int MPI_File_read_all(MPI_File fh, void *buf, int count,
                      MPI_Datatype datatype, MPI_Status *status)
int MPI_File_read_at_all_begin(MPI_File fh, MPI_Offset offset, void *buf,
                               int count, MPI_Datatype datatype)
int MPI_File_read_at_all_end(MPI_File fh, void *buf, MPI_Status *status)
int MPI_File_read_at_all(MPI_File fh, MPI_Offset offset, void *buf,
                          int count, MPI_Datatype datatype, MPI_Status *status)
int MPI_File_read_at(MPI_File fh, MPI_Offset offset, void *buf, int count,
                     MPI_Datatype datatype, MPI_Status *status)
int MPI_File_read(MPI_File fh, void *buf, int count, MPI_Datatype datatype,
                  MPI_Status *status)
int MPI_File_read_ordered_begin(MPI_File fh, void *buf, int count,
                                MPI_Datatype datatype)
int MPI_File_read_ordered_end(MPI_File fh, void *buf, MPI_Status *status)
int MPI_File_read_ordered(MPI_File fh, void *buf, int count,
                           MPI_Datatype datatype, MPI_Status *status)
int MPI_File_read_shared(MPI_File fh, void *buf, int count,
                          MPI_Datatype datatype, MPI_Status *status)
int MPI_File_seek(MPI_File fh, MPI_Offset offset, int whence)
int MPI_File_seek_shared(MPI_File fh, MPI_Offset offset, int whence)
int MPI_File_set_atomicity(MPI_File fh, int flag)
int MPI_File_set_info(MPI_File fh, MPI_Info info)
int MPI_File_set_size(MPI_File fh, MPI_Offset size)
int MPI_File_set_view(MPI_File fh, MPI_Offset disp, MPI_Datatype etype,
                      MPI_Datatype filetype, const char *datarep, MPI_Info info)
int MPI_File_sync(MPI_File fh)
int MPI_File_write_all_begin(MPI_File fh, const void *buf, int count,
                              MPI_Datatype datatype)
int MPI_File_write_all_end(MPI_File fh, const void *buf,
                             MPI_Status *status)

```

```

1  int MPI_File_write_all(MPI_File fh, const void *buf, int count,
2      MPI_Datatype datatype, MPI_Status *status)
3
4  int MPI_File_write_at_all_begin(MPI_File fh, MPI_Offset offset, const
5      void *buf, int count, MPI_Datatype datatype)
6
7  int MPI_File_write_at_all_end(MPI_File fh, const void *buf,
8      MPI_Status *status)
9
10 int MPI_File_write_at_all(MPI_File fh, MPI_Offset offset, const void *buf,
11     int count, MPI_Datatype datatype, MPI_Status *status)
12
13 int MPI_File_write_at(MPI_File fh, MPI_Offset offset, const void *buf,
14     int count, MPI_Datatype datatype, MPI_Status *status)
15
16 int MPI_File_write(MPI_File fh, const void *buf, int count,
17     MPI_Datatype datatype, MPI_Status *status)
18
19 int MPI_File_write_ordered_begin(MPI_File fh, const void *buf, int count,
20     MPI_Datatype datatype)
21
22 int MPI_File_write_ordered_end(MPI_File fh, const void *buf,
23     MPI_Status *status)
24
25 int MPI_File_write_ordered(MPI_File fh, const void *buf, int count,
26     MPI_Datatype datatype, MPI_Status *status)
27
28 int MPI_File_write_shared(MPI_File fh, const void *buf, int count,
29     MPI_Datatype datatype, MPI_Status *status)
30
31 int MPI_Register_datarep(const char *datarep,
32     MPI_Datarep_conversion_function *read_conversion_fn,
33     MPI_Datarep_conversion_function *write_conversion_fn,
34     MPI_Datarep_extent_function *dtype_file_extent_fn,
35     void *extra_state)
36
37
38
39
40
41
42
43
44
45
46
47
48

```

A.2.12 Language Bindings C Bindings

```

34  int MPI_Status_f082f(MPI_F08_status *f08_status, MPI_Fint *f_status)
35
36  int MPI_Status_f2f08(MPI_Fint *f_status, MPI_F08_status *f08_status)
37
38  int MPI_Type_create_f90_complex(int p, int r, MPI_Datatype *newtype)
39
40  int MPI_Type_create_f90_integer(int r, MPI_Datatype *newtype)
41
42  int MPI_Type_create_f90_real(int p, int r, MPI_Datatype *newtype)
43
44  int MPI_Type_match_size(int typeclass, int size, MPI_Datatype *datatype)
45
46  MPI_Fint MPI_Comm_c2f(MPI_Comm comm)
47
48  MPI_Comm MPI_Comm_f2c(MPI_Fint comm)
49
50  MPI_Fint MPI_Errhandler_c2f(MPI_Errhandler errhandler)
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

MPI_Errhandler MPI_Errhandler_f2c(MPI_Fint errhandler) 1
MPI_Fint MPI_File_c2f(MPI_File file) 2
MPI_File MPI_File_f2c(MPI_Fint file) 3
MPI_Fint MPI_Group_c2f(MPI_Group group) 4
MPI_Group MPI_Group_f2c(MPI_Fint group) 5
MPI_Fint MPI_Info_c2f(MPI_Info info) 6
MPI_Info MPI_Info_f2c(MPI_Fint info) 7
MPI_Fint MPI_Message_c2f(MPI_Message message) 8
MPI_Message MPI_Message_f2c(MPI_Fint message) 9
MPI_Fint MPI_Op_c2f(MPI_Op op) 10
MPI_Op MPI_Op_f2c(MPI_Fint op) 11
MPI_Fint MPI_Request_c2f(MPI_Request request) 12
MPI_Request MPI_Request_f2c(MPI_Fint request) 13
int MPI_Status_c2f08(const MPI_Status *c_status, MPI_F08_status 14
    *f08_status) 15
int MPI_Status_c2f(const MPI_Status *c_status, MPI_Fint *f_status) 16
int MPI_Status_f082c(const MPI_F08_status *f08_status, MPI_Status 17
    *c_status) 18
int MPI_Status_f2c(const MPI_Fint *f_status, MPI_Status *c_status) 19
MPI_Fint MPI_Type_c2f(MPI_Datatype datatype) 20
MPI_Datatype MPI_Type_f2c(MPI_Fint datatype) 21
MPI_Fint MPI_Win_c2f(MPI_Win win) 22
MPI_Win MPI_Win_f2c(MPI_Fint win) 23

```

A.2.13 Tools / Profiling Interface C Bindings

```

int MPI_Pcontrol(const int level, ...) 24

```

A.2.14 Tools / MPI Tool Information Interface C Bindings

```

int MPI_T_category_changed(int *stamp) 25
int MPI_T_category_get_categories(int cat_index, int len, int indices[]) 26
int MPI_T_category_get_cvars(int cat_index, int len, int indices[]) 27
int MPI_T_category_get_index(const char *name, int *cat_index) 28

```

```
1  int MPI_T_category_get_info(int cat_index, char *name, int *name_len,
2      char *desc, int *desc_len, int *num_cvars, int *num_pvars,
3      int *num_categories)
4
5  int MPI_T_category_get_num(int *num_cat)
6
7  int MPI_T_category_get_pvars(int cat_index, int len, int indices[])
8
9  int MPI_T_cvar_get_index(const char *name, int *cvar_index)
10
11 int MPI_T_cvar_get_info(int cvar_index, char *name, int *name_len, int
12     *verbosity, MPI_Datatype *datatype, MPI_T_enum *enumtype, char
13     *desc, int *desc_len, int *bind, int *scope)
14
15 int MPI_T_cvar_get_num(int *num_cvar)
16
17 int MPI_T_cvar_handle_alloc(int cvar_index, void *obj_handle,
18     MPI_T_cvar_handle *handle, int *count)
19
20 int MPI_T_cvar_handle_free(MPI_T_cvar_handle *handle)
21
22 int MPI_T_cvar_read(MPI_T_cvar_handle handle, void* buf)
23
24 int MPI_T_cvar_write(MPI_T_cvar_handle handle, const void* buf)
25
26 int MPI_T_enum_get_info(MPI_T_enum enumtype, int *num, char *name, int
27     *name_len)
28
29 int MPI_T_enum_get_item(MPI_T_enum enumtype, int index, int *value, char
30     *name, int *name_len)
31
32 int MPI_T_finalize(void)
33
34 int MPI_T_init_thread(int required, int *provided)
35
36 int MPI_T_pvar_get_index(const char *name, int var_class, int *pvar_index)
37
38 int MPI_T_pvar_get_info(int pvar_index, char *name, int *name_len,
39     int *verbosity, int *var_class, MPI_Datatype *datatype,
40     MPI_T_enum *enumtype, char *desc, int *desc_len, int *bind,
41     int *readonly, int *continuous, int *atomic)
42
43 int MPI_T_pvar_get_num(int *num_pvar)
44
45 int MPI_T_pvar_handle_alloc(MPI_T_pvar_session session, int pvar_index,
46     void *obj_handle, MPI_T_pvar_handle *handle, int *count)
47
48 int MPI_T_pvar_handle_free(MPI_T_pvar_session session, MPI_T_pvar_handle
49     *handle)
50
51 int MPI_T_pvar_read(MPI_T_pvar_session session, MPI_T_pvar_handle handle,
52     void* buf)
53
54 int MPI_T_pvar_readreset(MPI_T_pvar_session session, MPI_T_pvar_handle
55     handle, void* buf)
56
57 int MPI_T_pvar_reset(MPI_T_pvar_session session, MPI_T_pvar_handle handle)
```

```
int MPI_T_pvar_session_create(MPI_T_pvar_session *session)
int MPI_T_pvar_session_free(MPI_T_pvar_session *session)
int MPI_T_pvar_start(MPI_T_pvar_session session, MPI_T_pvar_handle handle)
int MPI_T_pvar_stop(MPI_T_pvar_session session, MPI_T_pvar_handle handle)
int MPI_T_pvar_write(MPI_T_pvar_session session, MPI_T_pvar_handle handle,
                    const void* buf)
```

A.2.15 Deprecated C Bindings

```
int MPI_Attr_delete(MPI_Comm comm, int keyval)
int MPI_Attr_get(MPI_Comm comm, int keyval, void *attribute_val, int *flag)
int MPI_Attr_put(MPI_Comm comm, int keyval, void* attribute_val)
int MPI_DUP_FN(MPI_Comm oldcomm, int keyval, void *extra_state,
              void *attribute_val_in, void *attribute_val_out, int *flag)
int MPI_Keyval_create(MPI_Copy_function *copy_fn, MPI_Delete_function
                    *delete_fn, int *keyval, void* extra_state)
int MPI_Keyval_free(int *keyval)
int MPI_NULL_COPY_FN(MPI_Comm oldcomm, int keyval, void *extra_state,
                    void *attribute_val_in, void *attribute_val_out, int *flag)
int MPI_NULL_DELETE_FN(MPI_Comm comm, int keyval, void *attribute_val,
                      void *extra_state)
```

A.3 Fortran 2008 Bindings with the mpi_f08 Module

A.3.1 Point-to-Point Communication Fortran 2008 Bindings

```

MPI_Bsend(buf, count, datatype, dest, tag, comm, ierror)
  TYPE(*), DIMENSION(..), INTENT(IN) :: buf
  INTEGER, INTENT(IN) :: count, dest, tag
  TYPE(MPI_Datatype), INTENT(IN) :: datatype
  TYPE(MPI_Comm), INTENT(IN) :: comm
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Bsend_init(buf, count, datatype, dest, tag, comm, request, ierror)
  TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
  INTEGER, INTENT(IN) :: count, dest, tag
  TYPE(MPI_Datatype), INTENT(IN) :: datatype
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Request), INTENT(OUT) :: request
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Buffer_attach(buffer, size, ierror)
  TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buffer
  INTEGER, INTENT(IN) :: size
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Buffer_detach(buffer_addr, size, ierror)
  USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
  TYPE(C_PTR), INTENT(OUT) :: buffer_addr
  INTEGER, INTENT(OUT) :: size
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Cancel(request, ierror)
  TYPE(MPI_Request), INTENT(IN) :: request
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Get_count(status, datatype, count, ierror)
  TYPE(MPI_Status), INTENT(IN) :: status
  TYPE(MPI_Datatype), INTENT(IN) :: datatype
  INTEGER, INTENT(OUT) :: count
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Ibsend(buf, count, datatype, dest, tag, comm, request, ierror)
  TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
  INTEGER, INTENT(IN) :: count, dest, tag
  TYPE(MPI_Datatype), INTENT(IN) :: datatype
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Request), INTENT(OUT) :: request
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Improbe(source, tag, comm, flag, message, status, ierror)
  INTEGER, INTENT(IN) :: source, tag
  TYPE(MPI_Comm), INTENT(IN) :: comm

```

```

LOGICAL, INTENT(OUT) :: flag
TYPE(MPI_Message), INTENT(OUT) :: message
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Imrecv(buf, count, datatype, message, request, ierror)
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Message), INTENT(INOUT) :: message
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Iprobe(source, tag, comm, flag, status, ierror)
INTEGER, INTENT(IN) :: source, tag
TYPE(MPI_Comm), INTENT(IN) :: comm
LOGICAL, INTENT(OUT) :: flag
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Irecv(buf, count, datatype, source, tag, comm, request, ierror)
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count, source, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Irsend(buf, count, datatype, dest, tag, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count, dest, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Isend(buf, count, datatype, dest, tag, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count, dest, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Issend(buf, count, datatype, dest, tag, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count, dest, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request

```

```

1      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
2
3      MPI_Mprobe(source, tag, comm, message, status, ierror)
4          INTEGER, INTENT(IN) :: source, tag
5          TYPE(MPI_Comm), INTENT(IN) :: comm
6          TYPE(MPI_Message), INTENT(OUT) :: message
7          TYPE(MPI_Status) :: status
8          INTEGER, OPTIONAL, INTENT(OUT) :: ierror
9
10     MPI_Mrecv(buf, count, datatype, message, status, ierror)
11         TYPE(*), DIMENSION(..) :: buf
12         INTEGER, INTENT(IN) :: count
13         TYPE(MPI_Datatype), INTENT(IN) :: datatype
14         TYPE(MPI_Message), INTENT(INOUT) :: message
15         TYPE(MPI_Status) :: status
16         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
17
18     MPI_Probe(source, tag, comm, status, ierror)
19         INTEGER, INTENT(IN) :: source, tag
20         TYPE(MPI_Comm), INTENT(IN) :: comm
21         TYPE(MPI_Status) :: status
22         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
23
24     MPI_Recv(buf, count, datatype, source, tag, comm, status, ierror)
25         TYPE(*), DIMENSION(..) :: buf
26         INTEGER, INTENT(IN) :: count, source, tag
27         TYPE(MPI_Datatype), INTENT(IN) :: datatype
28         TYPE(MPI_Comm), INTENT(IN) :: comm
29         TYPE(MPI_Status) :: status
30         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
31
32     MPI_Recv_init(buf, count, datatype, source, tag, comm, request, ierror)
33         TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
34         INTEGER, INTENT(IN) :: count, source, tag
35         TYPE(MPI_Datatype), INTENT(IN) :: datatype
36         TYPE(MPI_Comm), INTENT(IN) :: comm
37         TYPE(MPI_Request), INTENT(OUT) :: request
38         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
39
40     MPI_Request_free(request, ierror)
41         TYPE(MPI_Request), INTENT(INOUT) :: request
42         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
43
44     MPI_Request_get_status(request, flag, status, ierror)
45         TYPE(MPI_Request), INTENT(IN) :: request
46         LOGICAL, INTENT(OUT) :: flag
47         TYPE(MPI_Status) :: status
48         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
49
50     MPI_Rsend(buf, count, datatype, dest, tag, comm, ierror)
51         TYPE(*), DIMENSION(..), INTENT(IN) :: buf

```



```

    INTEGER, INTENT(IN) :: count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) :: datatype
    TYPE(MPI_Comm), INTENT(IN) :: comm
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Rsend_init(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
    INTEGER, INTENT(IN) :: count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) :: datatype
    TYPE(MPI_Comm), INTENT(IN) :: comm
    TYPE(MPI_Request), INTENT(OUT) :: request
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Send(buf, count, datatype, dest, tag, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) :: buf
    INTEGER, INTENT(IN) :: count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) :: datatype
    TYPE(MPI_Comm), INTENT(IN) :: comm
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Send_init(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
    INTEGER, INTENT(IN) :: count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) :: datatype
    TYPE(MPI_Comm), INTENT(IN) :: comm
    TYPE(MPI_Request), INTENT(OUT) :: request
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Sendrecv_replace(buf, count, datatype, dest, sendtag, source, recvtag,
    comm, status, ierror)
    TYPE(*), DIMENSION(..) :: buf
    INTEGER, INTENT(IN) :: count, dest, sendtag, source, recvtag
    TYPE(MPI_Datatype), INTENT(IN) :: datatype
    TYPE(MPI_Comm), INTENT(IN) :: comm
    TYPE(MPI_Status) :: status
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Sendrecv(sendbuf, sendcount, sendtype, dest, sendtag, recvbuf,
    recvcount, recvtype, source, recvtag, comm, status, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
    TYPE(*), DIMENSION(..) :: recvbuf
    INTEGER, INTENT(IN) :: sendcount, dest, sendtag, recvcount, source,
    recvtag
    TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) :: comm
    TYPE(MPI_Status) :: status
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Ssend(buf, count, datatype, dest, tag, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) :: buf

```

```

1      INTEGER, INTENT(IN) :: count, dest, tag
2      TYPE(MPI_Datatype), INTENT(IN) :: datatype
3      TYPE(MPI_Comm), INTENT(IN) :: comm
4      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
5
6  MPI_Ssend_init(buf, count, datatype, dest, tag, comm, request, ierror)
7      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
8      INTEGER, INTENT(IN) :: count, dest, tag
9      TYPE(MPI_Datatype), INTENT(IN) :: datatype
10     TYPE(MPI_Comm), INTENT(IN) :: comm
11     TYPE(MPI_Request), INTENT(OUT) :: request
12     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
13
14 MPI_Startall(count, array_of_requests, ierror)
15     INTEGER, INTENT(IN) :: count
16     TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(count)
17     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
18
19 MPI_Start(request, ierror)
20     TYPE(MPI_Request), INTENT(INOUT) :: request
21     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
22
23 MPI_Testall(count, array_of_requests, flag, array_of_statuses, ierror)
24     INTEGER, INTENT(IN) :: count
25     TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(count)
26     LOGICAL, INTENT(OUT) :: flag
27     TYPE(MPI_Status) :: array_of_statuses(*)
28     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
29
30 MPI_Testany(count, array_of_requests, index, flag, status, ierror)
31     INTEGER, INTENT(IN) :: count
32     TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(count)
33     INTEGER, INTENT(OUT) :: index
34     LOGICAL, INTENT(OUT) :: flag
35     TYPE(MPI_Status) :: status
36     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
37
38 MPI_Test_cancelled(status, flag, ierror)
39     TYPE(MPI_Status), INTENT(IN) :: status
40     LOGICAL, INTENT(OUT) :: flag
41     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
42
43 MPI_Test(request, flag, status, ierror)
44     TYPE(MPI_Request), INTENT(INOUT) :: request
45     LOGICAL, INTENT(OUT) :: flag
46     TYPE(MPI_Status) :: status
47     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
48
49 MPI_Testsome(incount, array_of_requests, outcount, array_of_indices,
50             array_of_statuses, ierror)
51     INTEGER, INTENT(IN) :: incount

```

```

TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(incount)
INTEGER, INTENT(OUT) :: outcount, array_of_indices(*)
TYPE(MPI_Status) :: array_of_statuses(*)
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Waitall(count, array_of_requests, array_of_statuses, ierror)
INTEGER, INTENT(IN) :: count
TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(count)
TYPE(MPI_Status) :: array_of_statuses(*)
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Waitany(count, array_of_requests, index, status, ierror)
INTEGER, INTENT(IN) :: count
TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(count)
INTEGER, INTENT(OUT) :: index
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Wait(request, status, ierror)
TYPE(MPI_Request), INTENT(INOUT) :: request
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Waitsome(incount, array_of_requests, outcount, array_of_indices,
             array_of_statuses, ierror)
INTEGER, INTENT(IN) :: incount
TYPE(MPI_Request), INTENT(INOUT) :: array_of_requests(incount)
INTEGER, INTENT(OUT) :: outcount, array_of_indices(*)
TYPE(MPI_Status) :: array_of_statuses(*)
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

A.3.2 Datatypes Fortran 2008 Bindings

```

INTEGER(KIND=MPI_ADDRESS_KIND) MPI_Aint_add(base, disp)
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: base, disp

INTEGER(KIND=MPI_ADDRESS_KIND) MPI_Aint_diff(addr1, addr2)
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: addr1, addr2

MPI_Get_address(location, address, ierror)
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: location
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: address
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Get_elements(status, datatype, count, ierror)
TYPE(MPI_Status), INTENT(IN) :: status
TYPE(MPI_Datatype), INTENT(IN) :: datatype
INTEGER, INTENT(OUT) :: count
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Get_elements_x(status, datatype, count, ierror)

