Point to Point Communication Routines: MPI Message Passing Routine Arguments

MPI point-to-point communication routines generally have an argument list that takes one of the following formats:

Non-blocking sends	**MPI_Isend(buffer,count,type,dest,tag,comm,request)**
Blocking receive	**MPI_Recv(buffer,count,type,source,tag,comm,status)**
Non-blocking receive	**MPI_Irecv(buffer,count,type,source,tag,comm,request)**

Buffer

Program (application) address space that references the data that is to be sent or received. In most cases, this is simply the variable name that is be sent/received. For C programs, this argument is passed by reference and usually must be prepended with an ampersand: `&var1`

Data Count

Indicates the number of data elements of a particular type to be sent.

Data Type

For reasons of portability, MPI predefines its elementary data types. The table below lists those required by the standard.

MPI_CHAR	char	**MPI_CHARACTER**	character(1)
MPI_WCHAR	wchar_t - wide character		
MPI_SHORT	signed short int		
MPI_INT	signed int	**MPI_INTEGER** **MPI_INTEGER1** **MPI_INTEGER2** **MPI_INTEGER4**	integer integer*1 integer*2 integer*4
MPI_LONG	signed long int		
MPI_LONG_LONG_INT **MPI_LONG_LONG**	signed long long int		
MPI_SIGNED_CHAR	signed char		
MPI_UNSIGNED_CHAR	unsigned char		
MPI_UNSIGNED_SHORT	unsigned short int		
MPI_UNSIGNED	unsigned int		
MPI_UNSIGNED_LONG	unsigned long int		
MPI_UNSIGNED_LONG_LONG	unsigned long long int		
MPI_FLOAT	float	**MPI_REAL** **MPI_REAL2** **MPI_REAL4** **MPI_REAL8**	real real*2 real*4 real*8
MPI_DOUBLE	double	**MPI_DOUBLE_PRECISION**	double precision
MPI_LONG_DOUBLE	long double		
MPI_C_COMPLEX **MPI_C_FLOAT_COMPLEX**	float _Complex	**MPI_COMPLEX**	complex
MPI_C_DOUBLE_COMPLEX	double _Complex	**MPI_DOUBLE_COMPLEX**	double complex
MPI_C_LONG_DOUBLE_COMPLEX	long double _Complex		
MPI_C_BOOL	_Bool	**MPI_LOGICAL**	logical
MPI_INT8_T **MPI_INT16_T** **MPI_INT32_T** **MPI_INT64_T**	int8_t int16_t int32_t int64_t		
MPI_UINT8_T **MPI_UINT16_T** **MPI_UINT32_T** **MPI_UINT64_T**	uint8_t uint16_t uint32_t uint64_t		
MPI_BYTE	8 binary digits	**MPI_BYTE**	8 binary digits
MPI_PACKED	data packed or unpacked with MPI_Pack()/ MPI_Unpack	**MPI_PACKED**	data packed or unpacked with MPI_Pack()/ MPI_Unpack

Notes:

- Programmers may also create their own data types (see Derived Data Types).
- MPI_BYTE and MPI_PACKED do not correspond to standard C or Fortran types.
- Types shown in GRAY FONT are recommended if possible.
- Some implementations may include additional elementary data types (MPI_LOGICAL2, MPI_COMPLEX32, etc.). Check the MPI header file.

Destination

An argument to send routines that indicates the process where a message should be delivered. Specified as the rank of the receiving process.

Source

An argument to receive routines that indicates the originating process of the message. Specified as the rank of the sending process. This may be set to the wild card MPI_ANY_SOURCE to receive a message from any task.

Tag

Arbitrary non-negative integer assigned by the programmer to uniquely identify a message. Send and receive operations should match message tags. For a receive operation, the wild card MPI_ANY_TAG can be used to receive any message regardless of its tag. The MPI standard guarantees that integers 0-32767 can be used as tags, but most implementations allow a much larger range than this.

Communicator

Indicates the communication context, or set of processes for which the source or destination fields are valid. Unless the programmer is explicitly creating new communicators, the predefined communicator MPI_COMM_WORLD is usually used.

Status

For a receive operation, indicates the source of the message and the tag of the message. In C, this argument is a pointer to a predefined structure MPI_Status (ex. stat.MPI_SOURCE stat.MPI_TAG). In Fortran, it is an integer array of size MPI_STATUS_SIZE (ex. stat(MPI_SOURCE) stat(MPI_TAG)). Additionally, the actual number of bytes received is obtainable from Status via the MPI_Get_count routine. The constants MPI_STATUS_IGNORE and MPI_STATUSES_IGNORE can be substituted if a message's source, tag or size will be be queried later.

Request

Used by non-blocking send and receive operations. Since non-blocking operations may return before the requested system buffer space is obtained, the system issues a unique "request number". The programmer uses this system assigned "handle" later (in a WAIT type routine) to determine completion of the non-blocking operation. In C, this argument is a pointer to a predefined structure MPI_Request. In Fortran, it is an integer.