NAME

MPIL_Comm_gps − LAM/MPI-specific function to return the GPS information of a given rank from a communicator

SYNOPSIS

```
#include <mpi.h>
int MPIL_Comm_gps(MPI_Comm comm, int rank, int *pnid,
        int *ppid)
```

INPUT PARAMETERS

comm - communicator (handle)
rank - rank of process to query (integer)

OUTPUT PARAMETERS

pnid - LAM node ID
ppid - LAM process ID

NOTES

In the LAM implementation of MPI, each communicator has an integer context ID associated with it for synchronizing on different contexts. This ID is global to all the processes in the communicator's group, and uniquely identifies that communicator for each process. These properties allow the processes to safely exchange messages without interference from operations on other communicators.

The MPI standard does not provide a way to access/view this implementation-dependent synchronization since communicators are opaque objects. Users do not need such access for normal MPI operations. On the other hand, when debugging MPI applications, the opaque nature of communicators hinders the user's efforts. This is especially true on fully observable systems such as LAM, where users can monitor the full state of the processes and message queues, which includes the context ID (see *mpitask* (1) and *mpimsg* (1)).

LAM supplementary functions do not operate with communicators and ranks but with nodes and process identifiers. *MPIL_Comm_gps* accepts the MPI values and returns the LAM values. In the case of an intercommunicator the values returned are those of the process with the given rank in the remote group.

This is a LAM/MPI-specific function and is intended mainly for debugging. If this function is used, it should be used in conjunction with the *LAM_MPI* C preprocessor macro

```
#if LAM_MPI
int nid, pid;
MPIL_Comm_gps(MPI_COMM_WORLD, 0, &nid, &pid);
#endif
```

NOTES FOR FORTRAN

All MPI routines in Fortran (except for *MPI_WTIME* and *MPI_WTICK*) have an additional argument *ierr* at the end of the argument list. *ierr* is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the *call* statement.

All MPI objects (e.g., *MPI_Datatype* , *MPI_Comm*) are of type *INTEGER* in Fortran.

ERRORS

If an error occurs in an MPI function, the current MPI error handler is called to handle it. By default, this error handler aborts the MPI job. The error handler may be changed with *MPI_Errhandler_set* ; the predefined error handler *MPI_ERRORS_RETURN* may be used to cause error values to be returned (in C and Fortran; this error handler is less useful in with the C++ MPI bindings. The predefined error

handler *MPI::ERRORS_THROW_EXCEPTIONS* should be used in C++ if the error value needs to be recovered). Note that MPI does *not* guarantee that an MPI program can continue past an error.

All MPI routines (except *MPI_Wtime* and *MPI_Wtick*) return an error value; C routines as the value of the function and Fortran routines in the last argument. The C++ bindings for MPI do not return error values; instead, error values are communicated by throwing exceptions of type *MPI::Exception* (but not by default). Exceptions are only thrown if the error value is not *MPI::SUCCESS* .

Note that if the *MPI::ERRORS_RETURN* handler is set in C++, while MPI functions will return upon an error, there will be no way to recover what the actual error value was.

MPI_SUCCESS
- No error; MPI routine completed successfully.

MPI_ERR_COMM
- Invalid communicator. A common error is to use a null communicator in a call (not even allowed in *MPI_Comm_rank*).

MPI_ERR_RANK
- Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (*MPI_Recv* , *MPI_Irecv* , *MPI_Sendrecv* , etc.) may also be *MPI_ANY_SOURCE* .

MPI_ERR_ARG
- Invalid argument. Some argument is invalid and is not identified by a specific error class. This is typically a NULL pointer or other such error.

SEE ALSO

MPIL_Comm_id, MPIL_Type_id

LOCATION

mpil_id.c