```

```

1      TYPE(MPI_Status), INTENT(IN) :: status
2      TYPE(MPI_Datatype), INTENT(IN) :: datatype
3      INTEGER(KIND = MPI_COUNT_KIND), INTENT(OUT) :: count
4      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
5
6      MPI_Pack_external(datarep, inbuf, incount, datatype, outbuf, outsize,
7          position, ierror)
8      CHARACTER(LEN=*), INTENT(IN) :: datarep
9      TYPE(*), DIMENSION(..), INTENT(IN) :: inbuf
10     TYPE(*), DIMENSION(..) :: outbuf
11     INTEGER, INTENT(IN) :: incount
12     TYPE(MPI_Datatype), INTENT(IN) :: datatype
13     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: outsize
14     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(INOUT) :: position
15     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
16
17     MPI_Pack_external_size(datarep, incount, datatype, size, ierror)
18     TYPE(MPI_Datatype), INTENT(IN) :: datatype
19     INTEGER, INTENT(IN) :: incount
20     CHARACTER(LEN=*), INTENT(IN) :: datarep
21     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: size
22     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
23
24     MPI_Pack(inbuf, incount, datatype, outbuf, outsize, position, comm, ierror)
25     TYPE(*), DIMENSION(..), INTENT(IN) :: inbuf
26     TYPE(*), DIMENSION(..) :: outbuf
27     INTEGER, INTENT(IN) :: incount, outsize
28     TYPE(MPI_Datatype), INTENT(IN) :: datatype
29     INTEGER, INTENT(INOUT) :: position
30     TYPE(MPI_Comm), INTENT(IN) :: comm
31     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
32
33     MPI_Pack_size(incount, datatype, comm, size, ierror)
34     INTEGER, INTENT(IN) :: incount
35     TYPE(MPI_Datatype), INTENT(IN) :: datatype
36     TYPE(MPI_Comm), INTENT(IN) :: comm
37     INTEGER, INTENT(OUT) :: size
38     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
39
40     MPI_Type_commit(datatype, ierror)
41     TYPE(MPI_Datatype), INTENT(INOUT) :: datatype
42     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
43
44     MPI_Type_contiguous(count, oldtype, newtype, ierror)
45     INTEGER, INTENT(IN) :: count
46     TYPE(MPI_Datatype), INTENT(IN) :: oldtype
47     TYPE(MPI_Datatype), INTENT(OUT) :: newtype
48     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
49
50     MPI_Type_create_darray(size, rank, ndims, array_of_gsizes,
51         array_of_distribs, array_of_dargs, array_of_psize, order,

```

```

        oldtype, newtype, ierror)
INTEGER, INTENT(IN) :: size, rank, ndims, array_of_gsizes(ndims),
array_of_distribs(ndims), array_of_dargs(ndims),
array_of_psize(ndims), order
TYPE(MPI_Datatype), INTENT(IN) :: oldtype
TYPE(MPI_Datatype), INTENT(OUT) :: newtype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Type_create_hindexed_block(count, blocklength, array_of_displacements,
        oldtype, newtype, ierror)
INTEGER, INTENT(IN) :: count, blocklength
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::
array_of_displacements(count)
TYPE(MPI_Datatype), INTENT(IN) :: oldtype
TYPE(MPI_Datatype), INTENT(OUT) :: newtype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Type_create_hindexed(count, array_of_blocklengths,
        array_of_displacements, oldtype, newtype, ierror)
INTEGER, INTENT(IN) :: count, array_of_blocklengths(count)
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::
array_of_displacements(count)
TYPE(MPI_Datatype), INTENT(IN) :: oldtype
TYPE(MPI_Datatype), INTENT(OUT) :: newtype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Type_create_hvector(count, blocklength, stride, oldtype, newtype,
        ierror)
INTEGER, INTENT(IN) :: count, blocklength
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: stride
TYPE(MPI_Datatype), INTENT(IN) :: oldtype
TYPE(MPI_Datatype), INTENT(OUT) :: newtype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Type_create_indexed_block(count, blocklength, array_of_displacements,
        oldtype, newtype, ierror)
INTEGER, INTENT(IN) :: count, blocklength,
array_of_displacements(count)
TYPE(MPI_Datatype), INTENT(IN) :: oldtype
TYPE(MPI_Datatype), INTENT(OUT) :: newtype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Type_create_resized(oldtype, lb, extent, newtype, ierror)
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: lb, extent
TYPE(MPI_Datatype), INTENT(IN) :: oldtype
TYPE(MPI_Datatype), INTENT(OUT) :: newtype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Type_create_struct(count, array_of_blocklengths,
        array_of_displacements, array_of_types, newtype, ierror)

```

```

1      INTEGER, INTENT(IN) :: count, array_of_blocklengths(count)
2      INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::
3      array_of_displacements(count)
4      TYPE(MPI_Datatype), INTENT(IN) :: array_of_types(count)
5      TYPE(MPI_Datatype), INTENT(OUT) :: newtype
6      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
7
8      MPI_Type_create_subarray(ndims, array_of_sizes, array_of_subsizes,
9                              array_of_starts, order, oldtype, newtype, ierror)
10     INTEGER, INTENT(IN) :: ndims, array_of_sizes(ndims),
11     array_of_subsizes(ndims), array_of_starts(ndims), order
12     TYPE(MPI_Datatype), INTENT(IN) :: oldtype
13     TYPE(MPI_Datatype), INTENT(OUT) :: newtype
14     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
15
16     MPI_Type_dup(oldtype, newtype, ierror)
17     TYPE(MPI_Datatype), INTENT(IN) :: oldtype
18     TYPE(MPI_Datatype), INTENT(OUT) :: newtype
19     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
20
21     MPI_Type_free(datatype, ierror)
22     TYPE(MPI_Datatype), INTENT(INOUT) :: datatype
23     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
24
25     MPI_Type_get_contents(datatype, max_integers, max_addresses, max_datatypes,
26                           array_of_integers, array_of_addresses, array_of_datatypes,
27                           ierror)
28     TYPE(MPI_Datatype), INTENT(IN) :: datatype
29     INTEGER, INTENT(IN) :: max_integers, max_addresses, max_datatypes
30     INTEGER, INTENT(OUT) :: array_of_integers(max_integers)
31     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::
32     array_of_addresses(max_addresses)
33     TYPE(MPI_Datatype), INTENT(OUT) :: array_of_datatypes(max_datatypes)
34     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
35
36     MPI_Type_get_envelope(datatype, num_integers, num_addresses, num_datatypes,
37                           combiner, ierror)
38     TYPE(MPI_Datatype), INTENT(IN) :: datatype
39     INTEGER, INTENT(OUT) :: num_integers, num_addresses, num_datatypes,
40     combiner
41     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
42
43     MPI_Type_get_extent(datatype, lb, extent, ierror)
44     TYPE(MPI_Datatype), INTENT(IN) :: datatype
45     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: lb, extent
46     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
47
48     MPI_Type_get_extent_x(datatype, lb, extent, ierror)
49     TYPE(MPI_Datatype), INTENT(IN) :: datatype
50     INTEGER(KIND = MPI_COUNT_KIND), INTENT(OUT) :: lb, extent
51     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

MPI_Type_get_true_extent(datatype, true_lb, true_extent, ierror)      1
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                        2
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: true_lb, true_extent 3
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          4
                                                                    5
MPI_Type_get_true_extent_x(datatype, true_lb, true_extent, ierror)    6
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                        7
    INTEGER(KIND = MPI_COUNT_KIND), INTENT(OUT) :: true_lb, true_extent 8
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          9
                                                                    10
MPI_Type_indexed(count, array_of_blocklengths, array_of_displacements, 11
    oldtype, newtype, ierror)
    INTEGER, INTENT(IN) :: count, array_of_blocklengths(count),      12
    array_of_displacements(count)                                     13
    TYPE(MPI_Datatype), INTENT(IN) :: oldtype                         14
    TYPE(MPI_Datatype), INTENT(OUT) :: newtype                       15
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                         16
                                                                    17
MPI_Type_size(datatype, size, ierror)                                  18
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                       19
    INTEGER, INTENT(OUT) :: size                                     20
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                         21
                                                                    22
MPI_Type_size_x(datatype, size, ierror)                                23
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                       24
    INTEGER(KIND=MPI_COUNT_KIND), INTENT(OUT) :: size                25
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                         26
                                                                    27
MPI_Type_vector(count, blocklength, stride, oldtype, newtype, ierror) 28
    INTEGER, INTENT(IN) :: count, blocklength, stride                29
    TYPE(MPI_Datatype), INTENT(IN) :: oldtype                        30
    TYPE(MPI_Datatype), INTENT(OUT) :: newtype                       31
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                         32
                                                                    33
MPI_Unpack_external(datarep, inbuf, insize, position, outbuf, outcount, 34
    datatype, ierror)
    CHARACTER(LEN=*), INTENT(IN) :: datarep                          35
    TYPE(*), DIMENSION(..), INTENT(IN) :: inbuf                     36
    TYPE(*), DIMENSION(..) :: outbuf                                37
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: insize             38
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(INOUT) :: position        39
    INTEGER, INTENT(IN) :: outcount                                  40
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                       41
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                         42
                                                                    43
MPI_Unpack(inbuf, insize, position, outbuf, outcount, datatype, comm, 44
    ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) :: inbuf                     45
    TYPE(*), DIMENSION(..) :: outbuf                                46
    INTEGER, INTENT(IN) :: insize, outcount                          47
    INTEGER, INTENT(INOUT) :: position                               48

```

```

1      TYPE(MPI_Datatype), INTENT(IN) :: datatype
2      TYPE(MPI_Comm), INTENT(IN) :: comm
3      INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

A.3.3 Collective Communication Fortran 2008 Bindings

```

7      MPI_Allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
8                  comm, ierror)

```

```

9      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
10     TYPE(*), DIMENSION(..) :: recvbuf
11     INTEGER, INTENT(IN) :: sendcount, recvcount
12     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
13     TYPE(MPI_Comm), INTENT(IN) :: comm
14     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

16     MPI_Allgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
17                   recvtype, comm, ierror)

```

```

18     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
19     TYPE(*), DIMENSION(..) :: recvbuf
20     INTEGER, INTENT(IN) :: sendcount, recvcounts(*), displs(*)
21     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
22     TYPE(MPI_Comm), INTENT(IN) :: comm
23     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

24     MPI_Allreduce(sendbuf, recvbuf, count, datatype, op, comm, ierror)

```

```

25     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
26     TYPE(*), DIMENSION(..) :: recvbuf
27     INTEGER, INTENT(IN) :: count
28     TYPE(MPI_Datatype), INTENT(IN) :: datatype
29     TYPE(MPI_Op), INTENT(IN) :: op
30     TYPE(MPI_Comm), INTENT(IN) :: comm
31     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

33     MPI_Alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
34                  comm, ierror)

```

```

35     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
36     TYPE(*), DIMENSION(..) :: recvbuf
37     INTEGER, INTENT(IN) :: sendcount, recvcount
38     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
39     TYPE(MPI_Comm), INTENT(IN) :: comm
40     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

41     MPI_Alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf, recvcounts,
42                  rdispls, recvtype, comm, ierror)

```

```

43     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
44     TYPE(*), DIMENSION(..) :: recvbuf
45     INTEGER, INTENT(IN) :: sendcounts(*), sdispls(*), recvcounts(*),
46     rdispls(*)
47     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype

```



```

TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Alltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf, recvcoun
    rdispls, recvtypes, comm, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
TYPE(*), DIMENSION(..) :: recvbuf
INTEGER, INTENT(IN) :: sendcounts(*), sdispls(*), recvcoun
    rdispls(*)
TYPE(MPI_Datatype), INTENT(IN) :: sendtypes(*)
TYPE(MPI_Datatype), INTENT(IN) :: recvtypes(*)
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Barrier(comm, ierror)
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Bcast(buffer, count, datatype, root, comm, ierror)
TYPE(*), DIMENSION(..) :: buffer
INTEGER, INTENT(IN) :: count, root
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Exscan(sendbuf, recvbuf, count, datatype, op, comm, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
TYPE(*), DIMENSION(..) :: recvbuf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Op), INTENT(IN) :: op
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Gather(sendbuf, sendcount, sendtype, recvbuf, recvcoun
    root, comm, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
TYPE(*), DIMENSION(..) :: recvbuf
INTEGER, INTENT(IN) :: sendcount, recvcoun
    root
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Gatherv(sendbuf, sendcount, sendtype, recvbuf, recvcoun
    recvtype, root, comm, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
TYPE(*), DIMENSION(..) :: recvbuf
INTEGER, INTENT(IN) :: sendcount, recvcoun
    root
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
TYPE(MPI_Comm), INTENT(IN) :: comm

```

```

1      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
2
3      MPI_Iallgather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
4                    comm, request, ierror)
5      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
6      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
7      INTEGER, INTENT(IN) :: sendcount, recvcount
8      TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
9      TYPE(MPI_Comm), INTENT(IN) :: comm
10     TYPE(MPI_Request), INTENT(OUT) :: request
11     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
12
13     MPI_Iallgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
14                   recvtype, comm, request, ierror)
15     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
16     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
17     INTEGER, INTENT(IN) :: sendcount
18     INTEGER, INTENT(IN), ASYNCHRONOUS :: recvcounts(*), displs(*)
19     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
20     TYPE(MPI_Comm), INTENT(IN) :: comm
21     TYPE(MPI_Request), INTENT(OUT) :: request
22     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
23
24     MPI_Iallreduce(sendbuf, recvbuf, count, datatype, op, comm, request,
25                   ierror)
26     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
27     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
28     INTEGER, INTENT(IN) :: count
29     TYPE(MPI_Datatype), INTENT(IN) :: datatype
30     TYPE(MPI_Op), INTENT(IN) :: op
31     TYPE(MPI_Comm), INTENT(IN) :: comm
32     TYPE(MPI_Request), INTENT(OUT) :: request
33     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
34
35     MPI_Ialltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
36                  comm, request, ierror)
37     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
38     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
39     INTEGER, INTENT(IN) :: sendcount, recvcount
40     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
41     TYPE(MPI_Comm), INTENT(IN) :: comm
42     TYPE(MPI_Request), INTENT(OUT) :: request
43     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
44
45     MPI_Ialltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf, recvcounts,
46                   rdispls, recvtype, comm, request, ierror)
47     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
48     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
49     INTEGER, INTENT(IN), ASYNCHRONOUS :: sendcounts(*), sdispls(*),
50     recvcounts(*), rdispls(*)

```



```

1  MPI_Igatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
2              recvtype, root, comm, request, ierror)
3      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
4      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
5      INTEGER, INTENT(IN) :: sendcount, root
6      INTEGER, INTENT(IN), ASYNCHRONOUS :: recvcounts(*), displs(*)
7      TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
8      TYPE(MPI_Comm), INTENT(IN) :: comm
9      TYPE(MPI_Request), INTENT(OUT) :: request
10     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
11
12 MPI_Ireduce_scatter_block(sendbuf, recvbuf, recvcount, datatype, op, comm,
13                           request, ierror)
14     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
15     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
16     INTEGER, INTENT(IN) :: recvcount
17     TYPE(MPI_Datatype), INTENT(IN) :: datatype
18     TYPE(MPI_Op), INTENT(IN) :: op
19     TYPE(MPI_Comm), INTENT(IN) :: comm
20     TYPE(MPI_Request), INTENT(OUT) :: request
21     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
22
23 MPI_Ireduce_scatter(sendbuf, recvbuf, recvcounts, datatype, op, comm,
24                     request, ierror)
25     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
26     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
27     INTEGER, INTENT(IN), ASYNCHRONOUS :: recvcounts(*)
28     TYPE(MPI_Datatype), INTENT(IN) :: datatype
29     TYPE(MPI_Op), INTENT(IN) :: op
30     TYPE(MPI_Comm), INTENT(IN) :: comm
31     TYPE(MPI_Request), INTENT(OUT) :: request
32     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
33
34 MPI_Ireduce(sendbuf, recvbuf, count, datatype, op, root, comm, request,
35             ierror)
36     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
37     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
38     INTEGER, INTENT(IN) :: count, root
39     TYPE(MPI_Datatype), INTENT(IN) :: datatype
40     TYPE(MPI_Op), INTENT(IN) :: op
41     TYPE(MPI_Comm), INTENT(IN) :: comm
42     TYPE(MPI_Request), INTENT(OUT) :: request
43     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
44
45 MPI_Iscan(sendbuf, recvbuf, count, datatype, op, comm, request, ierror)
46     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
47     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
48     INTEGER, INTENT(IN) :: count
49     TYPE(MPI_Datatype), INTENT(IN) :: datatype

```

```

TYPE(MPI_Op), INTENT(IN) :: op
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Iscatter(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
             root, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
INTEGER, INTENT(IN) :: sendcount, recvcount, root
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Iscatterv(sendbuf, sendcounts, displs, sendtype, recvbuf, recvcount,
              recvtype, root, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
INTEGER, INTENT(IN), ASYNCHRONOUS :: sendcounts(*), displs(*)
INTEGER, INTENT(IN) :: recvcount, root
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Op_commutative(op, commute, ierror)
TYPE(MPI_Op), INTENT(IN) :: op
LOGICAL, INTENT(OUT) :: commute
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Op_create(user_fn, commute, op, ierror)
PROCEDURE(MPI_User_function) :: user_fn
LOGICAL, INTENT(IN) :: commute
TYPE(MPI_Op), INTENT(OUT) :: op
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Op_free(op, ierror)
TYPE(MPI_Op), INTENT(INOUT) :: op
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Reduce_local(inbuf, inoutbuf, count, datatype, op, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: inbuf
TYPE(*), DIMENSION(..) :: inoutbuf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Op), INTENT(IN) :: op
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Reduce_scatter_block(sendbuf, recvbuf, recvcount, datatype, op, comm,
                         ierror)

```

```

1      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
2      TYPE(*), DIMENSION(..) :: recvbuf
3      INTEGER, INTENT(IN) :: recvcoun
4      TYPE(MPI_Datatype), INTENT(IN) :: datatype
5      TYPE(MPI_Op), INTENT(IN) :: op
6      TYPE(MPI_Comm), INTENT(IN) :: comm
7      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
8
9      MPI_Reduce_scatter(sendbuf, recvbuf, recvcoun, datatype, op, comm,
10         ierror)
11      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
12      TYPE(*), DIMENSION(..) :: recvbuf
13      INTEGER, INTENT(IN) :: recvcoun(*)
14      TYPE(MPI_Datatype), INTENT(IN) :: datatype
15      TYPE(MPI_Op), INTENT(IN) :: op
16      TYPE(MPI_Comm), INTENT(IN) :: comm
17      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
18
19      MPI_Reduce(sendbuf, recvbuf, count, datatype, op, root, comm, ierror)
20      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
21      TYPE(*), DIMENSION(..) :: recvbuf
22      INTEGER, INTENT(IN) :: count, root
23      TYPE(MPI_Datatype), INTENT(IN) :: datatype
24      TYPE(MPI_Op), INTENT(IN) :: op
25      TYPE(MPI_Comm), INTENT(IN) :: comm
26      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
27
28      MPI_Scan(sendbuf, recvbuf, count, datatype, op, comm, ierror)
29      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
30      TYPE(*), DIMENSION(..) :: recvbuf
31      INTEGER, INTENT(IN) :: count
32      TYPE(MPI_Datatype), INTENT(IN) :: datatype
33      TYPE(MPI_Op), INTENT(IN) :: op
34      TYPE(MPI_Comm), INTENT(IN) :: comm
35      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
36
37      MPI_Scatter(sendbuf, sendcount, sendtype, recvbuf, recvcoun, recvtype,
38         root, comm, ierror)
39      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
40      TYPE(*), DIMENSION(..) :: recvbuf
41      INTEGER, INTENT(IN) :: sendcount, recvcoun, root
42      TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
43      TYPE(MPI_Comm), INTENT(IN) :: comm
44      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
45
46      MPI_Scatterv(sendbuf, sendcounts, displs, sendtype, recvbuf, recvcoun,
47         recvtype, root, comm, ierror)
48      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
49      TYPE(*), DIMENSION(..) :: recvbuf
50      INTEGER, INTENT(IN) :: sendcounts(*), displs(*), recvcoun, root

```

```

TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

A.3.4 Groups, Contexts, Communicators, and Caching Fortran 2008 Bindings

```

MPI_Comm_compare(comm1, comm2, result, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm1, comm2
  INTEGER, INTENT(OUT) :: result
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Comm_create(comm, group, newcomm, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Group), INTENT(IN) :: group
  TYPE(MPI_Comm), INTENT(OUT) :: newcomm
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Comm_create_group(comm, group, tag, newcomm, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Group), INTENT(IN) :: group
  INTEGER, INTENT(IN) :: tag
  TYPE(MPI_Comm), INTENT(OUT) :: newcomm
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Comm_create_keyval(comm_copy_attr_fn, comm_delete_attr_fn, comm_keyval,
  extra_state, ierror)
  PROCEDURE(MPI_Comm_copy_attr_function) :: comm_copy_attr_fn
  PROCEDURE(MPI_Comm_delete_attr_function) :: comm_delete_attr_fn
  INTEGER, INTENT(OUT) :: comm_keyval
  INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: extra_state
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Comm_delete_attr(comm, comm_keyval, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  INTEGER, INTENT(IN) :: comm_keyval
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Comm_dup(comm, newcomm, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Comm), INTENT(OUT) :: newcomm
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_COMM_DUP_FN(oldcomm, comm_keyval, extra_state, attribute_val_in,
  attribute_val_out, flag, ierror)
  TYPE(MPI_Comm) :: oldcomm
  INTEGER :: comm_keyval
  INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in
  INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val_out
  LOGICAL :: flag
  INTEGER :: ierror

```

```

1  MPI_Comm_dup_with_info(comm, info, newcomm, ierror)
2      TYPE(MPI_Comm), INTENT(IN) :: comm
3      TYPE(MPI_Info), INTENT(IN) :: info
4      TYPE(MPI_Comm), INTENT(OUT) :: newcomm
5      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
6
7  MPI_Comm_free(comm, ierror)
8      TYPE(MPI_Comm), INTENT(INOUT) :: comm
9      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
10
11 MPI_Comm_free_keyval(comm_keyval, ierror)
12     INTEGER, INTENT(INOUT) :: comm_keyval
13     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
14
15 MPI_Comm_get_attr(comm, comm_keyval, attribute_val, flag, ierror)
16     TYPE(MPI_Comm), INTENT(IN) :: comm
17     INTEGER, INTENT(IN) :: comm_keyval
18     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: attribute_val
19     LOGICAL, INTENT(OUT) :: flag
20     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
21
22 MPI_Comm_get_info(comm, info_used, ierror)
23     TYPE(MPI_Comm), INTENT(IN) :: comm
24     TYPE(MPI_Info), INTENT(OUT) :: info_used
25     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
26
27 MPI_Comm_get_name(comm, comm_name, resultlen, ierror)
28     TYPE(MPI_Comm), INTENT(IN) :: comm
29     CHARACTER(LEN=MPI_MAX_OBJECT_NAME), INTENT(OUT) :: comm_name
30     INTEGER, INTENT(OUT) :: resultlen
31     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
32
33 MPI_Comm_group(comm, group, ierror)
34     TYPE(MPI_Comm), INTENT(IN) :: comm
35     TYPE(MPI_Group), INTENT(OUT) :: group
36     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
37
38 MPI_Comm_idup(comm, newcomm, request, ierror)
39     TYPE(MPI_Comm), INTENT(IN) :: comm
40     TYPE(MPI_Comm), INTENT(OUT), ASYNCHRONOUS :: newcomm
41     TYPE(MPI_Request), INTENT(OUT) :: request
42     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
43
44 MPI_COMM_NULL_COPY_FN(oldcomm, comm_keyval, extra_state, attribute_val_in,
45     attribute_val_out, flag, ierror)
46     TYPE(MPI_Comm) :: oldcomm
47     INTEGER :: comm_keyval
48     INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in
49     INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val_out
50     LOGICAL :: flag
51     INTEGER :: ierror

```



```

MPI_COMM_NULL_DELETE_FN(comm, comm_keyval, attribute_val, extra_state,      1
                          ierror)                                          2
    TYPE(MPI_Comm) :: comm                                              3
    INTEGER :: comm_keyval                                              4
    INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state        5
    INTEGER :: ierror                                                  6
                                                                    7
MPI_Comm_rank(comm, rank, ierror)                                         8
    TYPE(MPI_Comm), INTENT(IN) :: comm                                  9
    INTEGER, INTENT(OUT) :: rank                                       10
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          11
                                                                    12
MPI_Comm_remote_group(comm, group, ierror)                                13
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 14
    TYPE(MPI_Group), INTENT(OUT) :: group                             15
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          16
                                                                    17
MPI_Comm_remote_size(comm, size, ierror)                                  18
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 19
    INTEGER, INTENT(OUT) :: size                                       20
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          21
                                                                    22
MPI_Comm_set_attr(comm, comm_keyval, attribute_val, ierror)              23
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 24
    INTEGER, INTENT(IN) :: comm_keyval                                25
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: attribute_val        26
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          27
                                                                    28
MPI_Comm_set_info(comm, info, ierror)                                     29
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 30
    TYPE(MPI_Info), INTENT(IN) :: info                                 31
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          32
                                                                    33
MPI_Comm_set_name(comm, comm_name, ierror)                               34
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 35
    CHARACTER(LEN=*), INTENT(IN) :: comm_name                         36
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          37
                                                                    38
MPI_Comm_size(comm, size, ierror)                                         39
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 40
    INTEGER, INTENT(OUT) :: size                                       41
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          42
                                                                    43
MPI_Comm_split(comm, color, key, newcomm, ierror)                       44
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 45
    INTEGER, INTENT(IN) :: color, key                                  46
    TYPE(MPI_Comm), INTENT(OUT) :: newcomm                             47
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                          48
                                                                    49
MPI_Comm_split_type(comm, split_type, key, info, newcomm, ierror)        50
    TYPE(MPI_Comm), INTENT(IN) :: comm                                 51
    INTEGER, INTENT(IN) :: split_type, key                            52

```

```

1      TYPE(MPI_Info), INTENT(IN) :: info
2      TYPE(MPI_Comm), INTENT(OUT) :: newcomm
3      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
4
5      MPI_Comm_test_inter(comm, flag, ierror)
6      TYPE(MPI_Comm), INTENT(IN) :: comm
7      LOGICAL, INTENT(OUT) :: flag
8      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
9
10     MPI_Group_compare(group1, group2, result, ierror)
11     TYPE(MPI_Group), INTENT(IN) :: group1, group2
12     INTEGER, INTENT(OUT) :: result
13     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
14
15     MPI_Group_difference(group1, group2, newgroup, ierror)
16     TYPE(MPI_Group), INTENT(IN) :: group1, group2
17     TYPE(MPI_Group), INTENT(OUT) :: newgroup
18     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
19
20     MPI_Group_excl(group, n, ranks, newgroup, ierror)
21     TYPE(MPI_Group), INTENT(IN) :: group
22     INTEGER, INTENT(IN) :: n, ranks(n)
23     TYPE(MPI_Group), INTENT(OUT) :: newgroup
24     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
25
26     MPI_Group_free(group, ierror)
27     TYPE(MPI_Group), INTENT(INOUT) :: group
28     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
29
30     MPI_Group_incl(group, n, ranks, newgroup, ierror)
31     TYPE(MPI_Group), INTENT(IN) :: group
32     INTEGER, INTENT(IN) :: n, ranks(n)
33     TYPE(MPI_Group), INTENT(OUT) :: newgroup
34     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
35
36     MPI_Group_intersection(group1, group2, newgroup, ierror)
37     TYPE(MPI_Group), INTENT(IN) :: group1, group2
38     TYPE(MPI_Group), INTENT(OUT) :: newgroup
39     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
40
41     MPI_Group_range_excl(group, n, ranges, newgroup, ierror)
42     TYPE(MPI_Group), INTENT(IN) :: group
43     INTEGER, INTENT(IN) :: n, ranges(3,n)
44     TYPE(MPI_Group), INTENT(OUT) :: newgroup
45     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
46
47     MPI_Group_range_incl(group, n, ranges, newgroup, ierror)
48     TYPE(MPI_Group), INTENT(IN) :: group
49     INTEGER, INTENT(IN) :: n, ranges(3,n)
50     TYPE(MPI_Group), INTENT(OUT) :: newgroup
51     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

MPI_Group_rank(group, rank, ierror)                                1
    TYPE(MPI_Group), INTENT(IN) :: group                          2
    INTEGER, INTENT(OUT) :: rank                                  3
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      4
                                                                    5
MPI_Group_size(group, size, ierror)                                6
    TYPE(MPI_Group), INTENT(IN) :: group                          7
    INTEGER, INTENT(OUT) :: size                                  8
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      9
                                                                    10
MPI_Group_translate_ranks(group1, n, ranks1, group2, ranks2, ierror) 11
    TYPE(MPI_Group), INTENT(IN) :: group1, group2                12
    INTEGER, INTENT(IN) :: n, ranks1(n)                          13
    INTEGER, INTENT(OUT) :: ranks2(n)                            14
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      15
                                                                    16
MPI_Group_union(group1, group2, newgroup, ierror)                 17
    TYPE(MPI_Group), INTENT(IN) :: group1, group2                18
    TYPE(MPI_Group), INTENT(OUT) :: newgroup                      19
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      20
                                                                    21
MPI_Intercomm_create(local_comm, local_leader, peer_comm, remote_leader, 22
    tag, newintercomm, ierror)                                     23
    TYPE(MPI_Comm), INTENT(IN) :: local_comm, peer_comm           24
    INTEGER, INTENT(IN) :: local_leader, remote_leader, tag       25
    TYPE(MPI_Comm), INTENT(OUT) :: newintercomm                   26
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      27
                                                                    28
MPI_Intercomm_merge(intercomm, high, newintracomm, ierror)        29
    TYPE(MPI_Comm), INTENT(IN) :: intercomm                       30
    LOGICAL, INTENT(IN) :: high                                    31
    TYPE(MPI_Comm), INTENT(OUT) :: newintracomm                   32
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      33
                                                                    34
MPI_Type_create_keyval(type_copy_attr_fn, type_delete_attr_fn, type_keyval, 35
    extra_state, ierror)                                           36
    PROCEDURE(MPI_Type_copy_attr_function) :: type_copy_attr_fn   37
    PROCEDURE(MPI_Type_delete_attr_function) :: type_delete_attr_fn 38
    INTEGER, INTENT(OUT) :: type_keyval                           39
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: extra_state     40
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      41
                                                                    42
MPI_Type_delete_attr(datatype, type_keyval, ierror)               43
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                    44
    INTEGER, INTENT(IN) :: type_keyval                             45
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      46
                                                                    47
MPI_Type_DUP_FN(oldtype, type_keyval, extra_state, attribute_val_in, 48
    attribute_val_out, flag, ierror)
    TYPE(MPI_Datatype) :: oldtype
    INTEGER :: type_keyval

```

```

1      INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in
2      INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val_out
3      LOGICAL :: flag
4      INTEGER :: ierror
5
6      MPI_Type_free_keyval(type_keyval, ierror)
7          INTEGER, INTENT(INOUT) :: type_keyval
8          INTEGER, OPTIONAL, INTENT(OUT) :: ierror
9
10     MPI_Type_get_attr(datatype, type_keyval, attribute_val, flag, ierror)
11         TYPE(MPI_Datatype), INTENT(IN) :: datatype
12         INTEGER, INTENT(IN) :: type_keyval
13         INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: attribute_val
14         LOGICAL, INTENT(OUT) :: flag
15         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
16
17     MPI_Type_get_name(datatype, type_name, resultlen, ierror)
18         TYPE(MPI_Datatype), INTENT(IN) :: datatype
19         CHARACTER(LEN=MPI_MAX_OBJECT_NAME), INTENT(OUT) :: type_name
20         INTEGER, INTENT(OUT) :: resultlen
21         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
22
23     MPI_TYPE_NULL_COPY_FN(oldtype, type_keyval, extra_state, attribute_val_in,
24         attribute_val_out, flag, ierror)
25         TYPE(MPI_Datatype) :: oldtype
26         INTEGER :: type_keyval
27         INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in
28         INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val_out
29         LOGICAL :: flag
30         INTEGER :: ierror
31
32     MPI_TYPE_NULL_DELETE_FN(datatype, type_keyval, attribute_val, extra_state,
33         ierror)
34         TYPE(MPI_Datatype) :: datatype
35         INTEGER :: type_keyval
36         INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state
37         INTEGER, INTENT(OUT) :: ierror
38
39     MPI_Type_set_attr(datatype, type_keyval, attribute_val, ierror)
40         TYPE(MPI_Datatype), INTENT(IN) :: datatype
41         INTEGER, INTENT(IN) :: type_keyval
42         INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: attribute_val
43         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
44
45     MPI_Type_set_name(datatype, type_name, ierror)
46         TYPE(MPI_Datatype), INTENT(IN) :: datatype
47         CHARACTER(LEN=*), INTENT(IN) :: type_name
48         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
49
50     MPI_Win_create_keyval(win_copy_attr_fn, win_delete_attr_fn, win_keyval,
51         extra_state, ierror)

```

```

PROCEDURE(MPI_Win_copy_attr_function) :: win_copy_attr_fn      1
PROCEDURE(MPI_Win_delete_attr_function) :: win_delete_attr_fn  2
INTEGER, INTENT(OUT) :: win_keyval                             3
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: extra_state      4
INTEGER, OPTIONAL, INTENT(OUT) :: ierror                       5
                                                                6
MPI_Win_delete_attr(win, win_keyval, ierror)                   7
  TYPE(MPI_Win), INTENT(IN) :: win                             8
  INTEGER, INTENT(IN) :: win_keyval                             9
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      10
                                                                11
MPI_WIN_DUP_FN(oldwin, win_keyval, extra_state, attribute_val_in,
               attribute_val_out, flag, ierror)                 12
  TYPE(MPI_Win) :: oldwin                                       13
  INTEGER :: win_keyval                                         14
  INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in 15
  INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val_out           16
  LOGICAL :: flag                                              17
  INTEGER :: ierror                                             18
                                                                19
MPI_Win_free_keyval(win_keyval, ierror)                        20
  INTEGER, INTENT(INOUT) :: win_keyval                          21
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      22
                                                                23
MPI_Win_get_attr(win, win_keyval, attribute_val, flag, ierror) 24
  TYPE(MPI_Win), INTENT(IN) :: win                             25
  INTEGER, INTENT(IN) :: win_keyval                             26
  INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: attribute_val 27
  LOGICAL, INTENT(OUT) :: flag                                  28
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      29
                                                                30
MPI_Win_get_name(win, win_name, resultlen, ierror)             31
  TYPE(MPI_Win), INTENT(IN) :: win                             32
  CHARACTER(LEN=MPI_MAX_OBJECT_NAME), INTENT(OUT) :: win_name 33
  INTEGER, INTENT(OUT) :: resultlen                             34
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror                      35
                                                                36
MPI_WIN_NULL_COPY_FN(oldwin, win_keyval, extra_state, attribute_val_in,
                    attribute_val_out, flag, ierror)            37
  TYPE(MPI_Win) :: oldwin                                       38
  INTEGER :: win_keyval                                         39
  INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in 40
  INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val_out           41
  LOGICAL :: flag                                              42
  INTEGER :: ierror                                             43
                                                                44
MPI_WIN_NULL_DELETE_FN(win, win_keyval, attribute_val, extra_state, ierror)
  TYPE(MPI_Win) :: win                                           45
  INTEGER :: win_keyval                                           46
  INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state 47
  INTEGER :: ierror                                             48

```

```

1 MPI_Win_set_attr(win, win_keyval, attribute_val, ierror)
2     TYPE(MPI_Win), INTENT(IN) :: win
3     INTEGER, INTENT(IN) :: win_keyval
4     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: attribute_val
5     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

6 MPI_Win_set_name(win, win_name, ierror)
7     TYPE(MPI_Win), INTENT(IN) :: win
8     CHARACTER(LEN=*), INTENT(IN) :: win_name
9     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

A.3.5 Process Topologies Fortran 2008 Bindings

```

14 MPI_Cart_coords(comm, rank, maxdims, coords, ierror)
15     TYPE(MPI_Comm), INTENT(IN) :: comm
16     INTEGER, INTENT(IN) :: rank, maxdims
17     INTEGER, INTENT(OUT) :: coords(maxdims)
18     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

19 MPI_Cart_create(comm_old, ndims, dims, periods, reorder, comm_cart, ierror)
20     TYPE(MPI_Comm), INTENT(IN) :: comm_old
21     INTEGER, INTENT(IN) :: ndims, dims(ndims)
22     LOGICAL, INTENT(IN) :: periods(ndims), reorder
23     TYPE(MPI_Comm), INTENT(OUT) :: comm_cart
24     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

26 MPI_Cartdim_get(comm, ndims, ierror)
27     TYPE(MPI_Comm), INTENT(IN) :: comm
28     INTEGER, INTENT(OUT) :: ndims
29     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

30 MPI_Cart_get(comm, maxdims, dims, periods, coords, ierror)
31     TYPE(MPI_Comm), INTENT(IN) :: comm
32     INTEGER, INTENT(IN) :: maxdims
33     INTEGER, INTENT(OUT) :: dims(maxdims), coords(maxdims)
34     LOGICAL, INTENT(OUT) :: periods(maxdims)
35     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

37 MPI_Cart_map(comm, ndims, dims, periods, newrank, ierror)
38     TYPE(MPI_Comm), INTENT(IN) :: comm
39     INTEGER, INTENT(IN) :: ndims, dims(ndims)
40     LOGICAL, INTENT(IN) :: periods(ndims)
41     INTEGER, INTENT(OUT) :: newrank
42     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

43 MPI_Cart_rank(comm, coords, rank, ierror)
44     TYPE(MPI_Comm), INTENT(IN) :: comm
45     INTEGER, INTENT(IN) :: coords(*)
46     INTEGER, INTENT(OUT) :: rank
47     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

MPI_Cart_shift(comm, direction, disp, rank_source, rank_dest, ierror)      1
    TYPE(MPI_Comm), INTENT(IN) :: comm                                    2
    INTEGER, INTENT(IN) :: direction, disp                              3
    INTEGER, INTENT(OUT) :: rank_source, rank_dest                      4
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                            5
                                                                           6
MPI_Cart_sub(comm, remain_dims, newcomm, ierror)                          7
    TYPE(MPI_Comm), INTENT(IN) :: comm                                    8
    LOGICAL, INTENT(IN) :: remain_dims(*)                               9
    TYPE(MPI_Comm), INTENT(OUT) :: newcomm                             10
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                            11
                                                                           12
MPI_Dims_create(nnodes, ndims, dims, ierror)                              13
    INTEGER, INTENT(IN) :: nnodes, ndims                                14
    INTEGER, INTENT(INOUT) :: dims(ndims)                              15
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                            16
                                                                           17
MPI_Dist_graph_create_adjacent(comm_old, indegree, sources, sourceweights, 18
    outdegree, destinations, destweights, info, reorder,               19
    comm_dist_graph, ierror)                                            20
    TYPE(MPI_Comm), INTENT(IN) :: comm_old                             21
    INTEGER, INTENT(IN) :: indegree, sources(indegree), outdegree,     22
    destinations(outdegree)                                            23
    INTEGER, INTENT(IN) :: sourceweights(*), destweights(*)           24
    TYPE(MPI_Info), INTENT(IN) :: info                                  25
    LOGICAL, INTENT(IN) :: reorder                                     26
    TYPE(MPI_Comm), INTENT(OUT) :: comm_dist_graph                     27
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                            28
                                                                           29
MPI_Dist_graph_create(comm_old, n, sources, degrees, destinations, weights, 30
    info, reorder, comm_dist_graph, ierror)                             31
    TYPE(MPI_Comm), INTENT(IN) :: comm_old                             32
    INTEGER, INTENT(IN) :: n, sources(n), degrees(n), destinations(*) 33
    INTEGER, INTENT(IN) :: weights(*)                                   34
    TYPE(MPI_Info), INTENT(IN) :: info                                  35
    LOGICAL, INTENT(IN) :: reorder                                     36
    TYPE(MPI_Comm), INTENT(OUT) :: comm_dist_graph                     37
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                            38
                                                                           39
MPI_Dist_graph_neighbors(comm, maxindegree, sources, sourceweights,      40
    maxoutdegree, destinations, destweights, ierror)                   41
    TYPE(MPI_Comm), INTENT(IN) :: comm                                    42
    INTEGER, INTENT(IN) :: maxindegree, maxoutdegree                   43
    INTEGER, INTENT(OUT) :: sources(maxindegree),                      44
    destinations(maxoutdegree)                                         45
    INTEGER :: sourceweights(*), destweights(*)                        46
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                            47
                                                                           48
MPI_Dist_graph_neighbors_count(comm, indegree, outdegree, weighted, ierror)
    TYPE(MPI_Comm), INTENT(IN) :: comm

```

```

1      INTEGER, INTENT(OUT) :: indegree, outdegree
2      LOGICAL, INTENT(OUT) :: weighted
3      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
4
5      MPI_Graph_create(comm_old, nnodes, index, edges, reorder, comm_graph,
6                      ierror)
7      TYPE(MPI_Comm), INTENT(IN) :: comm_old
8      INTEGER, INTENT(IN) :: nnodes, index(nnodes), edges(*)
9      LOGICAL, INTENT(IN) :: reorder
10     TYPE(MPI_Comm), INTENT(OUT) :: comm_graph
11     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
12
13     MPI_Graphdims_get(comm, nnodes, nedges, ierror)
14     TYPE(MPI_Comm), INTENT(IN) :: comm
15     INTEGER, INTENT(OUT) :: nnodes, nedges
16     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
17
18     MPI_Graph_get(comm, maxindex, maxedges, index, edges, ierror)
19     TYPE(MPI_Comm), INTENT(IN) :: comm
20     INTEGER, INTENT(IN) :: maxindex, maxedges
21     INTEGER, INTENT(OUT) :: index(maxindex), edges(maxedges)
22     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
23
24     MPI_Graph_map(comm, nnodes, index, edges, newrank, ierror)
25     TYPE(MPI_Comm), INTENT(IN) :: comm
26     INTEGER, INTENT(IN) :: nnodes, index(nnodes), edges(*)
27     INTEGER, INTENT(OUT) :: newrank
28     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
29
30     MPI_Graph_neighbors(comm, rank, maxneighbors, neighbors, ierror)
31     TYPE(MPI_Comm), INTENT(IN) :: comm
32     INTEGER, INTENT(IN) :: rank, maxneighbors
33     INTEGER, INTENT(OUT) :: neighbors(maxneighbors)
34     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
35
36     MPI_Graph_neighbors_count(comm, rank, nneighbors, ierror)
37     TYPE(MPI_Comm), INTENT(IN) :: comm
38     INTEGER, INTENT(IN) :: rank
39     INTEGER, INTENT(OUT) :: nneighbors
40     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
41
42     MPI_Ineighbor_allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount,
43                             recvtype, comm, request, ierror)
44     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
45     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
46     INTEGER, INTENT(IN) :: sendcount, recvcount
47     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
48     TYPE(MPI_Comm), INTENT(IN) :: comm
49     TYPE(MPI_Request), INTENT(OUT) :: request
50     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```



```

MPI_Ineighbor_allgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts,
    displs, recvttype, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
INTEGER, INTENT(IN) :: sendcount
INTEGER, INTENT(IN), ASYNCHRONOUS :: recvcounts(*), displs(*)
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Ineighbor_alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount,
    recvttype, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
INTEGER, INTENT(IN) :: sendcount, recvcount
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Ineighbor_alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf,
    recvcounts, rdispls, recvttype, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
INTEGER, INTENT(IN), ASYNCHRONOUS :: sendcounts(*), sdispls(*),
    recvcounts(*), rdispls(*)
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Ineighbor_alltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf,
    recvcounts, rdispls, recvtypes, comm, request, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: sendbuf
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: recvbuf
INTEGER, INTENT(IN), ASYNCHRONOUS :: sendcounts(*), recvcounts(*)
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN), ASYNCHRONOUS ::
    sdispls(*), rdispls(*)
TYPE(MPI_Datatype), INTENT(IN), ASYNCHRONOUS :: sendtypes(*),
    recvtypes(*)
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Neighbor_allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount,
    recvttype, comm, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
TYPE(*), DIMENSION(..) :: recvbuf

```

```

1      INTEGER, INTENT(IN) :: sendcount, recvcount
2      TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
3      TYPE(MPI_Comm), INTENT(IN) :: comm
4      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
5
6  MPI_Neighbor_allgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts,
7      displs, recvttype, comm, ierror)
8      TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
9      TYPE(*), DIMENSION(..) :: recvbuf
10     INTEGER, INTENT(IN) :: sendcount, recvcounts(*), displs(*)
11     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
12     TYPE(MPI_Comm), INTENT(IN) :: comm
13     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
14
15 MPI_Neighbor_alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount,
16     recvttype, comm, ierror)
17     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
18     TYPE(*), DIMENSION(..) :: recvbuf
19     INTEGER, INTENT(IN) :: sendcount, recvcount
20     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
21     TYPE(MPI_Comm), INTENT(IN) :: comm
22     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
23
24 MPI_Neighbor_alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf,
25     recvcounts, rdispls, recvttype, comm, ierror)
26     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
27     TYPE(*), DIMENSION(..) :: recvbuf
28     INTEGER, INTENT(IN) :: sendcounts(*), sdispls(*), recvcounts(*),
29     rdispls(*)
30     TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvttype
31     TYPE(MPI_Comm), INTENT(IN) :: comm
32     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
33
34 MPI_Neighbor_alltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf,
35     recvcounts, rdispls, recvtypes, comm, ierror)
36     TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
37     TYPE(*), DIMENSION(..) :: recvbuf
38     INTEGER, INTENT(IN) :: sendcounts(*), recvcounts(*)
39     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: sdispls(*), rdispls(*)
40     TYPE(MPI_Datatype), INTENT(IN) :: sendtypes(*), recvtypes(*)
41     TYPE(MPI_Comm), INTENT(IN) :: comm
42     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
43
44 MPI_Topo_test(comm, status, ierror)
45     TYPE(MPI_Comm), INTENT(IN) :: comm
46     INTEGER, INTENT(OUT) :: status
47     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
48

```

A.3.6 MPI Environmental Management Fortran 2008 Bindings

```

DOUBLE PRECISION MPI_Wtick()
DOUBLE PRECISION MPI_Wtime()
MPI_Abort(comm, errorcode, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  INTEGER, INTENT(IN) :: errorcode
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Add_error_class(errorclass, ierror)
  INTEGER, INTENT(OUT) :: errorclass
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Add_error_code(errorclass, errorcode, ierror)
  INTEGER, INTENT(IN) :: errorclass
  INTEGER, INTENT(OUT) :: errorcode
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Add_error_string(errorcode, string, ierror)
  INTEGER, INTENT(IN) :: errorcode
  CHARACTER(LEN=*), INTENT(IN) :: string
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Alloc_mem(size, info, baseptr, ierror)
  USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
  INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
  TYPE(MPI_Info), INTENT(IN) :: info
  TYPE(C_PTR), INTENT(OUT) :: baseptr
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_call_errhandler(comm, errorcode, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  INTEGER, INTENT(IN) :: errorcode
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_create_errhandler(comm_errhandler_fn, errhandler, ierror)
  PROCEDURE(MPI_Comm_errhandler_function) :: comm_errhandler_fn
  TYPE(MPI_Errhandler), INTENT(OUT) :: errhandler
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_get_errhandler(comm, errhandler, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Errhandler), INTENT(OUT) :: errhandler
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_set_errhandler(comm, errhandler, ierror)
  TYPE(MPI_Comm), INTENT(IN) :: comm
  TYPE(MPI_Errhandler), INTENT(IN) :: errhandler
  INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Errhandler_free(errhandler, ierror)

```

```

1      TYPE(MPI_Errhandler), INTENT(INOUT) :: errhandler
2      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
3
4      MPI_Error_class(errorcode, errorclass, ierror)
5          INTEGER, INTENT(IN) :: errorcode
6          INTEGER, INTENT(OUT) :: errorclass
7          INTEGER, OPTIONAL, INTENT(OUT) :: ierror
8
9      MPI_Error_string(errorcode, string, resultlen, ierror)
10         INTEGER, INTENT(IN) :: errorcode
11         CHARACTER(LEN=MPI_MAX_ERROR_STRING), INTENT(OUT) :: string
12         INTEGER, INTENT(OUT) :: resultlen
13         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
14
15      MPI_File_call_errhandler(fh, errorcode, ierror)
16         TYPE(MPI_File), INTENT(IN) :: fh
17         INTEGER, INTENT(IN) :: errorcode
18         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
19
20      MPI_File_create_errhandler(file_errhandler_fn, errhandler, ierror)
21         PROCEDURE(MPI_File_errhandler_function) :: file_errhandler_fn
22         TYPE(MPI_Errhandler), INTENT(OUT) :: errhandler
23         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
24
25      MPI_File_get_errhandler(file, errhandler, ierror)
26         TYPE(MPI_File), INTENT(IN) :: file
27         TYPE(MPI_Errhandler), INTENT(OUT) :: errhandler
28         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
29
30      MPI_File_set_errhandler(file, errhandler, ierror)
31         TYPE(MPI_File), INTENT(IN) :: file
32         TYPE(MPI_Errhandler), INTENT(IN) :: errhandler
33         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
34
35      MPI_Finalized(flag, ierror)
36         LOGICAL, INTENT(OUT) :: flag
37         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
38
39      MPI_Finalize(ierror)
40         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
41
42      MPI_Free_mem(base, ierror)
43         TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: base
44         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
45
46      MPI_Get_library_version(version, resultlen, ierror)
47         CHARACTER(LEN=MPI_MAX_LIBRARY_VERSION_STRING), INTENT(OUT) :: version
48         INTEGER, INTENT(OUT) :: resultlen
49         INTEGER, OPTIONAL, INTENT(OUT) :: ierror
50
51      MPI_Get_processor_name(name, resultlen, ierror)
52         CHARACTER(LEN=MPI_MAX_PROCESSOR_NAME), INTENT(OUT) :: name
53         INTEGER, INTENT(OUT) :: resultlen

```

```

    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Get_version(version, subversion, ierror)
    INTEGER, INTENT(OUT) :: version, subversion
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Initialized(flag, ierror)
    LOGICAL, INTENT(OUT) :: flag
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Init(ierror)
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_call_errhandler(win, errorcode, ierror)
    TYPE(MPI_Win), INTENT(IN) :: win
    INTEGER, INTENT(IN) :: errorcode
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_create_errhandler(win_errhandler_fn, errhandler, ierror)
    PROCEDURE(MPI_Win_errhandler_function) :: win_errhandler_fn
    TYPE(MPI_Errhandler), INTENT(OUT) :: errhandler
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_get_errhandler(win, errhandler, ierror)
    TYPE(MPI_Win), INTENT(IN) :: win
    TYPE(MPI_Errhandler), INTENT(OUT) :: errhandler
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_set_errhandler(win, errhandler, ierror)
    TYPE(MPI_Win), INTENT(IN) :: win
    TYPE(MPI_Errhandler), INTENT(IN) :: errhandler
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

A.3.7 The Info Object Fortran 2008 Bindings

```

MPI_Info_create(info, ierror)
    TYPE(MPI_Info), INTENT(OUT) :: info
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Info_delete(info, key, ierror)
    TYPE(MPI_Info), INTENT(IN) :: info
    CHARACTER(LEN=*), INTENT(IN) :: key
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Info_dup(info, newinfo, ierror)
    TYPE(MPI_Info), INTENT(IN) :: info
    TYPE(MPI_Info), INTENT(OUT) :: newinfo
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Info_free(info, ierror)
    TYPE(MPI_Info), INTENT(INOUT) :: info
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

1  MPI_Info_get(info, key, valuelen, value, flag, ierror)
2      TYPE(MPI_Info), INTENT(IN) :: info
3      CHARACTER(LEN=*), INTENT(IN) :: key
4      INTEGER, INTENT(IN) :: valuelen
5      CHARACTER(LEN=valuelen), INTENT(OUT) :: value
6      LOGICAL, INTENT(OUT) :: flag
7      INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

8  MPI_Info_get_nkeys(info, nkeys, ierror)
9      TYPE(MPI_Info), INTENT(IN) :: info
10     INTEGER, INTENT(OUT) :: nkeys
11     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

13 MPI_Info_get_nthkey(info, n, key, ierror)
14     TYPE(MPI_Info), INTENT(IN) :: info
15     INTEGER, INTENT(IN) :: n
16     CHARACTER(LEN=*), INTENT(OUT) :: key
17     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

19 MPI_Info_get_valuelen(info, key, valuelen, flag, ierror)
20     TYPE(MPI_Info), INTENT(IN) :: info
21     CHARACTER(LEN=*), INTENT(IN) :: key
22     INTEGER, INTENT(OUT) :: valuelen
23     LOGICAL, INTENT(OUT) :: flag
24     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

25 MPI_Info_set(info, key, value, ierror)
26     TYPE(MPI_Info), INTENT(IN) :: info
27     CHARACTER(LEN=*), INTENT(IN) :: key, value
28     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

31 A.3.8 Process Creation and Management Fortran 2008 Bindings

```

32 MPI_Close_port(port_name, ierror)
33     CHARACTER(LEN=*), INTENT(IN) :: port_name
34     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

36 MPI_Comm_accept(port_name, info, root, comm, newcomm, ierror)
37     CHARACTER(LEN=*), INTENT(IN) :: port_name
38     TYPE(MPI_Info), INTENT(IN) :: info
39     INTEGER, INTENT(IN) :: root
40     TYPE(MPI_Comm), INTENT(IN) :: comm
41     TYPE(MPI_Comm), INTENT(OUT) :: newcomm
42     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

43 MPI_Comm_connect(port_name, info, root, comm, newcomm, ierror)
44     CHARACTER(LEN=*), INTENT(IN) :: port_name
45     TYPE(MPI_Info), INTENT(IN) :: info
46     INTEGER, INTENT(IN) :: root
47     TYPE(MPI_Comm), INTENT(IN) :: comm

```

```

        TYPE(MPI_Comm), INTENT(OUT) :: newcomm
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_disconnect(comm, ierror)
        TYPE(MPI_Comm), INTENT(INOUT) :: comm
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_get_parent(parent, ierror)
        TYPE(MPI_Comm), INTENT(OUT) :: parent
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_join(fd, intercomm, ierror)
        INTEGER, INTENT(IN) :: fd
        TYPE(MPI_Comm), INTENT(OUT) :: intercomm
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_spawn(command, argv, maxprocs, info, root, comm, intercomm,
               array_of_errcodes, ierror)
        CHARACTER(LEN=*), INTENT(IN) :: command, argv(*)
        INTEGER, INTENT(IN) :: maxprocs, root
        TYPE(MPI_Info), INTENT(IN) :: info
        TYPE(MPI_Comm), INTENT(IN) :: comm
        TYPE(MPI_Comm), INTENT(OUT) :: intercomm
        INTEGER :: array_of_errcodes(*)
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Comm_spawn_multiple(count, array_of_commands, array_of_argv,
                       array_of_maxprocs, array_of_info, root, comm, intercomm,
                       array_of_errcodes, ierror)
        INTEGER, INTENT(IN) :: count, array_of_maxprocs(*), root
        CHARACTER(LEN=*), INTENT(IN) :: array_of_commands(*)
        CHARACTER(LEN=*), INTENT(IN) :: array_of_argv(count, *)
        TYPE(MPI_Info), INTENT(IN) :: array_of_info(*)
        TYPE(MPI_Comm), INTENT(IN) :: comm
        TYPE(MPI_Comm), INTENT(OUT) :: intercomm
        INTEGER :: array_of_errcodes(*)
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Lookup_name(service_name, info, port_name, ierror)
        CHARACTER(LEN=*), INTENT(IN) :: service_name
        TYPE(MPI_Info), INTENT(IN) :: info
        CHARACTER(LEN=MPI_MAX_PORT_NAME), INTENT(OUT) :: port_name
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Open_port(info, port_name, ierror)
        TYPE(MPI_Info), INTENT(IN) :: info
        CHARACTER(LEN=MPI_MAX_PORT_NAME), INTENT(OUT) :: port_name
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Publish_name(service_name, info, port_name, ierror)
        TYPE(MPI_Info), INTENT(IN) :: info

```

```

1      CHARACTER(LEN=*), INTENT(IN) :: service_name, port_name
2      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
3
4      MPI_Unpublish_name(service_name, info, port_name, ierror)
5      CHARACTER(LEN=*), INTENT(IN) :: service_name, port_name
6      TYPE(MPI_Info), INTENT(IN) :: info
7      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
8

```

A.3.9 One-Sided Communications Fortran 2008 Bindings

```

10
11      MPI_Accumulate(origin_addr, origin_count, origin_datatype, target_rank,
12                    target_disp, target_count, target_datatype, op, win, ierror)
13      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
14      INTEGER, INTENT(IN) :: origin_count, target_rank, target_count
15      TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype
16      INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
17      TYPE(MPI_Op), INTENT(IN) :: op
18      TYPE(MPI_Win), INTENT(IN) :: win
19      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
20
21      MPI_Compare_and_swap(origin_addr, compare_addr, result_addr, datatype,
22                           target_rank, target_disp, win, ierror)
23      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
24      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: compare_addr
25      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result_addr
26      TYPE(MPI_Datatype), INTENT(IN) :: datatype
27      INTEGER, INTENT(IN) :: target_rank
28      INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
29      TYPE(MPI_Win), INTENT(IN) :: win
30      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
31
32      MPI_Fetch_and_op(origin_addr, result_addr, datatype, target_rank,
33                      target_disp, op, win, ierror)
34      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
35      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result_addr
36      TYPE(MPI_Datatype), INTENT(IN) :: datatype
37      INTEGER, INTENT(IN) :: target_rank
38      INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
39      TYPE(MPI_Op), INTENT(IN) :: op
40      TYPE(MPI_Win), INTENT(IN) :: win
41      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
42
43      MPI_Get_accumulate(origin_addr, origin_count, origin_datatype, result_addr,
44                        result_count, result_datatype, target_rank, target_disp,
45                        target_count, target_datatype, op, win, ierror)
46      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
47      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result_addr
48      INTEGER, INTENT(IN) :: origin_count, result_count, target_rank,
49      target_count

```



```

TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype,
result_datatype
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
TYPE(MPI_Op), INTENT(IN) :: op
TYPE(MPI_Win), INTENT(IN) :: win
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Get(origin_addr, origin_count, origin_datatype, target_rank,
        target_disp, target_count, target_datatype, win, ierror)
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: origin_addr
INTEGER, INTENT(IN) :: origin_count, target_rank, target_count
TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
TYPE(MPI_Win), INTENT(IN) :: win
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Put(origin_addr, origin_count, origin_datatype, target_rank,
        target_disp, target_count, target_datatype, win, ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
INTEGER, INTENT(IN) :: origin_count, target_rank, target_count
TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
TYPE(MPI_Win), INTENT(IN) :: win
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Raccumulate(origin_addr, origin_count, origin_datatype, target_rank,
        target_disp, target_count, target_datatype, op, win, request,
        ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
INTEGER, INTENT(IN) :: origin_count, target_rank, target_count
TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
TYPE(MPI_Op), INTENT(IN) :: op
TYPE(MPI_Win), INTENT(IN) :: win
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Rget_accumulate(origin_addr, origin_count, origin_datatype,
        result_addr, result_count, result_datatype, target_rank,
        target_disp, target_count, target_datatype, op, win, request,
        ierror)
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result_addr
INTEGER, INTENT(IN) :: origin_count, result_count, target_rank,
target_count
TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype,
result_datatype
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
TYPE(MPI_Op), INTENT(IN) :: op

```

```

1      TYPE(MPI_Win), INTENT(IN) :: win
2      TYPE(MPI_Request), INTENT(OUT) :: request
3      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
4
5      MPI_Rget(origin_addr, origin_count, origin_datatype, target_rank,
6              target_disp, target_count, target_datatype, win, request,
7              ierror)
8      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: origin_addr
9      INTEGER, INTENT(IN) :: origin_count, target_rank, target_count
10     TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype
11     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
12     TYPE(MPI_Win), INTENT(IN) :: win
13     TYPE(MPI_Request), INTENT(OUT) :: request
14     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
15
16     MPI_Rput(origin_addr, origin_count, origin_datatype, target_rank,
17             target_disp, target_count, target_datatype, win, request,
18             ierror)
19     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin_addr
20     INTEGER, INTENT(IN) :: origin_count, target_rank, target_count
21     TYPE(MPI_Datatype), INTENT(IN) :: origin_datatype, target_datatype
22     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: target_disp
23     TYPE(MPI_Win), INTENT(IN) :: win
24     TYPE(MPI_Request), INTENT(OUT) :: request
25     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
26
27     MPI_Win_allocate_shared(size, disp_unit, info, comm, baseptr, win, ierror)
28     USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
29     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
30     INTEGER, INTENT(IN) :: disp_unit
31     TYPE(MPI_Info), INTENT(IN) :: info
32     TYPE(MPI_Comm), INTENT(IN) :: comm
33     TYPE(C_PTR), INTENT(OUT) :: baseptr
34     TYPE(MPI_Win), INTENT(OUT) :: win
35     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
36
37     MPI_Win_allocate(size, disp_unit, info, comm, baseptr, win, ierror)
38     USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
39     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
40     INTEGER, INTENT(IN) :: disp_unit
41     TYPE(MPI_Info), INTENT(IN) :: info
42     TYPE(MPI_Comm), INTENT(IN) :: comm
43     TYPE(C_PTR), INTENT(OUT) :: baseptr
44     TYPE(MPI_Win), INTENT(OUT) :: win
45     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
46
47     MPI_Win_attach(win, base, size, ierror)
48     TYPE(MPI_Win), INTENT(IN) :: win
49     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: base
50     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size

```

```

    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
1
MPI_Win_complete(win, ierror)
2
    TYPE(MPI_Win), INTENT(IN) :: win
3
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
4
5
MPI_Win_create(base, size, disp_unit, info, comm, win, ierror)
6
    TYPE(*), DIMENSION(..), ASYNCHRONOUS :: base
7
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
8
    INTEGER, INTENT(IN) :: disp_unit
9
    TYPE(MPI_Info), INTENT(IN) :: info
10
    TYPE(MPI_Comm), INTENT(IN) :: comm
11
    TYPE(MPI_Win), INTENT(OUT) :: win
12
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
13
14
MPI_Win_create_dynamic(info, comm, win, ierror)
15
    TYPE(MPI_Info), INTENT(IN) :: info
16
    TYPE(MPI_Comm), INTENT(IN) :: comm
17
    TYPE(MPI_Win), INTENT(OUT) :: win
18
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
19
20
MPI_Win_detach(win, base, ierror)
21
    TYPE(MPI_Win), INTENT(IN) :: win
22
    TYPE(*), DIMENSION(..), ASYNCHRONOUS :: base
23
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
24
25
MPI_Win_fence(assert, win, ierror)
26
    INTEGER, INTENT(IN) :: assert
27
    TYPE(MPI_Win), INTENT(IN) :: win
28
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
29
30
MPI_Win_flush_all(win, ierror)
31
    TYPE(MPI_Win), INTENT(IN) :: win
32
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
33
34
MPI_Win_flush_local_all(win, ierror)
35
    TYPE(MPI_Win), INTENT(IN) :: win
36
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
37
38
MPI_Win_flush_local(rank, win, ierror)
39
    INTEGER, INTENT(IN) :: rank
40
    TYPE(MPI_Win), INTENT(IN) :: win
41
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
42
43
MPI_Win_flush(rank, win, ierror)
44
    INTEGER, INTENT(IN) :: rank
45
    TYPE(MPI_Win), INTENT(IN) :: win
46
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
47
48
MPI_Win_free(win, ierror)
49
    TYPE(MPI_Win), INTENT(INOUT) :: win
50
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
51

```

```

1  MPI_Win_get_group(win, group, ierror)
2      TYPE(MPI_Win), INTENT(IN) :: win
3      TYPE(MPI_Group), INTENT(OUT) :: group
4      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
5
6  MPI_Win_get_info(win, info_used, ierror)
7      TYPE(MPI_Win), INTENT(IN) :: win
8      TYPE(MPI_Info), INTENT(OUT) :: info_used
9      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
10
11 MPI_Win_lock_all(assert, win, ierror)
12     INTEGER, INTENT(IN) :: assert
13     TYPE(MPI_Win), INTENT(IN) :: win
14     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
15
16 MPI_Win_lock(lock_type, rank, assert, win, ierror)
17     INTEGER, INTENT(IN) :: lock_type, rank, assert
18     TYPE(MPI_Win), INTENT(IN) :: win
19     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
20
21 MPI_Win_post(group, assert, win, ierror)
22     TYPE(MPI_Group), INTENT(IN) :: group
23     INTEGER, INTENT(IN) :: assert
24     TYPE(MPI_Win), INTENT(IN) :: win
25     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
26
27 MPI_Win_set_info(win, info, ierror)
28     TYPE(MPI_Win), INTENT(IN) :: win
29     TYPE(MPI_Info), INTENT(IN) :: info
30     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
31
32 MPI_Win_shared_query(win, rank, size, disp_unit, baseptr, ierror)
33     USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
34     TYPE(MPI_Win), INTENT(IN) :: win
35     INTEGER, INTENT(IN) :: rank
36     INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: size
37     INTEGER, INTENT(OUT) :: disp_unit
38     TYPE(C_PTR), INTENT(OUT) :: baseptr
39     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
40
41 MPI_Win_start(group, assert, win, ierror)
42     TYPE(MPI_Group), INTENT(IN) :: group
43     INTEGER, INTENT(IN) :: assert
44     TYPE(MPI_Win), INTENT(IN) :: win
45     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
46
47 MPI_Win_sync(win, ierror)
48     TYPE(MPI_Win), INTENT(IN) :: win
49     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
50
51 MPI_Win_test(win, flag, ierror)
52     TYPE(MPI_Win), INTENT(IN) :: win
53     INTEGER, INTENT(OUT) :: flag
54     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

LOGICAL, INTENT(OUT) :: flag
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_unlock_all(win, ierror)
TYPE(MPI_Win), INTENT(IN) :: win
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_unlock(rank, win, ierror)
INTEGER, INTENT(IN) :: rank
TYPE(MPI_Win), INTENT(IN) :: win
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Win_wait(win, ierror)
TYPE(MPI_Win), INTENT(IN) :: win
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

A.3.10 External Interfaces Fortran 2008 Bindings
MPI_Grequest_complete(request, ierror)
TYPE(MPI_Request), INTENT(IN) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Grequest_start(query_fn, free_fn, cancel_fn, extra_state, request,
ierror)
PROCEDURE(MPI_Grequest_query_function) :: query_fn
PROCEDURE(MPI_Grequest_free_function) :: free_fn
PROCEDURE(MPI_Grequest_cancel_function) :: cancel_fn
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: extra_state
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Init_thread(required, provided, ierror)
INTEGER, INTENT(IN) :: required
INTEGER, INTENT(OUT) :: provided
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Is_thread_main(flag, ierror)
LOGICAL, INTENT(OUT) :: flag
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Query_thread(provided, ierror)
INTEGER, INTENT(OUT) :: provided
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Status_set_cancelled(status, flag, ierror)
TYPE(MPI_Status), INTENT(INOUT) :: status
LOGICAL, INTENT(OUT) :: flag
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
MPI_Status_set_elements(status, datatype, count, ierror)
TYPE(MPI_Status), INTENT(INOUT) :: status
TYPE(MPI_Datatype), INTENT(IN) :: datatype

```

```

1      INTEGER, INTENT(IN) :: count
2      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
3
4      MPI_Status_set_elements_x(status, datatype, count, ierror)
5      TYPE(MPI_Status), INTENT(INOUT) :: status
6      TYPE(MPI_Datatype), INTENT(IN) :: datatype
7      INTEGER(KIND = MPI_COUNT_KIND), INTENT(IN) :: count
8      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
9

```

A.3.11 I/O Fortran 2008 Bindings

```

10
11
12      MPI_CONVERSION_FN_NULL(userbuf, datatype, count, filebuf, position,
13          extra_state, ierror)
14      USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
15      TYPE(C_PTR), VALUE :: userbuf, filebuf
16      TYPE(MPI_Datatype) :: datatype
17      INTEGER :: count, ierror
18      INTEGER(KIND=MPI_OFFSET_KIND) :: position
19      INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state
20
21      MPI_File_close(fh, ierror)
22      TYPE(MPI_File), INTENT(INOUT) :: fh
23      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
24
25      MPI_File_delete(filename, info, ierror)
26      CHARACTER(LEN=*), INTENT(IN) :: filename
27      TYPE(MPI_Info), INTENT(IN) :: info
28      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
29
30      MPI_File_get_amode(fh, amode, ierror)
31      TYPE(MPI_File), INTENT(IN) :: fh
32      INTEGER, INTENT(OUT) :: amode
33      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
34
35      MPI_File_get_atomicity(fh, flag, ierror)
36      TYPE(MPI_File), INTENT(IN) :: fh
37      LOGICAL, INTENT(OUT) :: flag
38      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
39
40      MPI_File_get_byte_offset(fh, offset, disp, ierror)
41      TYPE(MPI_File), INTENT(IN) :: fh
42      INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
43      INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) :: disp
44      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
45
46      MPI_File_get_group(fh, group, ierror)
47      TYPE(MPI_File), INTENT(IN) :: fh
48      TYPE(MPI_Group), INTENT(OUT) :: group
49      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
50
51      MPI_File_get_info(fh, info_used, ierror)
52

```

```

TYPE(MPI_File), INTENT(IN) :: fh
TYPE(MPI_Info), INTENT(OUT) :: info_used
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_get_position(fh, offset, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) :: offset
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_get_position_shared(fh, offset, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) :: offset
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_get_size(fh, size, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) :: size
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_get_type_extent(fh, datatype, extent, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(MPI_Datatype), INTENT(IN) :: datatype
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) :: extent
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_get_view(fh, disp, etype, filetype, datarep, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) :: disp
TYPE(MPI_Datatype), INTENT(OUT) :: etype, filetype
CHARACTER(LEN=*), INTENT(OUT) :: datarep
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_iread_all(fh, buf, count, datatype, request, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_iread_at_all(fh, offset, buf, count, datatype, request, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_iread_at(fh, offset, buf, count, datatype, request, ierror)
TYPE(MPI_File), INTENT(IN) :: fh

```

```

1      INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
2      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
3      INTEGER, INTENT(IN) :: count
4      TYPE(MPI_Datatype), INTENT(IN) :: datatype
5      TYPE(MPI_Request), INTENT(OUT) :: request
6      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
7
8      MPI_File_iread(fh, buf, count, datatype, request, ierror)
9      TYPE(MPI_File), INTENT(IN) :: fh
10     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
11     INTEGER, INTENT(IN) :: count
12     TYPE(MPI_Datatype), INTENT(IN) :: datatype
13     TYPE(MPI_Request), INTENT(OUT) :: request
14     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
15
16     MPI_File_iread_shared(fh, buf, count, datatype, request, ierror)
17     TYPE(MPI_File), INTENT(IN) :: fh
18     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
19     INTEGER, INTENT(IN) :: count
20     TYPE(MPI_Datatype), INTENT(IN) :: datatype
21     TYPE(MPI_Request), INTENT(OUT) :: request
22     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
23
24     MPI_File_iwrite_all(fh, buf, count, datatype, request, ierror)
25     TYPE(MPI_File), INTENT(IN) :: fh
26     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
27     INTEGER, INTENT(IN) :: count
28     TYPE(MPI_Datatype), INTENT(IN) :: datatype
29     TYPE(MPI_Request), INTENT(OUT) :: request
30     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
31
32     MPI_File_iwrite_at_all(fh, offset, buf, count, datatype, request, ierror)
33     TYPE(MPI_File), INTENT(IN) :: fh
34     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
35     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
36     INTEGER, INTENT(IN) :: count
37     TYPE(MPI_Datatype), INTENT(IN) :: datatype
38     TYPE(MPI_Request), INTENT(OUT) :: request
39     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
40
41     MPI_File_iwrite_at(fh, offset, buf, count, datatype, request, ierror)
42     TYPE(MPI_File), INTENT(IN) :: fh
43     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
44     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
45     INTEGER, INTENT(IN) :: count
46     TYPE(MPI_Datatype), INTENT(IN) :: datatype
47     TYPE(MPI_Request), INTENT(OUT) :: request
48     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
49
50     MPI_File_iwrite(fh, buf, count, datatype, request, ierror)

```



```

TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_iwrite_shared(fh, buf, count, datatype, request, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_open(comm, filename, amode, info, fh, ierror)
TYPE(MPI_Comm), INTENT(IN) :: comm
CHARACTER(LEN=*), INTENT(IN) :: filename
INTEGER, INTENT(IN) :: amode
TYPE(MPI_Info), INTENT(IN) :: info
TYPE(MPI_File), INTENT(OUT) :: fh
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_preallocate(fh, size, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: size
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_read_all_begin(fh, buf, count, datatype, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_read_all_end(fh, buf, status, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_read_all(fh, buf, count, datatype, status, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..) :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_read_at_all_begin(fh, offset, buf, count, datatype, ierror)
TYPE(MPI_File), INTENT(IN) :: fh

```

```

1      INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
2      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
3      INTEGER, INTENT(IN) :: count
4      TYPE(MPI_Datatype), INTENT(IN) :: datatype
5      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
6
7  MPI_File_read_at_all_end(fh, buf, status, ierror)
8      TYPE(MPI_File), INTENT(IN) :: fh
9      TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
10     TYPE(MPI_Status) :: status
11     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
12
13 MPI_File_read_at_all(fh, offset, buf, count, datatype, status, ierror)
14     TYPE(MPI_File), INTENT(IN) :: fh
15     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
16     TYPE(*), DIMENSION(..) :: buf
17     INTEGER, INTENT(IN) :: count
18     TYPE(MPI_Datatype), INTENT(IN) :: datatype
19     TYPE(MPI_Status) :: status
20     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
21
22 MPI_File_read_at(fh, offset, buf, count, datatype, status, ierror)
23     TYPE(MPI_File), INTENT(IN) :: fh
24     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
25     TYPE(*), DIMENSION(..) :: buf
26     INTEGER, INTENT(IN) :: count
27     TYPE(MPI_Datatype), INTENT(IN) :: datatype
28     TYPE(MPI_Status) :: status
29     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
30
31 MPI_File_read(fh, buf, count, datatype, status, ierror)
32     TYPE(MPI_File), INTENT(IN) :: fh
33     TYPE(*), DIMENSION(..) :: buf
34     INTEGER, INTENT(IN) :: count
35     TYPE(MPI_Datatype), INTENT(IN) :: datatype
36     TYPE(MPI_Status) :: status
37     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
38
39 MPI_File_read_ordered_begin(fh, buf, count, datatype, ierror)
40     TYPE(MPI_File), INTENT(IN) :: fh
41     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
42     INTEGER, INTENT(IN) :: count
43     TYPE(MPI_Datatype), INTENT(IN) :: datatype
44     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
45
46 MPI_File_read_ordered_end(fh, buf, status, ierror)
47     TYPE(MPI_File), INTENT(IN) :: fh
48     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
49     TYPE(MPI_Status) :: status
50     INTEGER, OPTIONAL, INTENT(OUT) :: ierror

```

```

MPI_File_read_ordered(fh, buf, count, datatype, status, ierror)      1
    TYPE(MPI_File), INTENT(IN) :: fh                                2
    TYPE(*), DIMENSION(..) :: buf                                  3
    INTEGER, INTENT(IN) :: count                                    4
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                     5
    TYPE(MPI_Status) :: status                                      6
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        7
                                                                    8
MPI_File_read_shared(fh, buf, count, datatype, status, ierror)      9
    TYPE(MPI_File), INTENT(IN) :: fh                                10
    TYPE(*), DIMENSION(..) :: buf                                  11
    INTEGER, INTENT(IN) :: count                                    12
    TYPE(MPI_Datatype), INTENT(IN) :: datatype                     13
    TYPE(MPI_Status) :: status                                      14
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        15
                                                                    16
MPI_File_seek(fh, offset, whence, ierror)                            17
    TYPE(MPI_File), INTENT(IN) :: fh                                18
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset           19
    INTEGER, INTENT(IN) :: whence                                  20
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        21
                                                                    22
MPI_File_seek_shared(fh, offset, whence, ierror)                     23
    TYPE(MPI_File), INTENT(IN) :: fh                                24
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset           25
    INTEGER, INTENT(IN) :: whence                                  26
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        27
                                                                    28
MPI_File_set_atomicity(fh, flag, ierror)                             29
    TYPE(MPI_File), INTENT(IN) :: fh                                30
    LOGICAL, INTENT(IN) :: flag                                    31
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        32
                                                                    33
MPI_File_set_info(fh, info, ierror)                                   34
    TYPE(MPI_File), INTENT(IN) :: fh                                35
    TYPE(MPI_Info), INTENT(IN) :: info                             36
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        37
                                                                    38
MPI_File_set_size(fh, size, ierror)                                   39
    TYPE(MPI_File), INTENT(IN) :: fh                                40
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: size             41
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                        42
                                                                    43
MPI_File_set_view(fh, disp, etype, filetype, datarep, info, ierror) 44
    TYPE(MPI_File), INTENT(IN) :: fh                                45
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: disp             46
    TYPE(MPI_Datatype), INTENT(IN) :: etype, filetype             47
    CHARACTER(LEN=*), INTENT(IN) :: datarep                       48
    TYPE(MPI_Info), INTENT(IN) :: info                             49
    INTEGER, OPTIONAL, INTENT(OUT) :: ierror                       50

```

```

1  MPI_File_sync(fh, ierror)
2      TYPE(MPI_File), INTENT(IN) :: fh
3      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
4
5  MPI_File_write_all_begin(fh, buf, count, datatype, ierror)
6      TYPE(MPI_File), INTENT(IN) :: fh
7      TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
8      INTEGER, INTENT(IN) :: count
9      TYPE(MPI_Datatype), INTENT(IN) :: datatype
10     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
11
12 MPI_File_write_all_end(fh, buf, status, ierror)
13     TYPE(MPI_File), INTENT(IN) :: fh
14     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
15     TYPE(MPI_Status) :: status
16     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
17
18 MPI_File_write_all(fh, buf, count, datatype, status, ierror)
19     TYPE(MPI_File), INTENT(IN) :: fh
20     TYPE(*), DIMENSION(..), INTENT(IN) :: buf
21     INTEGER, INTENT(IN) :: count
22     TYPE(MPI_Datatype), INTENT(IN) :: datatype
23     TYPE(MPI_Status) :: status
24     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
25
26 MPI_File_write_at_all_begin(fh, offset, buf, count, datatype, ierror)
27     TYPE(MPI_File), INTENT(IN) :: fh
28     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
29     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
30     INTEGER, INTENT(IN) :: count
31     TYPE(MPI_Datatype), INTENT(IN) :: datatype
32     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
33
34 MPI_File_write_at_all_end(fh, buf, status, ierror)
35     TYPE(MPI_File), INTENT(IN) :: fh
36     TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
37     TYPE(MPI_Status) :: status
38     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
39
40 MPI_File_write_at_all(fh, offset, buf, count, datatype, status, ierror)
41     TYPE(MPI_File), INTENT(IN) :: fh
42     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset
43     TYPE(*), DIMENSION(..), INTENT(IN) :: buf
44     INTEGER, INTENT(IN) :: count
45     TYPE(MPI_Datatype), INTENT(IN) :: datatype
46     TYPE(MPI_Status) :: status
47     INTEGER, OPTIONAL, INTENT(OUT) :: ierror
48
49 MPI_File_write_at(fh, offset, buf, count, datatype, status, ierror)
50     TYPE(MPI_File), INTENT(IN) :: fh
51     INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) :: offset

```

```

TYPE(*), DIMENSION(..), INTENT(IN) :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_write(fh, buf, count, datatype, status, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN) :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_write_ordered_begin(fh, buf, count, datatype, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_write_ordered_end(fh, buf, status, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: buf
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_write_ordered(fh, buf, count, datatype, status, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN) :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_File_write_shared(fh, buf, count, datatype, status, ierror)
TYPE(MPI_File), INTENT(IN) :: fh
TYPE(*), DIMENSION(..), INTENT(IN) :: buf
INTEGER, INTENT(IN) :: count
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror

MPI_Register_datarep(datarep, read_conversion_fn, write_conversion_fn,
dtype_file_extent_fn, extra_state, ierror)
CHARACTER(LEN=*), INTENT(IN) :: datarep
PROCEDURE(MPI_Datarep_conversion_function) :: read_conversion_fn
PROCEDURE(MPI_Datarep_conversion_function) :: write_conversion_fn
PROCEDURE(MPI_Datarep_extent_function) :: dtype_file_extent_fn
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: extra_state

```

```
1      INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

4 A.3.12 Language Bindings Fortran 2008 Bindings

```
5 MPI_F_sync_reg(buf)
```

```
6     TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
```

```
8 MPI_Sizeof(x, size, ierror)
```

```
9     TYPE(*), DIMENSION(..) :: x
```

```
10    INTEGER, INTENT(OUT) :: size
```

```
11    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

```
12 MPI_Status_f082f(f08_status, f_status, ierror)
```

```
13    TYPE(MPI_Status), INTENT(IN) :: f08_status
```

```
14    INTEGER, INTENT(OUT) :: f_status(MPI_STATUS_SIZE)
```

```
15    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

```
17 MPI_Status_f2f08(f_status, f08_status, ierror)
```

```
18    INTEGER, INTENT(IN) :: f_status(MPI_STATUS_SIZE)
```

```
19    TYPE(MPI_Status), INTENT(OUT) :: f08_status
```

```
20    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

```
21 MPI_Type_create_f90_complex(p, r, newtype, ierror)
```

```
22    INTEGER, INTENT(IN) :: p, r
```

```
23    TYPE(MPI_Datatype), INTENT(OUT) :: newtype
```

```
24    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

```
26 MPI_Type_create_f90_integer(r, newtype, ierror)
```

```
27    INTEGER, INTENT(IN) :: r
```

```
28    TYPE(MPI_Datatype), INTENT(OUT) :: newtype
```

```
29    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

```
30 MPI_Type_create_f90_real(p, r, newtype, ierror)
```

```
31    INTEGER, INTENT(IN) :: p, r
```

```
32    TYPE(MPI_Datatype), INTENT(OUT) :: newtype
```

```
33    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

```
35 MPI_Type_match_size(typeclass, size, datatype, ierror)
```

```
36    INTEGER, INTENT(IN) :: typeclass, size
```

```
37    TYPE(MPI_Datatype), INTENT(OUT) :: datatype
```

```
38    INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

41 A.3.13 Tools / Profiling Interface Fortran 2008 Bindings

```
42 MPI_Pcontrol(level)
```

```
43    INTEGER, INTENT(IN) :: level
```

A.4 Fortran Bindings with mpif.h or the mpi Module

A.4.1 Point-to-Point Communication Fortran Bindings

```

MPI_BSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR

MPI_BSEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_BUFFER_ATTACH(BUFFER, SIZE, IERROR)
    <type> BUFFER(*)
    INTEGER SIZE, IERROR

MPI_BUFFER_DETACH(BUFFER_ADDR, SIZE, IERROR)
    <type> BUFFER_ADDR(*)
    INTEGER SIZE, IERROR

MPI_CANCEL(REQUEST, IERROR)
    INTEGER REQUEST, IERROR

MPI_GET_COUNT(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, COUNT, IERROR

MPI_IBSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_IMPROBE(SOURCE, TAG, COMM, FLAG, MESSAGE, STATUS, IERROR)
    INTEGER SOURCE, TAG, COMM, MESSAGE, STATUS(MPI_STATUS_SIZE), IERROR
    LOGICAL FLAG

MPI_IMRECV(BUF, COUNT, DATATYPE, MESSAGE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, MESSAGE, REQUEST, IERROR

MPI_IPROBE(SOURCE, TAG, COMM, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR

MPI_Irecv(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR

MPI_IRSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_ISEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

```

```

1  MPI_ISSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
2      <type> BUF(*)
3      INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR
4
5  MPI_MPROBE(SOURCE, TAG, COMM, MESSAGE, STATUS, IERROR)
6      INTEGER SOURCE, TAG, COMM, MESSAGE, STATUS(MPI_STATUS_SIZE), IERROR
7
8  MPI_MRECV(BUF, COUNT, DATATYPE, MESSAGE, STATUS, IERROR)
9      <type> BUF(*)
10     INTEGER COUNT, DATATYPE, MESSAGE, STATUS(MPI_STATUS_SIZE), IERROR
11
12 MPI_PROBE(SOURCE, TAG, COMM, STATUS, IERROR)
13     INTEGER SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR
14
15 MPI_RECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS, IERROR)
16     <type> BUF(*)
17     INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE),
18     IERROR
19
20 MPI_RECV_INIT(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR)
21     <type> BUF(*)
22     INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR
23
24 MPI_REQUEST_FREE(REQUEST, IERROR)
25     INTEGER REQUEST, IERROR
26
27 MPI_REQUEST_GET_STATUS( REQUEST, FLAG, STATUS, IERROR)
28     INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR
29     LOGICAL FLAG
30
31 MPI_RSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
32     <type> BUF(*)
33     INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR
34
35 MPI_RSEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
36     <type> BUF(*)
37     INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR
38
39 MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
40     <type> BUF(*)
41     INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR
42
43 MPI_SEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
44     <type> BUF(*)
45     INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR
46
47 MPI_SENDRECV_REPLACE(BUF, COUNT, DATATYPE, DEST, SENDTAG, SOURCE, RECVTAG,
48     COMM, STATUS, IERROR)
49     <type> BUF(*)
50     INTEGER COUNT, DATATYPE, DEST, SENDTAG, SOURCE, RECVTAG, COMM,
51     STATUS(MPI_STATUS_SIZE), IERROR
52
53 MPI_SENDRECV(SENDBUF, SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVBUFF,
54     RECVCOUNT, RECVMODE, SOURCE, RECVTAG, COMM, STATUS, IERROR)

```



```

    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVCOUNT, RECVTYPE,
    SOURCE, RECVTAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR
MPI_SSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR
MPI_SSEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR
MPI_STARTALL(COUNT, ARRAY_OF_REQUESTS, IERROR)
    INTEGER COUNT, ARRAY_OF_REQUESTS(*), IERROR
MPI_START(REQUEST, IERROR)
    INTEGER REQUEST, IERROR
MPI_TESTALL(COUNT, ARRAY_OF_REQUESTS, FLAG, ARRAY_OF_STATUSES, IERROR)
    LOGICAL FLAG
    INTEGER COUNT, ARRAY_OF_REQUESTS(*),
    ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR
MPI_TESTANY(COUNT, ARRAY_OF_REQUESTS, INDEX, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER COUNT, ARRAY_OF_REQUESTS(*), INDEX, STATUS(MPI_STATUS_SIZE),
    IERROR
MPI_TEST_CANCELLED(STATUS, FLAG, IERROR)
    LOGICAL FLAG
    INTEGER STATUS(MPI_STATUS_SIZE), IERROR
MPI_TEST(REQUEST, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR
MPI_TESTSOME(INCOUNT, ARRAY_OF_REQUESTS, OUTCOUNT, ARRAY_OF_INDICES,
    ARRAY_OF_STATUSES, IERROR)
    INTEGER INCOUNT, ARRAY_OF_REQUESTS(*), OUTCOUNT, ARRAY_OF_INDICES(*),
    ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR
MPI_WAITALL(COUNT, ARRAY_OF_REQUESTS, ARRAY_OF_STATUSES, IERROR)
    INTEGER COUNT, ARRAY_OF_REQUESTS(*)
    INTEGER ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR
MPI_WAITANY(COUNT, ARRAY_OF_REQUESTS, INDEX, STATUS, IERROR)
    INTEGER COUNT, ARRAY_OF_REQUESTS(*), INDEX, STATUS(MPI_STATUS_SIZE),
    IERROR
MPI_WAIT(REQUEST, STATUS, IERROR)
    INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR
MPI_WAITSOME(INCOUNT, ARRAY_OF_REQUESTS, OUTCOUNT, ARRAY_OF_INDICES,
    ARRAY_OF_STATUSES, IERROR)

```

```

1      INTEGER INCOUNT, ARRAY_OF_REQUESTS(*), OUTCOUNT, ARRAY_OF_INDICES(*),
2      ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR

```

A.4.2 Datatypes Fortran Bindings

```

6      INTEGER(KIND=MPI_ADDRESS_KIND) MPI_AINT_ADD(BASE, DISP)
7          INTEGER(KIND=MPI_ADDRESS_KIND) BASE, DISP
8
9      INTEGER(KIND=MPI_ADDRESS_KIND) MPI_AINT_DIFF(ADDR1, ADDR2)
10         INTEGER(KIND=MPI_ADDRESS_KIND) ADDR1, ADDR2
11
12      MPI_GET_ADDRESS(LOCATION, ADDRESS, IERROR)
13         <type> LOCATION(*)
14         INTEGER IERROR
15         INTEGER(KIND=MPI_ADDRESS_KIND) ADDRESS
16
17      MPI_GET_ELEMENTS(STATUS, DATATYPE, COUNT, IERROR)
18         INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, COUNT, IERROR
19
20      MPI_GET_ELEMENTS_X(STATUS, DATATYPE, COUNT, IERROR)
21         INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, IERROR
22         INTEGER(KIND=MPI_COUNT_KIND) COUNT
23
24      MPI_PACK_EXTERNAL(DATAREP, INBUF, INCOUNT, DATATYPE, OUTBUF, OUTSIZE,
25         POSITION, IERROR)
26         INTEGER INCOUNT, DATATYPE, IERROR
27         INTEGER(KIND=MPI_ADDRESS_KIND) OUTSIZE, POSITION
28         CHARACTER*(*) DATAREP
29         <type> INBUF(*), OUTBUF(*)
30
31      MPI_PACK_EXTERNAL_SIZE(DATAREP, INCOUNT, DATATYPE, SIZE, IERROR)
32         INTEGER INCOUNT, DATATYPE, IERROR
33         INTEGER(KIND=MPI_ADDRESS_KIND) SIZE
34         CHARACTER*(*) DATAREP
35
36      MPI_PACK(INBUF, INCOUNT, DATATYPE, OUTBUF, OUTSIZE, POSITION, COMM, IERROR)
37         <type> INBUF(*), OUTBUF(*)
38         INTEGER INCOUNT, DATATYPE, OUTSIZE, POSITION, COMM, IERROR
39
40      MPI_PACK_SIZE(INCOUNT, DATATYPE, COMM, SIZE, IERROR)
41         INTEGER INCOUNT, DATATYPE, COMM, SIZE, IERROR
42
43      MPI_TYPE_COMMIT(DATATYPE, IERROR)
44         INTEGER DATATYPE, IERROR
45
46      MPI_TYPE_CONTIGUOUS(COUNT, OLDTYPE, NEWTYPE, IERROR)
47         INTEGER COUNT, OLDTYPE, NEWTYPE, IERROR
48
49      MPI_TYPE_CREATE_DARRAY(SIZE, RANK, NDIMS, ARRAY_OF_GSIZES,
50         ARRAY_OF_DISTIBS, ARRAY_OF_DARGS, ARRAY_OF_PSIZEs, ORDER,
51         OLDTYPE, NEWTYPE, IERROR)
52         INTEGER SIZE, RANK, NDIMS, ARRAY_OF_GSIZES(*), ARRAY_OF_DISTIBS(*),
53         ARRAY_OF_DARGS(*), ARRAY_OF_PSIZEs(*), ORDER, OLDTYPE, NEWTYPE, IERROR

```

```

MPI_TYPE_CREATE_HINDEXED_BLOCK(COUNT, BLOCKLENGTH, ARRAY_OF_DISPLACEMENTS,
                                OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)

MPI_TYPE_CREATE_HINDEXED(COUNT, ARRAY_OF_BLOCKLENGTHS,
                          ARRAY_OF_DISPLACEMENTS, OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)

MPI_TYPE_CREATE_HVECTOR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE,
                        IERROR)
    INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) STRIDE

MPI_TYPE_CREATE_INDEXED_BLOCK(COUNT, BLOCKLENGTH, ARRAY_OF_DISPLACEMENTS,
                              OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, BLOCKLENGTH, ARRAY_OF_DISPLACEMENTS(*), OLDTYPE,
    NEWTYPE, IERROR

MPI_TYPE_CREATE_RESIZED(OLDTYPE, LB, EXTENT, NEWTYPE, IERROR)
    INTEGER OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) LB, EXTENT

MPI_TYPE_CREATE_STRUCT(COUNT, ARRAY_OF_BLOCKLENGTHS,
                      ARRAY_OF_DISPLACEMENTS, ARRAY_OF_TYPES, NEWTYPE, IERROR)
    INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), ARRAY_OF_TYPES(*), NEWTYPE,
    IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)

MPI_TYPE_CREATE_SUBARRAY(NDIMS, ARRAY_OF_SIZES, ARRAY_OF_SUBSIZES,
                        ARRAY_OF_STARTS, ORDER, OLDTYPE, NEWTYPE, IERROR)
    INTEGER NDIMS, ARRAY_OF_SIZES(*), ARRAY_OF_SUBSIZES(*),
    ARRAY_OF_STARTS(*), ORDER, OLDTYPE, NEWTYPE, IERROR

MPI_TYPE_DUP(OLDTYPE, NEWTYPE, IERROR)
    INTEGER OLDTYPE, NEWTYPE, IERROR

MPI_TYPE_FREE(DATATYPE, IERROR)
    INTEGER DATATYPE, IERROR

MPI_TYPE_GET_CONTENTS(DATATYPE, MAX_INTEGERS, MAX_ADDRESSES, MAX_DATATYPES,
                     ARRAY_OF_INTEGERS, ARRAY_OF_ADDRESSES, ARRAY_OF_DATATYPES,
                     IERROR)
    INTEGER DATATYPE, MAX_INTEGERS, MAX_ADDRESSES, MAX_DATATYPES,
    ARRAY_OF_INTEGERS(*), ARRAY_OF_DATATYPES(*), IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_ADDRESSES(*)

MPI_TYPE_GET_ENVELOPE(DATATYPE, NUM_INTEGERS, NUM_ADDRESSES, NUM_DATATYPES,
                     COMBINER, IERROR)
    INTEGER DATATYPE, NUM_INTEGERS, NUM_ADDRESSES, NUM_DATATYPES, COMBINER,
    IERROR

```

```

1  MPI_TYPE_GET_EXTENT(DATATYPE, LB, EXTENT, IERROR)
2      INTEGER DATATYPE, IERROR
3      INTEGER(KIND = MPI_ADDRESS_KIND) LB, EXTENT
4
5  MPI_TYPE_GET_EXTENT_X(DATATYPE, LB, EXTENT, IERROR)
6      INTEGER DATATYPE, IERROR
7      INTEGER(KIND = MPI_COUNT_KIND) LB, EXTENT
8
9  MPI_TYPE_GET_TRUE_EXTENT(DATATYPE, TRUE_LB, TRUE_EXTENT, IERROR)
10     INTEGER DATATYPE, IERROR
11     INTEGER(KIND = MPI_ADDRESS_KIND) TRUE_LB, TRUE_EXTENT
12
13 MPI_TYPE_GET_TRUE_EXTENT_X(DATATYPE, TRUE_LB, TRUE_EXTENT, IERROR)
14     INTEGER DATATYPE, IERROR
15     INTEGER(KIND = MPI_COUNT_KIND) TRUE_LB, TRUE_EXTENT
16
17 MPI_TYPE_INDEXED(COUNT, ARRAY_OF_BLOCKLENGTHS, ARRAY_OF_DISPLACEMENTS,
18                 OLDTYPE, NEWTYPE, IERROR)
19     INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), ARRAY_OF_DISPLACEMENTS(*),
20     OLDTYPE, NEWTYPE, IERROR
21
22 MPI_TYPE_SIZE(DATATYPE, SIZE, IERROR)
23     INTEGER DATATYPE, SIZE, IERROR
24
25 MPI_TYPE_SIZE_X(DATATYPE, SIZE, IERROR)
26     INTEGER DATATYPE, IERROR
27     INTEGER(KIND = MPI_COUNT_KIND) SIZE
28
29 MPI_TYPE_VECTOR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE, IERROR)
30     INTEGER COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE, IERROR
31
32 MPI_UNPACK_EXTERNAL(DATAREP, INBUF, INSIZE, POSITION, OUTBUF, OUTCOUNT,
33                     DATATYPE, IERROR)
34     INTEGER OUTCOUNT, DATATYPE, IERROR
35     INTEGER(KIND=MPI_ADDRESS_KIND) INSIZE, POSITION
36     CHARACTER*(*) DATAREP
37     <type> INBUF(*), OUTBUF(*)
38
39 MPI_UNPACK(INBUF, INSIZE, POSITION, OUTBUF, OUTCOUNT, DATATYPE, COMM,
40            IERROR)
41     <type> INBUF(*), OUTBUF(*)
42     INTEGER INSIZE, POSITION, OUTCOUNT, DATATYPE, COMM, IERROR

```

A.4.3 Collective Communication Fortran Bindings

```

42 MPI_ALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
43              COMM, IERROR)
44     <type> SENDBUF(*), RECVBUF(*)
45     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR
46
47 MPI_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
48               RECVTYPE, COMM, IERROR)

```

```

    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
    IERROR
MPI_ALLREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, IERROR
MPI_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
    COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR
MPI_ALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF, RECVCOUNTS,
    RDISPLS, RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
    RECVTYPE, COMM, IERROR
MPI_ALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF, RECVCOUNTS,
    RDISPLS, RECVTYPES, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPES(*), RECVCOUNTS(*),
    RDISPLS(*), RECVTYPES(*), COMM, IERROR
MPI_BARRIER(COMM, IERROR)
    INTEGER COMM, IERROR
MPI_BCAST(BUFFER, COUNT, DATATYPE, ROOT, COMM, IERROR)
    <type> BUFFER(*)
    INTEGER COUNT, DATATYPE, ROOT, COMM, IERROR
MPI_EXSCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, IERROR
MPI_GATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
    ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, IERROR
MPI_GATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
    RECVTYPE, ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, ROOT,
    COMM, IERROR
MPI_IALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
    COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR
MPI_IALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,

```

```

1         RECVTYPE, COMM, REQUEST, IERROR)
2     <type> SENDBUF(*), RECVBUF(*)
3     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
4     REQUEST, IERROR
5
6 MPI_IALLREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, REQUEST,
7     IERROR)
8     <type> SENDBUF(*), RECVBUF(*)
9     INTEGER COUNT, DATATYPE, OP, COMM, REQUEST, IERROR
10
11 MPI_IALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
12     COMM, REQUEST, IERROR)
13     <type> SENDBUF(*), RECVBUF(*)
14     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR
15
16 MPI_IALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF, RECVCOUNTS,
17     RDISPLS, RECVTYPE, COMM, REQUEST, IERROR)
18     <type> SENDBUF(*), RECVBUF(*)
19     INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
20     RECVTYPE, COMM, REQUEST, IERROR
21
22 MPI_IALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF,
23     RECVCOUNTS, RDISPLS, RECVTYPES, COMM, REQUEST, IERROR)
24     <type> SENDBUF(*), RECVBUF(*)
25     INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPES(*), RECVCOUNTS(*),
26     RDISPLS(*), RECVTYPES(*), COMM, REQUEST, IERROR
27
28 MPI_IBARRIER(COMM, REQUEST, IERROR)
29     INTEGER COMM, REQUEST, IERROR
30
31 MPI_IBCAST(BUFFER, COUNT, DATATYPE, ROOT, COMM, REQUEST, IERROR)
32     <type> BUFFER(*)
33     INTEGER COUNT, DATATYPE, ROOT, COMM, REQUEST, IERROR
34
35 MPI_IEXSCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, REQUEST, IERROR)
36     <type> SENDBUF(*), RECVBUF(*)
37     INTEGER COUNT, DATATYPE, OP, COMM, REQUEST, IERROR
38
39 MPI_IGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
40     ROOT, COMM, REQUEST, IERROR)
41     <type> SENDBUF(*), RECVBUF(*)
42     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, REQUEST,
43     IERROR
44
45 MPI_IGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
46     RECVTYPE, ROOT, COMM, REQUEST, IERROR)
47     <type> SENDBUF(*), RECVBUF(*)
48     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, ROOT,
49     COMM, REQUEST, IERROR
50
51 MPI_IREDUCE_SCATTER_BLOCK(SENDBUF, RECVBUF, RECVCOUNT, DATATYPE, OP, COMM,
52     REQUEST, IERROR)

```

```

    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNT, DATATYPE, OP, COMM, REQUEST, IERROR
MPI_IREDUCE_SCATTER(SENDBUF, RECVBUF, RECVCOUNTS, DATATYPE, OP, COMM,
    REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNTS(*), DATATYPE, OP, COMM, REQUEST, IERROR
MPI_IREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM, REQUEST,
    IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, ROOT, COMM, REQUEST, IERROR
MPI_ISCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, REQUEST, IERROR
MPI_ISCATTER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
    ROOT, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, REQUEST,
    IERROR
MPI_ISCATTERV(SENDBUF, SENDCOUNTS, DISPLS, SENDTYPE, RECVBUF, RECVCOUNT,
    RECVTYPE, ROOT, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), DISPLS(*), SENDTYPE, RECVCOUNT, RECVTYPE, ROOT,
    COMM, REQUEST, IERROR
MPI_OP_COMMUTATIVE(OP, COMMUTE, IERROR)
    LOGICAL COMMUTE
    INTEGER OP, IERROR
MPI_OP_CREATE( USER_FN, COMMUTE, OP, IERROR)
    EXTERNAL USER_FN
    LOGICAL COMMUTE
    INTEGER OP, IERROR
MPI_OP_FREE(OP, IERROR)
    INTEGER OP, IERROR
MPI_REDUCE_LOCAL(INBUF, INOUTBUF, COUNT, DATATYPE, OP, IERROR)
    <type> INBUF(*), INOUTBUF(*)
    INTEGER COUNT, DATATYPE, OP, IERROR
MPI_REDUCE_SCATTER_BLOCK(SENDBUF, RECVBUF, RECVCOUNT, DATATYPE, OP, COMM,
    IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNT, DATATYPE, OP, COMM, IERROR
MPI_REDUCE_SCATTER(SENDBUF, RECVBUF, RECVCOUNTS, DATATYPE, OP, COMM,
    IERROR)
    <type> SENDBUF(*), RECVBUF(*)

```

```

1      INTEGER RECVCOUNTS(*), DATATYPE, OP, COMM, IERROR
2
3      MPI_REDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM, IERROR)
4          <type> SENDBUF(*), RECVBUF(*)
5          INTEGER COUNT, DATATYPE, OP, ROOT, COMM, IERROR
6
7      MPI_SCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
8          <type> SENDBUF(*), RECVBUF(*)
9          INTEGER COUNT, DATATYPE, OP, COMM, IERROR
10
11     MPI_SCATTER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
12                ROOT, COMM, IERROR)
13         <type> SENDBUF(*), RECVBUF(*)
14         INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, IERROR
15
16     MPI_SCATTERV(SENDBUF, SENDCOUNTS, DISPLS, SENDTYPE, RECVBUF, RECVCOUNT,
17                  RECVTYPE, ROOT, COMM, IERROR)
18         <type> SENDBUF(*), RECVBUF(*)
19         INTEGER SENDCOUNTS(*), DISPLS(*), SENDTYPE, RECVCOUNT, RECVTYPE, ROOT,
20         COMM, IERROR

```

A.4.4 Groups, Contexts, Communicators, and Caching Fortran Bindings

```

22     MPI_COMM_COMPARE(COMM1, COMM2, RESULT, IERROR)
23         INTEGER COMM1, COMM2, RESULT, IERROR
24
25     MPI_COMM_CREATE(COMM, GROUP, NEWCOMM, IERROR)
26         INTEGER COMM, GROUP, NEWCOMM, IERROR
27
28     MPI_COMM_CREATE_GROUP(COMM, GROUP, TAG, NEWCOMM, IERROR)
29         INTEGER COMM, GROUP, TAG, NEWCOMM, IERROR
30
31     MPI_COMM_CREATE_KEYVAL(COMM_COPY_ATTR_FN, COMM_DELETE_ATTR_FN, COMM_KEYVAL,
32                             EXTRA_STATE, IERROR)
33         EXTERNAL COMM_COPY_ATTR_FN, COMM_DELETE_ATTR_FN
34         INTEGER COMM_KEYVAL, IERROR
35         INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
36
37     MPI_COMM_DELETE_ATTR(COMM, COMM_KEYVAL, IERROR)
38         INTEGER COMM, COMM_KEYVAL, IERROR
39
40     MPI_COMM_DUP(COMM, NEWCOMM, IERROR)
41         INTEGER COMM, NEWCOMM, IERROR
42
43     MPI_COMM_DUP_FN(OLDCOMM, COMM_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
44                    ATTRIBUTE_VAL_OUT, FLAG, IERROR)
45         INTEGER OLDCOMM, COMM_KEYVAL, IERROR
46         INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
47         ATTRIBUTE_VAL_OUT
48         LOGICAL FLAG
49
50     MPI_COMM_DUP_WITH_INFO(COMM, INFO, NEWCOMM, IERROR)
51         INTEGER COMM, INFO, NEWCOMM, IERROR

```


MPI_COMM_FREE(COMM, IERROR)	1
INTEGER COMM, IERROR	2
	3
MPI_COMM_FREE_KEYVAL(COMM_KEYVAL, IERROR)	4
INTEGER COMM_KEYVAL, IERROR	5
	6
MPI_COMM_GET_ATTR(COMM, COMM_KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)	7
INTEGER COMM, COMM_KEYVAL, IERROR	8
INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL	9
LOGICAL FLAG	10
	11
MPI_COMM_GET_INFO(COMM, INFO_USED, IERROR)	12
INTEGER COMM, INFO_USED, IERROR	13
	14
MPI_COMM_GET_NAME(COMM, COMM_NAME, RESULTLEN, IERROR)	15
INTEGER COMM, RESULTLEN, IERROR	16
CHARACTER*(*) COMM_NAME	17
	18
MPI_COMM_GROUP(COMM, GROUP, IERROR)	19
INTEGER COMM, GROUP, IERROR	20
	21
MPI_COMM_NULL_COPY_FN(OLDCOMM, COMM_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,	22
ATTRIBUTE_VAL_OUT, FLAG, IERROR)	23
INTEGER OLDCOMM, COMM_KEYVAL, IERROR	24
INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,	25
ATTRIBUTE_VAL_OUT	26
LOGICAL FLAG	27
	28
MPI_COMM_NULL_DELETE_FN(COMM, COMM_KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE,	29
IERROR)	30
INTEGER COMM, COMM_KEYVAL, IERROR	31
INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE	32
	33
MPI_COMM_RANK(COMM, RANK, IERROR)	34
INTEGER COMM, RANK, IERROR	35
	36
MPI_COMM_REMOTE_GROUP(COMM, GROUP, IERROR)	37
INTEGER COMM, GROUP, IERROR	38
	39
MPI_COMM_REMOTE_SIZE(COMM, SIZE, IERROR)	40
INTEGER COMM, SIZE, IERROR	41
	42
MPI_COMM_SET_ATTR(COMM, COMM_KEYVAL, ATTRIBUTE_VAL, IERROR)	43
INTEGER COMM, COMM_KEYVAL, IERROR	44
INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL	45
	46
MPI_COMM_SET_INFO(COMM, INFO, IERROR)	47
INTEGER COMM, INFO, IERROR	48
MPI_COMM_SET_NAME(COMM, COMM_NAME, IERROR)	
INTEGER COMM, IERROR	

```

1      CHARACTER*(*) COMM_NAME
2
3      MPI_COMM_SIZE(COMM, SIZE, IERROR)
4          INTEGER COMM, SIZE, IERROR
5
6      MPI_COMM_SPLIT(COMM, COLOR, KEY, NEWCOMM, IERROR)
7          INTEGER COMM, COLOR, KEY, NEWCOMM, IERROR
8
9      MPI_COMM_SPLIT_TYPE(COMM, SPLIT_TYPE, KEY, INFO, NEWCOMM, IERROR)
10         INTEGER COMM, SPLIT_TYPE, KEY, INFO, NEWCOMM, IERROR
11
12      MPI_COMM_TEST_INTER(COMM, FLAG, IERROR)
13         INTEGER COMM, IERROR
14         LOGICAL FLAG
15
16      MPI_GROUP_COMPARE(GROUP1, GROUP2, RESULT, IERROR)
17         INTEGER GROUP1, GROUP2, RESULT, IERROR
18
19      MPI_GROUP_DIFFERENCE(GROUP1, GROUP2, NEWGROUP, IERROR)
20         INTEGER GROUP1, GROUP2, NEWGROUP, IERROR
21
22      MPI_GROUP_EXCL(GROUP, N, RANKS, NEWGROUP, IERROR)
23         INTEGER GROUP, N, RANKS(*), NEWGROUP, IERROR
24
25      MPI_GROUP_FREE(GROUP, IERROR)
26         INTEGER GROUP, IERROR
27
28      MPI_GROUP_INCL(GROUP, N, RANKS, NEWGROUP, IERROR)
29         INTEGER GROUP, N, RANKS(*), NEWGROUP, IERROR
30
31      MPI_GROUP_INTERSECTION(GROUP1, GROUP2, NEWGROUP, IERROR)
32         INTEGER GROUP1, GROUP2, NEWGROUP, IERROR
33
34      MPI_GROUP_RANGE_EXCL(GROUP, N, RANGES, NEWGROUP, IERROR)
35         INTEGER GROUP, N, RANGES(3,*), NEWGROUP, IERROR
36
37      MPI_GROUP_RANGE_INCL(GROUP, N, RANGES, NEWGROUP, IERROR)
38         INTEGER GROUP, N, RANGES(3,*), NEWGROUP, IERROR
39
40      MPI_GROUP_RANK(GROUP, RANK, IERROR)
41         INTEGER GROUP, RANK, IERROR
42
43      MPI_GROUP_SIZE(GROUP, SIZE, IERROR)
44         INTEGER GROUP, SIZE, IERROR
45
46      MPI_GROUP_TRANSLATE_RANKS(GROUP1, N, RANKS1, GROUP2, RANKS2, IERROR)
47         INTEGER GROUP1, N, RANKS1(*), GROUP2, RANKS2(*), IERROR
48
49      MPI_GROUP_UNION(GROUP1, GROUP2, NEWGROUP, IERROR)
50         INTEGER GROUP1, GROUP2, NEWGROUP, IERROR
51
52      MPI_INTERCOMM_CREATE(LOCAL_COMM, LOCAL_LEADER, PEER_COMM, REMOTE_LEADER,
53                          TAG, NEWINTERCOMM, IERROR)
54         INTEGER LOCAL_COMM, LOCAL_LEADER, PEER_COMM, REMOTE_LEADER, TAG,
55         NEWINTERCOMM, IERROR

```

```

MPI_INTERCOMM_MERGE(INTERCOMM, HIGH, NEWINTRACOMM, IERROR)      1
    INTEGER INTERCOMM, NEWINTRACOMM, IERROR                      2
    LOGICAL HIGH                                                  3
                                                                    4
MPI_TYPE_CREATE_KEYVAL(TYPE_COPY_ATTR_FN, TYPE_DELETE_ATTR_FN, TYPE_KEYVAL,
    EXTRA_STATE, IERROR)                                         5
    EXTERNAL TYPE_COPY_ATTR_FN, TYPE_DELETE_ATTR_FN              6
    INTEGER TYPE_KEYVAL, IERROR                                   7
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE                   8
                                                                    9
MPI_TYPE_DELETE_ATTR(DATATYPE, TYPE_KEYVAL, IERROR)              10
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR                        11
                                                                    12
MPI_TYPE_DUP_FN(OLDTYPE, TYPE_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
    ATTRIBUTE_VAL_OUT, FLAG, IERROR)                              13
    INTEGER OLDTYPE, TYPE_KEYVAL, IERROR                         14
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN, 15
    ATTRIBUTE_VAL_OUT                                             16
    LOGICAL FLAG                                                  17
                                                                    18
MPI_TYPE_FREE_KEYVAL(TYPE_KEYVAL, IERROR)                         19
    INTEGER TYPE_KEYVAL, IERROR                                  20
                                                                    21
MPI_TYPE_GET_ATTR(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR) 22
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR                        23
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL                 24
    LOGICAL FLAG                                                  25
                                                                    26
MPI_TYPE_GET_NAME(DATATYPE, TYPE_NAME, RESULTLEN, IERROR)        27
    INTEGER DATATYPE, RESULTLEN, IERROR                          28
    CHARACTER*(*) TYPE_NAME                                       29
                                                                    30
MPI_TYPE_NULL_COPY_FN(OLDTYPE, TYPE_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
    ATTRIBUTE_VAL_OUT, FLAG, IERROR)                              31
    INTEGER OLDTYPE, TYPE_KEYVAL, IERROR                         32
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN, 33
    ATTRIBUTE_VAL_OUT                                             34
    LOGICAL FLAG                                                  35
                                                                    36
MPI_TYPE_NULL_DELETE_FN(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE,
    IERROR)                                                        37
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR                        38
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE    39
                                                                    40
MPI_TYPE_SET_ATTR(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL, IERROR)  41
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR                        42
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL                 43
                                                                    44
MPI_TYPE_SET_NAME(DATATYPE, TYPE_NAME, IERROR)                   45
    INTEGER DATATYPE, IERROR                                      46
    CHARACTER*(*) TYPE_NAME                                       47
                                                                    48
MPI_WIN_CREATE_KEYVAL(WIN_COPY_ATTR_FN, WIN_DELETE_ATTR_FN, WIN_KEYVAL,

```

```

1          EXTRA_STATE, IERROR)
2      EXTERNAL WIN_COPY_ATTR_FN, WIN_DELETE_ATTR_FN
3      INTEGER WIN_KEYVAL, IERROR
4      INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
5
6      MPI_WIN_DELETE_ATTR(WIN, WIN_KEYVAL, IERROR)
7          INTEGER WIN, WIN_KEYVAL, IERROR
8
9      MPI_WIN_DUP_FN(OLDWIN, WIN_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
10         ATTRIBUTE_VAL_OUT, FLAG, IERROR)
11          INTEGER OLDWIN, WIN_KEYVAL, IERROR
12          INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
13             ATTRIBUTE_VAL_OUT
14          LOGICAL FLAG
15
16      MPI_WIN_FREE_KEYVAL(WIN_KEYVAL, IERROR)
17          INTEGER WIN_KEYVAL, IERROR
18
19      MPI_WIN_GET_ATTR(WIN, WIN_KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)
20          INTEGER WIN, WIN_KEYVAL, IERROR
21          INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL
22          LOGICAL FLAG
23
24      MPI_WIN_GET_NAME(WIN, WIN_NAME, RESULTLEN, IERROR)
25          INTEGER WIN, RESULTLEN, IERROR
26          CHARACTER*(*) WIN_NAME
27
28      MPI_WIN_NULL_COPY_FN(OLDWIN, WIN_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
29         ATTRIBUTE_VAL_OUT, FLAG, IERROR)
30          INTEGER OLDWIN, WIN_KEYVAL, IERROR
31          INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
32             ATTRIBUTE_VAL_OUT
33          LOGICAL FLAG
34
35      MPI_WIN_NULL_DELETE_FN(WIN, WIN_KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERROR)
36          INTEGER WIN, WIN_KEYVAL, IERROR
37          INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE
38
39      MPI_WIN_SET_ATTR(WIN, WIN_KEYVAL, ATTRIBUTE_VAL, IERROR)
40          INTEGER WIN, WIN_KEYVAL, IERROR
41          INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL
42
43      MPI_WIN_SET_NAME(WIN, WIN_NAME, IERROR)
44          INTEGER WIN, IERROR
45          CHARACTER*(*) WIN_NAME

```

A.4.5 Process Topologies Fortran Bindings

```

45      MPI_CART_COORDS(COMM, RANK, MAXDIMS, COORDS, IERROR)
46          INTEGER COMM, RANK, MAXDIMS, COORDS(*), IERROR
47
48      MPI_CART_CREATE(COMM_OLD, NDIMS, DIMS, PERIODS, REORDER, COMM_CART, IERROR)

```

```

    INTEGER COMM_OLD, NDIMS, DIMS(*), COMM_CART, IERROR
    LOGICAL PERIODS(*), REORDER
1
2
3
MPI_CARTDIM_GET(COMM, NDIMS, IERROR)
4
    INTEGER COMM, NDIMS, IERROR
5
MPI_CART_GET(COMM, MAXDIMS, DIMS, PERIODS, COORDS, IERROR)
6
    INTEGER COMM, MAXDIMS, DIMS(*), COORDS(*), IERROR
7
    LOGICAL PERIODS(*)
8
9
MPI_CART_MAP(COMM, NDIMS, DIMS, PERIODS, NEWRANK, IERROR)
10
    INTEGER COMM, NDIMS, DIMS(*), NEWRANK, IERROR
11
    LOGICAL PERIODS(*)
12
13
MPI_CART_RANK(COMM, COORDS, RANK, IERROR)
14
    INTEGER COMM, COORDS(*), RANK, IERROR
15
MPI_CART_SHIFT(COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST, IERROR)
16
    INTEGER COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST, IERROR
17
18
MPI_CART_SUB(COMM, REMAIN_DIMS, NEWCOMM, IERROR)
19
    INTEGER COMM, NEWCOMM, IERROR
20
    LOGICAL REMAIN_DIMS(*)
21
22
MPI_DIMS_CREATE(NNODES, NDIMS, DIMS, IERROR)
23
    INTEGER NNODES, NDIMS, DIMS(*), IERROR
24
25
MPI_DIST_GRAPH_CREATE_ADJACENT(COMM_OLD, INDEGREE, SOURCES, SOURCEWEIGHTS,
    OUTDEGREE, DESTINATIONS, DESTWEIGHTS, INFO, REORDER,
26
    COMM_DIST_GRAPH, IERROR)
27
    INTEGER COMM_OLD, INDEGREE, SOURCES(*), SOURCEWEIGHTS(*), OUTDEGREE,
28
    DESTINATIONS(*), DESTWEIGHTS(*), INFO, COMM_DIST_GRAPH, IERROR
29
    LOGICAL REORDER
30
31
MPI_DIST_GRAPH_CREATE(COMM_OLD, N, SOURCES, DEGREES, DESTINATIONS, WEIGHTS,
    INFO, REORDER, COMM_DIST_GRAPH, IERROR)
32
    INTEGER COMM_OLD, N, SOURCES(*), DEGREES(*), DESTINATIONS(*),
33
    WEIGHTS(*), INFO, COMM_DIST_GRAPH, IERROR
34
    LOGICAL REORDER
35
36
MPI_DIST_GRAPH_NEIGHBORS(COMM, MAXINDEGREE, SOURCES, SOURCEWEIGHTS,
    MAXOUTDEGREE, DESTINATIONS, DESTWEIGHTS, IERROR)
37
    INTEGER COMM, MAXINDEGREE, SOURCES(*), SOURCEWEIGHTS(*), MAXOUTDEGREE,
38
    DESTINATIONS(*), DESTWEIGHTS(*), IERROR
39
40
MPI_DIST_GRAPH_NEIGHBORS_COUNT(COMM, INDEGREE, OUTDEGREE, WEIGHTED, IERROR)
41
    INTEGER COMM, INDEGREE, OUTDEGREE, IERROR
42
    LOGICAL WEIGHTED
43
44
MPI_GRAPH_CREATE(COMM_OLD, NNODES, INDEX, EDGES, REORDER, COMM_GRAPH,
    IERROR)
45
    INTEGER COMM_OLD, NNODES, INDEX(*), EDGES(*), COMM_GRAPH, IERROR
46
    LOGICAL REORDER
47
48

```

```

1  MPI_GRAPHDIMS_GET(COMM, NNODES, NEDGES, IERROR)
2      INTEGER COMM, NNODES, NEDGES, IERROR
3
4  MPI_GRAPH_GET(COMM, MAXINDEX, MAXEDGES, INDEX, EDGES, IERROR)
5      INTEGER COMM, MAXINDEX, MAXEDGES, INDEX(*), EDGES(*), IERROR
6
7  MPI_GRAPH_MAP(COMM, NNODES, INDEX, EDGES, NEWRANK, IERROR)
8      INTEGER COMM, NNODES, INDEX(*), EDGES(*), NEWRANK, IERROR
9
10 MPI_GRAPH_NEIGHBORS(COMM, RANK, MAXNEIGHBORS, NEIGHBORS, IERROR)
11     INTEGER COMM, RANK, MAXNEIGHBORS, NEIGHBORS(*), IERROR
12
13 MPI_GRAPH_NEIGHBORS_COUNT(COMM, RANK, NNEIGHBORS, IERROR)
14     INTEGER COMM, RANK, NNEIGHBORS, IERROR
15
16 MPI_INEIGHBOR_ALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
17     RECVTYPE, COMM, REQUEST, IERROR)
18     <type> SENDBUF(*), RECVBUF(*)
19     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR
20
21 MPI_INEIGHBOR_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS,
22     DISPLS, RECVTYPE, COMM, REQUEST, IERROR)
23     <type> SENDBUF(*), RECVBUF(*)
24     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
25     REQUEST, IERROR
26
27 MPI_INEIGHBOR_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
28     RECVTYPE, COMM, REQUEST, IERROR)
29     <type> SENDBUF(*), RECVBUF(*)
30     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR
31
32 MPI_INEIGHBOR_ALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF,
33     RECVCOUNTS, RDISPLS, RECVTYPE, COMM, REQUEST, IERROR)
34     <type> SENDBUF(*), RECVBUF(*)
35     INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
36     RECVTYPE, COMM, REQUEST, IERROR
37
38 MPI_INEIGHBOR_ALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF,
39     RECVCOUNTS, RDISPLS, RECVTYPES, COMM, REQUEST, IERROR)
40     <type> SENDBUF(*), RECVBUF(*)
41     INTEGER(KIND=MPI_ADDRESS_KIND) SDISPLS(*), RDISPLS(*)
42     INTEGER SENDCOUNTS(*), SENDTYPES(*), RECVCOUNTS(*), RECVTYPES(*), COMM,
43     REQUEST, IERROR
44
45 MPI_NEIGHBOR_ALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
46     RECVTYPE, COMM, IERROR)
47     <type> SENDBUF(*), RECVBUF(*)
48     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR
49
50 MPI_NEIGHBOR_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS,
51     DISPLS, RECVTYPE, COMM, IERROR)
52     <type> SENDBUF(*), RECVBUF(*)
53     INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
54     REQUEST, IERROR

```

```

IERROR
1
2
MPI_NEIGHBOR_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, REVCOUNT,
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
MPI_NEIGHBOR_ALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF,
REVCOUNTS, RDISPLS, RECVTYPE, COMM, IERROR)
MPI_NEIGHBOR_ALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF,
REVCOUNTS, RDISPLS, RECVTYPES, COMM, IERROR)
MPI_TOPO_TEST(COMM, STATUS, IERROR)
INTEGER COMM, STATUS, IERROR

```

A.4.6 MPI Environmental Management Fortran Bindings

```

DOUBLE PRECISION MPI_WTICK()
DOUBLE PRECISION MPI_WTIME()
MPI_ABORT(COMM, ERRORCODE, IERROR)
INTEGER COMM, ERRORCODE, IERROR
MPI_ADD_ERROR_CLASS(ERRORCLASS, IERROR)
INTEGER ERRORCLASS, IERROR
MPI_ADD_ERROR_CODE(ERRORCLASS, ERRORCODE, IERROR)
INTEGER ERRORCLASS, ERRORCODE, IERROR
MPI_ADD_ERROR_STRING(ERRORCODE, STRING, IERROR)
INTEGER ERRORCODE, IERROR
CHARACTER*(*) STRING
MPI_ALLOC_MEM(SIZE, INFO, BASEPTR, IERROR)
INTEGER INFO, IERROR
INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR

```

If the Fortran compiler provides TYPE(C_PTR), then overloaded by:

```

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

```

```
1      END SUBROUTINE
2      END INTERFACE
3
4      MPI_COMM_CALL_ERRHANDLER(COMM, ERRORCODE, IERROR)
5          INTEGER COMM, ERRORCODE, IERROR
6
7      MPI_COMM_CREATE_ERRHANDLER(COMM_ERRHANDLER_FN, ERRHANDLER, IERROR)
8          EXTERNAL COMM_ERRHANDLER_FN
9          INTEGER ERRHANDLER, IERROR
10
11     MPI_COMM_GET_ERRHANDLER(COMM, ERRHANDLER, IERROR)
12         INTEGER COMM, ERRHANDLER, IERROR
13
14     MPI_COMM_SET_ERRHANDLER(COMM, ERRHANDLER, IERROR)
15         INTEGER COMM, ERRHANDLER, IERROR
16
17     MPI_ERRHANDLER_FREE(ERRHANDLER, IERROR)
18         INTEGER ERRHANDLER, IERROR
19
20     MPI_ERROR_CLASS(ERRORCODE, ERRORCLASS, IERROR)
21         INTEGER ERRORCODE, ERRORCLASS, IERROR
22
23     MPI_ERROR_STRING(ERRORCODE, STRING, RESULTLEN, IERROR)
24         INTEGER ERRORCODE, RESULTLEN, IERROR
25         CHARACTER*(*) STRING
26
27     MPI_FILE_CALL_ERRHANDLER(FH, ERRORCODE, IERROR)
28         INTEGER FH, ERRORCODE, IERROR
29
30     MPI_FILE_CREATE_ERRHANDLER(FILE_ERRHANDLER_FN, ERRHANDLER, IERROR)
31         EXTERNAL FILE_ERRHANDLER_FN
32         INTEGER ERRHANDLER, IERROR
33
34     MPI_FILE_GET_ERRHANDLER(FILE, ERRHANDLER, IERROR)
35         INTEGER FILE, ERRHANDLER, IERROR
36
37     MPI_FILE_SET_ERRHANDLER(FILE, ERRHANDLER, IERROR)
38         INTEGER FILE, ERRHANDLER, IERROR
39
40     MPI_FINALIZED(FLAG, IERROR)
41         LOGICAL FLAG
42         INTEGER IERROR
43
44     MPI_FINALIZE(IERROR)
45         INTEGER IERROR
46
47     MPI_FREE_MEM(BASE, IERROR)
48         <type> BASE(*)
49         INTEGER IERROR
50
51     MPI_GET_LIBRARY_VERSION(VERSION, RESULTLEN, IERROR)
52         CHARACTER*(*) VERSION
53         INTEGER RESULTLEN, IERROR
54
55     MPI_GET_PROCESSOR_NAME( NAME, RESULTLEN, IERROR)
```



```

    CHARACTER*(*) NAME                                1
    INTEGER RESULTLEN, IERROR                          2
                                                    3
MPI_GET_VERSION(VERSION, SUBVERSION, IERROR)          4
    INTEGER VERSION, SUBVERSION, IERROR                5
                                                    6
MPI_INITIALIZED(FLAG, IERROR)                          7
    LOGICAL FLAG                                       8
    INTEGER IERROR                                    9
                                                    10
MPI_INIT(IERROR)                                       11
    INTEGER IERROR                                    12
                                                    13
MPI_WIN_CALL_ERRHANDLER(WIN, ERRORCODE, IERROR)       14
    INTEGER WIN, ERRORCODE, IERROR                    15
                                                    16
MPI_WIN_CREATE_ERRHANDLER(WIN_ERRHANDLER_FN, ERRHANDLER, IERROR) 17
    EXTERNAL WIN_ERRHANDLER_FN                       18
    INTEGER ERRHANDLER, IERROR                        19
                                                    20
MPI_WIN_GET_ERRHANDLER(WIN, ERRHANDLER, IERROR)       21
    INTEGER WIN, ERRHANDLER, IERROR                   22
                                                    23
MPI_WIN_SET_ERRHANDLER(WIN, ERRHANDLER, IERROR)       24
    INTEGER WIN, ERRHANDLER, IERROR                   25

```

A.4.7 The Info Object Fortran Bindings

```

MPI_INFO_CREATE(INFO, IERROR)                          26
    INTEGER INFO, IERROR                              27
                                                    28
MPI_INFO_DELETE(INFO, KEY, IERROR)                     29
    INTEGER INFO, IERROR                              30
    CHARACTER*(*) KEY                                31
                                                    32
MPI_INFO_DUP(INFO, NEWINFO, IERROR)                   33
    INTEGER INFO, NEWINFO, IERROR                     34
                                                    35
MPI_INFO_FREE(INFO, IERROR)                            36
    INTEGER INFO, IERROR                              37
                                                    38
MPI_INFO_GET(INFO, KEY, VALUELEN, VALUE, FLAG, IERROR) 39
    INTEGER INFO, VALUELEN, IERROR                    40
    CHARACTER*(*) KEY, VALUE                          41
    LOGICAL FLAG                                       42
                                                    43
MPI_INFO_GET_NKEYS(INFO, NKEYS, IERROR)                44
    INTEGER INFO, NKEYS, IERROR                       45
                                                    46
MPI_INFO_GET_NTHKEY(INFO, N, KEY, IERROR)              47
    INTEGER INFO, N, IERROR                           48
    CHARACTER*(*) KEY
MPI_INFO_GET_VALUELEN(INFO, KEY, VALUELEN, FLAG, IERROR)

```

```

1      INTEGER INFO, VALUELEN, IERROR
2      LOGICAL FLAG
3      CHARACTER*(*) KEY
4
5      MPI_INFO_SET(INFO, KEY, VALUE, IERROR)
6      INTEGER INFO, IERROR
7      CHARACTER*(*) KEY, VALUE
8

```

A.4.8 Process Creation and Management Fortran Bindings

```

9
10
11     MPI_CLOSE_PORT(PORT_NAME, IERROR)
12         CHARACTER*(*) PORT_NAME
13         INTEGER IERROR
14
15     MPI_COMM_ACCEPT(PORT_NAME, INFO, ROOT, COMM, NEWCOMM, IERROR)
16         CHARACTER*(*) PORT_NAME
17         INTEGER INFO, ROOT, COMM, NEWCOMM, IERROR
18
19     MPI_COMM_CONNECT(PORT_NAME, INFO, ROOT, COMM, NEWCOMM, IERROR)
20         CHARACTER*(*) PORT_NAME
21         INTEGER INFO, ROOT, COMM, NEWCOMM, IERROR
22
23     MPI_COMM_DISCONNECT(COMM, IERROR)
24         INTEGER COMM, IERROR
25
26     MPI_COMM_GET_PARENT(PARENT, IERROR)
27         INTEGER PARENT, IERROR
28
29     MPI_COMM_JOIN(FD, INTERCOMM, IERROR)
30         INTEGER FD, INTERCOMM, IERROR
31
32     MPI_COMM_SPAWN(COMMAND, ARGV, MAXPROCS, INFO, ROOT, COMM, INTERCOMM,
33         ARRAY_OF_ERRCODES, IERROR)
34         CHARACTER*(*) COMMAND, ARGV(*)
35         INTEGER INFO, MAXPROCS, ROOT, COMM, INTERCOMM, ARRAY_OF_ERRCODES(*),
36         IERROR
37
38     MPI_COMM_SPAWN_MULTIPLE(COUNT, ARRAY_OF_COMMANDS, ARRAY_OF_ARGV,
39         ARRAY_OF_MAXPROCS, ARRAY_OF_INFO, ROOT, COMM, INTERCOMM,
40         ARRAY_OF_ERRCODES, IERROR)
41         INTEGER COUNT, ARRAY_OF_INFO(*), ARRAY_OF_MAXPROCS(*), ROOT, COMM,
42         INTERCOMM, ARRAY_OF_ERRCODES(*), IERROR
43         CHARACTER*(*) ARRAY_OF_COMMANDS(*), ARRAY_OF_ARGV(COUNT, *)
44
45     MPI_LOOKUP_NAME(SERVICE_NAME, INFO, PORT_NAME, IERROR)
46         CHARACTER*(*) SERVICE_NAME, PORT_NAME
47         INTEGER INFO, IERROR
48
49     MPI_OPEN_PORT(INFO, PORT_NAME, IERROR)
50         CHARACTER*(*) PORT_NAME
51         INTEGER INFO, IERROR
52
53     MPI_PUBLISH_NAME(SERVICE_NAME, INFO, PORT_NAME, IERROR)

```

```

    INTEGER INFO, IERROR
    CHARACTER*(*) SERVICE_NAME, PORT_NAME
MPI_UNPUBLISH_NAME(SERVICE_NAME, INFO, PORT_NAME, IERROR)
    INTEGER INFO, IERROR
    CHARACTER*(*) SERVICE_NAME, PORT_NAME

A.4.9 One-Sided Communications Fortran Bindings

MPI_ACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
               TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, OP, WIN, IERROR

MPI_COMPARE_AND_SWAP(ORIGIN_ADDR, COMPARE_ADDR, RESULT_ADDR, DATATYPE,
                    TARGET_RANK, TARGET_DISP, WIN, IERROR)
    <type> ORIGIN_ADDR(*), COMPARE_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER DATATYPE, TARGET_RANK, WIN, IERROR

MPI_FETCH_AND_OP(ORIGIN_ADDR, RESULT_ADDR, DATATYPE, TARGET_RANK,
                TARGET_DISP, OP, WIN, IERROR)
    <type> ORIGIN_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER DATATYPE, TARGET_RANK, OP, WIN, IERROR

MPI_GET_ACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, RESULT_ADDR,
                  RESULT_COUNT, RESULT_DATATYPE, TARGET_RANK, TARGET_DISP,
                  TARGET_COUNT, TARGET_DATATYPE, OP, WIN, IERROR)
    <type> ORIGIN_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, RESULT_COUNT, RESULT_DATATYPE,
    TARGET_RANK, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, IERROR

MPI_GET(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
        TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, WIN, IERROR

MPI_PUT(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
        TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, WIN, IERROR

MPI_RACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,

```

```

1          TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, REQUEST,
2          IERROR)
3      <type> ORIGIN_ADDR(*)
4      INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
5      INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
6      TARGET_DATATYPE, OP, WIN, REQUEST, IERROR
7
8      MPI_RGET_ACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE,
9          RESULT_ADDR, RESULT_COUNT, RESULT_DATATYPE, TARGET_RANK,
10         TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, REQUEST,
11         IERROR)
12      <type> ORIGIN_ADDR(*), RESULT_ADDR(*)
13      INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
14      INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, RESULT_COUNT, RESULT_DATATYPE,
15      TARGET_RANK, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, REQUEST, IERROR
16
17      MPI_RGET(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
18          TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, REQUEST,
19          IERROR)
20      <type> ORIGIN_ADDR(*)
21      INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
22      INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
23      TARGET_DATATYPE, WIN, REQUEST, IERROR
24
25      MPI_RPUT(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
26          TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, REQUEST,
27          IERROR)
28      <type> ORIGIN_ADDR(*)
29      INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
30      INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
31      TARGET_DATATYPE, WIN, REQUEST, IERROR
32
33      MPI_WIN_ALLOCATE_SHARED(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
34      INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
35      INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
36
37      If the Fortran compiler provides TYPE(C_PTR), then overloaded by:
38      INTERFACE MPI_WIN_ALLOCATE_SHARED
39          SUBROUTINE MPI_WIN_ALLOCATE_SHARED_CPTR(SIZE, DISP_UNIT, INFO, COMM, &
40              BASEPTR, WIN, IERROR)
41              USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
42              INTEGER :: DISP_UNIT, INFO, COMM, WIN, IERROR
43              INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
44              TYPE(C_PTR) :: BASEPTR
45          END SUBROUTINE
46      END INTERFACE
47
48      MPI_WIN_ALLOCATE(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
49      INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
50      INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
51
52      If the Fortran compiler provides TYPE(C_PTR), then overloaded by:

```

```

INTERFACE MPI_WIN_ALLOCATE                                1
  SUBROUTINE MPI_WIN_ALLOCATE_CPTR(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, & 2
    WIN, IERROR)                                          3
    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR        4
    INTEGER :: DISP_UNIT, INFO, COMM, WIN, IERROR        5
    INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE              6
    TYPE(C_PTR) :: BASEPTR                              7
  END SUBROUTINE                                          8
END INTERFACE                                            9
                                                         10
MPI_WIN_ATTACH(WIN, BASE, SIZE, IERROR)                 11
  INTEGER WIN, IERROR                                    12
  <type> BASE(*)                                         13
  INTEGER (KIND=MPI_ADDRESS_KIND) SIZE                  14
                                                         15
MPI_WIN_COMPLETE(WIN, IERROR)                           16
  INTEGER WIN, IERROR                                   17
                                                         18
MPI_WIN_CREATE(BASE, SIZE, DISP_UNIT, INFO, COMM, WIN, IERROR) 19
  <type> BASE(*)                                         20
  INTEGER(KIND=MPI_ADDRESS_KIND) SIZE                  21
  INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR           22
                                                         23
MPI_WIN_CREATE_DYNAMIC(INFO, COMM, WIN, IERROR)          24
  INTEGER INFO, COMM, WIN, IERROR                     25
                                                         26
MPI_WIN_DETACH(WIN, BASE, IERROR)                       27
  INTEGER WIN, IERROR                                   28
  <type> BASE(*)                                         29
                                                         30
MPI_WIN_FENCE(ASSERT, WIN, IERROR)                      31
  INTEGER ASSERT, WIN, IERROR                          32
                                                         33
MPI_WIN_FLUSH_ALL(WIN, IERROR)                          34
  INTEGER WIN, IERROR                                  35
                                                         36
MPI_WIN_FLUSH_LOCAL_ALL(WIN, IERROR)                    37
  INTEGER WIN, IERROR                                  38
                                                         39
MPI_WIN_FLUSH_LOCAL(RANK, WIN, IERROR)                   40
  INTEGER RANK, WIN, IERROR                            41
                                                         42
MPI_WIN_FLUSH(RANK, WIN, IERROR)                        43
  INTEGER RANK, WIN, IERROR                            44
                                                         45
MPI_WIN_FREE(WIN, IERROR)                               46
  INTEGER WIN, IERROR                                  47
                                                         48
MPI_WIN_GET_GROUP(WIN, GROUP, IERROR)                   48
  INTEGER WIN, GROUP, IERROR

```

```

1  MPI_WIN_LOCK_ALL(ASSERT, WIN, IERROR)
2      INTEGER ASSERT, WIN, IERROR
3
4  MPI_WIN_LOCK(LOCK_TYPE, RANK, ASSERT, WIN, IERROR)
5      INTEGER LOCK_TYPE, RANK, ASSERT, WIN, IERROR
6
7  MPI_WIN_POST(GROUP, ASSERT, WIN, IERROR)
8      INTEGER GROUP, ASSERT, WIN, IERROR
9
10 MPI_WIN_SET_INFO(WIN, INFO, IERROR)
11     INTEGER WIN, INFO, IERROR
12
13 MPI_WIN_SHARED_QUERY(WIN, RANK, SIZE, DISP_UNIT, BASEPTR, IERROR)
14     INTEGER WIN, RANK, DISP_UNIT, IERROR
15     INTEGER (KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
16
17 If the Fortran compiler provides TYPE(C_PTR), then overloaded by:
18
19 INTERFACE MPI_WIN_SHARED_QUERY
20     SUBROUTINE MPI_WIN_SHARED_QUERY_CPTR(WIN, RANK, SIZE, DISP_UNIT, &
21         BASEPTR, IERROR)
22         USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
23         INTEGER :: WIN, RANK, DISP_UNIT, IERROR
24         INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
25         TYPE(C_PTR) :: BASEPTR
26     END SUBROUTINE
27 END INTERFACE
28
29 MPI_WIN_START(GROUP, ASSERT, WIN, IERROR)
30     INTEGER GROUP, ASSERT, WIN, IERROR
31
32 MPI_WIN_SYNC(WIN, IERROR)
33     INTEGER WIN, IERROR
34
35 MPI_WIN_TEST(WIN, FLAG, IERROR)
36     INTEGER WIN, IERROR
37     LOGICAL FLAG
38
39 MPI_WIN_UNLOCK_ALL(WIN, IERROR)
40     INTEGER WIN, IERROR
41
42 MPI_WIN_UNLOCK(RANK, WIN, IERROR)
43     INTEGER RANK, WIN, IERROR
44
45 MPI_WIN_WAIT(WIN, IERROR)
46     INTEGER WIN, IERROR
47
48
49 A.4.10 External Interfaces Fortran Bindings
50
51 MPI_GREQUEST_COMPLETE(REQUEST, IERROR)
52     INTEGER REQUEST, IERROR
53
54 MPI_GREQUEST_START(QUERY_FN, FREE_FN, CANCEL_FN, EXTRA_STATE, REQUEST,
55     IERROR)

```

```

    INTEGER REQUEST, IERROR
    EXTERNAL QUERY_FN, FREE_FN, CANCEL_FN
    INTEGER (KIND=MPI_ADDRESS_KIND) EXTRA_STATE
MPI_INIT_THREAD(REQUIRED, PROVIDED, IERROR)
    INTEGER REQUIRED, PROVIDED, IERROR
MPI_IS_THREAD_MAIN(FLAG, IERROR)
    LOGICAL FLAG
    INTEGER IERROR
MPI_QUERY_THREAD(PROVIDED, IERROR)
    INTEGER PROVIDED, IERROR
MPI_STATUS_SET_CANCELLED(STATUS, FLAG, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), IERROR
    LOGICAL FLAG
MPI_STATUS_SET_ELEMENTS(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, COUNT, IERROR
MPI_STATUS_SET_ELEMENTS_X(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, IERROR
    INTEGER (KIND=MPI_COUNT_KIND) COUNT

```

A.4.11 I/O Fortran Bindings

```

MPI_CONVERSION_FN_NULL(USERBUF, DATATYPE, COUNT, FILEBUF, POSITION,
    EXTRA_STATE, IERROR)
    <TYPE> USERBUF(*), FILEBUF(*)
    INTEGER COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) POSITION
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
MPI_FILE_CLOSE(FH, IERROR)
    INTEGER FH, IERROR
MPI_FILE_DELETE(FILENAME, INFO, IERROR)
    CHARACTER*(*) FILENAME
    INTEGER INFO, IERROR
MPI_FILE_GET_AMODE(FH, AMODE, IERROR)
    INTEGER FH, AMODE, IERROR
MPI_FILE_GET_ATOMICITY(FH, FLAG, IERROR)
    INTEGER FH, IERROR
    LOGICAL FLAG
MPI_FILE_GET_BYTE_OFFSET(FH, OFFSET, DISP, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET, DISP
MPI_FILE_GET_GROUP(FH, GROUP, IERROR)

```

```

1      INTEGER FH, GROUP, IERROR
2
3      MPI_FILE_GET_INFO(FH, INFO_USED, IERROR)
4      INTEGER FH, INFO_USED, IERROR
5
6      MPI_FILE_GET_POSITION(FH, OFFSET, IERROR)
7      INTEGER FH, IERROR
8      INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
9
10     MPI_FILE_GET_POSITION_SHARED(FH, OFFSET, IERROR)
11     INTEGER FH, IERROR
12     INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
13
14     MPI_FILE_GET_SIZE(FH, SIZE, IERROR)
15     INTEGER FH, IERROR
16     INTEGER(KIND=MPI_OFFSET_KIND) SIZE
17
18     MPI_FILE_GET_TYPE_EXTENT(FH, DATATYPE, EXTENT, IERROR)
19     INTEGER FH, DATATYPE, IERROR
20     INTEGER(KIND=MPI_ADDRESS_KIND) EXTENT
21
22     MPI_FILE_GET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, IERROR)
23     INTEGER FH, ETYPE, FILETYPE, IERROR
24     CHARACTER*(*) DATAREP
25     INTEGER(KIND=MPI_OFFSET_KIND) DISP
26
27     MPI_FILE_IREAD_ALL(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
28     <type> BUF(*)
29     INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
30
31     MPI_FILE_IREAD_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
32     <type> BUF(*)
33     INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
34     INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
35
36     MPI_FILE_IREAD_AT(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
37     <type> BUF(*)
38     INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
39     INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
40
41     MPI_FILE_IREAD(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
42     <type> BUF(*)
43     INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
44
45     MPI_FILE_IREAD_SHARED(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
46     <type> BUF(*)
47     INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
48
49     MPI_FILE_IWRITE_ALL(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
50     <type> BUF(*)
51     INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
52
53     MPI_FILE_IWRITE_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
54     <type> BUF(*)

```



```

    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_IWRITE_AT(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_IWRITE(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
MPI_FILE_IWRITE_SHARED(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
MPI_FILE_OPEN(COMM, FILENAME, AMODE, INFO, FH, IERROR)
    CHARACTER*(*) FILENAME
    INTEGER COMM, AMODE, INFO, FH, IERROR
MPI_FILE_PREALLOCATE(FH, SIZE, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) SIZE
MPI_FILE_READ_ALL_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR
MPI_FILE_READ_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_READ_ALL(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_READ_AT_ALL_BEGIN(FH, OFFSET, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_READ_AT_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_READ_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_READ_AT(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

```

```

1      INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
2
3      MPI_FILE_READ(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
4          <type> BUF(*)
5          INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
6
7      MPI_FILE_READ_ORDERED_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
8          <type> BUF(*)
9          INTEGER FH, COUNT, DATATYPE, IERROR
10
11     MPI_FILE_READ_ORDERED_END(FH, BUF, STATUS, IERROR)
12         <type> BUF(*)
13         INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR
14
15     MPI_FILE_READ_ORDERED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
16         <type> BUF(*)
17         INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
18
19     MPI_FILE_READ_SHARED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
20         <type> BUF(*)
21         INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
22
23     MPI_FILE_SEEK(FH, OFFSET, WHENCE, IERROR)
24         INTEGER FH, WHENCE, IERROR
25         INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
26
27     MPI_FILE_SEEK_SHARED(FH, OFFSET, WHENCE, IERROR)
28         INTEGER FH, WHENCE, IERROR
29         INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
30
31     MPI_FILE_SET_ATOMICITY(FH, FLAG, IERROR)
32         INTEGER FH, IERROR
33         LOGICAL FLAG
34
35     MPI_FILE_SET_INFO(FH, INFO, IERROR)
36         INTEGER FH, INFO, IERROR
37
38     MPI_FILE_SET_SIZE(FH, SIZE, IERROR)
39         INTEGER FH, IERROR
40         INTEGER(KIND=MPI_OFFSET_KIND) SIZE
41
42     MPI_FILE_SET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, INFO, IERROR)
43         INTEGER FH, ETYPE, FILETYPE, INFO, IERROR
44         CHARACTER*(*) DATAREP
45         INTEGER(KIND=MPI_OFFSET_KIND) DISP
46
47     MPI_FILE_SYNC(FH, IERROR)
48         INTEGER FH, IERROR
49
50     MPI_FILE_WRITE_ALL_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
51         <type> BUF(*)
52         INTEGER FH, COUNT, DATATYPE, IERROR
53
54     MPI_FILE_WRITE_ALL_END(FH, BUF, STATUS, IERROR)

```

```

    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_ALL(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_AT_ALL_BEGIN(FH, OFFSET, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_WRITE_AT_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_WRITE_AT(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET
MPI_FILE_WRITE(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_ORDERED_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR
MPI_FILE_WRITE_ORDERED_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_ORDERED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_SHARED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_REGISTER_DATAREP(DATAREP, READ_CONVERSION_FN, WRITE_CONVERSION_FN,
    DTYPE_FILE_EXTENT_FN, EXTRA_STATE, IERROR)
    CHARACTER*(*) DATAREP
    EXTERNAL READ_CONVERSION_FN, WRITE_CONVERSION_FN, DTYPE_FILE_EXTENT_FN
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
    INTEGER IERROR

```

A.4.12 Language Bindings Fortran Bindings

```

MPI_F_SYNC_REG(buf)
    <type> buf(*)

MPI_SIZEOF(X, SIZE, IERROR)
    <type> X
    INTEGER SIZE, IERROR

MPI_STATUS_F082F(F08_STATUS, F_STATUS, IERROR)
    TYPE(MPI_Status) :: F08_STATUS
    INTEGER :: F_STATUS(MPI_STATUS_SIZE)
    INTEGER IERROR

MPI_STATUS_F2F08(F_STATUS, F08_STATUS, IERROR)
    INTEGER :: F_STATUS(MPI_STATUS_SIZE)
    TYPE(MPI_Status) :: F08_STATUS
    INTEGER IERROR

MPI_TYPE_CREATE_F90_COMPLEX(P, R, NEWTYPE, IERROR)
    INTEGER P, R, NEWTYPE, IERROR

MPI_TYPE_CREATE_F90_INTEGER(R, NEWTYPE, IERROR)
    INTEGER R, NEWTYPE, IERROR

MPI_TYPE_CREATE_F90_REAL(P, R, NEWTYPE, IERROR)
    INTEGER P, R, NEWTYPE, IERROR

MPI_TYPE_MATCH_SIZE(TYPECLASS, SIZE, DATATYPE, IERROR)
    INTEGER TYPECLASS, SIZE, DATATYPE, IERROR

```

A.4.13 Tools / Profiling Interface Fortran Bindings

```

MPI_PCONTROL(LEVEL)
    INTEGER LEVEL

```

A.4.14 Deprecated Fortran Bindings

```

MPI_ATTR_DELETE(COMM, KEYVAL, IERROR)
    INTEGER COMM, KEYVAL, IERROR

MPI_ATTR_GET(COMM, KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, IERROR
    LOGICAL FLAG

MPI_ATTR_PUT(COMM, KEYVAL, ATTRIBUTE_VAL, IERROR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, IERROR

MPI_DUP_FN(OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
           ATTRIBUTE_VAL_OUT, FLAG, IERR)
    INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
    ATTRIBUTE_VAL_OUT, IERR

```

```
LOGICAL FLAG
MPI_KEYVAL_CREATE(COPY_FN, DELETE_FN, KEYVAL, EXTRA_STATE, IERROR)
EXTERNAL COPY_FN, DELETE_FN
INTEGER KEYVAL, EXTRA_STATE, IERROR
MPI_KEYVAL_FREE(KEYVAL, IERROR)
INTEGER KEYVAL, IERROR
MPI_NULL_COPY_FN(OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
ATTRIBUTE_VAL_OUT, FLAG, IERR)
INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
ATTRIBUTE_VAL_OUT, IERR
LOGICAL FLAG
MPI_NULL_DELETE_FN(COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERROR)
INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERROR
SUBROUTINE COPY_FUNCTION(OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
ATTRIBUTE_VAL_OUT, FLAG, IERR)
INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
ATTRIBUTE_VAL_OUT, IERR
LOGICAL FLAG
SUBROUTINE DELETE_FUNCTION(COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR)
INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48