MPI Version 1.1 Validation Suite 1.0
User's Guide

Document #665509

September 27, 1996

Document Revision 1.5

Contents

1.0 Preface

The Message Passing Interface Standard (MPI version 1.1) is a document published by the Message Passing Interface Forum, and describes the work supported in part by ARPA and the National Science Foundation under Grant ASC-9310330, the NSF Science & Technology Center Cooperative Agreement No. CCR-8809615, and by the Commission of the European Community through Esprit project P6643. The document contains 231 pages of specifications that define 128 distinct functions and dozens of constants. The specifications include both a C and Fortran binding to these functions and constants.

With any specification this large, it is inevitable that there will be discrepancies and disagreements between different implementations. It is thus the primary goal of the U. S. government to produce or have produced a portable MPI validation suite (both in C and Fortran) that verifies the compliance of an MPI implementation to the MPI Standard. Intel Corporation entered into a Cooperative Research and Development Agreement with Rome Labs to pursue this goal. The goal of this suite is to provide a set of systematic tests that encourage higher quality MPI implementations, better portability, and hence a wider acceptance of MPI. This validation suite is the result produced by this project and is available through public domain, subject to a royalty free copyright license from Intel and Intel's disclaimer of all warranty and liability. As described on the cover of this guide, Intel does not offer maintenance, technical support, updates, enhancements, or modifications for the Validation Suite.

2.0 Intended Audience

This document is targeted at MPI Validation Suite 1.0 users.

3.0 Related Documents / References

In addition to this guide, user may find the following documents or references to be useful:

1. *MPI: A Message-Passing Interface Standard version 1.1*; MPI Forum; June 12, 1995.
2. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*; Gropp, Lusk, and Skjellum; MIT Press; 1994
3. *MPI The Complete Reference*; Snir, Otto, Huss-Lederman, Walker, and Dongarra; MIT Press; 1996

4.0 Definition of Terms

blocking operation - an operation which when completed will allow the user to re-use resources specified in the call.

collective operation - an operation which involves all members of a given communicator.

communicator - the communication domain for a communication operation. It encapsulates process rank(s), group, and sometimes the virtual topology of communication processes.

CPP - C Preprocessor.

equivalence class - test cases belong to the same equivalence class if they test the same thing. If one case catches a bug, other cases of the same equivalence class would catch the same bug, and if one case doesn't catch a bug, other cases of the equivalence class would not catch the same bug either.

error test - tests focused on invalid parameter value(s) or typographical error(s) on a call or command. Is the correct error generated for the case? Does the system gracefully handle the unexpected input? Does the application return a reasonable error?

extent - extent of a datatype is defined to be the span from the first byte to the last byte occupied by entries in this datatype, rounded up to satisfy alignment requirements.

functional test - tests designed to verify that the given command, library call or application behaves and operates as specified in its own specification. The test would include exercising all switches or running various equivalence class cases through the parameters. This is the largest portion of the test cases to be designed and executed.

group - an ordered collection of processes participating in a communication, each with a rank ranging from 0 to 1 less than the size of the group.

intercommunicator - a communication structure allowing communication between two different intracommunicators. A message passed within an intercommunicator must not be received within the intracommunicator from which it was sent.

intracommunicator - a particular example of a context. This is a (non-empty) set of all processes in the application. Messages passed in this communication domain must stay within this communicator.

local - the completion of the operation depends only on the local executing process.

message - a piece of information to be sent or received.

message envelope - information (source, destination, tag and communicator) that can be used to distinguish messages and selectively receive them.

message tag - an integer value used by a program to distinguish different types of messages.

MPI - (Message Passing Interface) is a standard specification for message-passing libraries.

MPICH - a freely available implementation of the MPI standard that runs on a wide variety of systems jointly developed by Argonne National Lab and Mississippi State University.

non-blocking operation - operation may return before it is completed and before the user is allowed to re-use resources specified in the call.

non-local - completion of operation may require execution of some MPI procedure on another process.

persistent request - pre-allocated, pre-defined request that may be reused until it is explicitly freed.

platform - for the purpose of this document, platform refers to the combination of the system architecture and the MPI implementation pair.

process topology - an optional attribute added to an intracommunicator. It provides a convenient naming mechanism for the processes of a group within a communicator.

rank - an integer to identify a process ranging from 0 to one less than the size of the group the process is in.

type map - a sequence of pairs of datatype and displacement.

type signature - a sequence of datatypes.

5.0 Introduction

The MPI Validation Suite version 1.0 contains a set of portable test programs written in ANSI C and Fortran 77[1], a set of common library routines and portable test driver environment. Test driver environment implementation for the Intel Paragon™ Supercomputer system / Paragon OS R1.4 and Sun 4 workstations running SunOS 4.1.4 and MPICH 1.0.12 are included in the package. Please refer to section 17.0 for details in porting the driver environment to other platforms.

This document describes the details of the MPI Validation Suite and test driver environment. Information about how to use or set up the environment to build, execute and / or clean test(s) in C and Fortran, interpret and analyze test results, how to add new tests, or port the driver environment to other platforms are also include. For a list of the most frequently asked questions, please refer to Appendix A.

6.0 System Requirement(s)

6.1 Hardware

Since this is intended to be a portable test implementation, a list of specific supported platforms is not practical. In general, the hardware requirement for the MPI implementation used will have to be met. The test cases have been designed to scale up to 64 nodes (ranks). While they may scale further, some of the test algorithms may become cumbersome.

This suite was designed to operate on a homogeneous platform. Automatic test(s) compilation and execution in a heterogenous system is not supported. The -build option of the mpitest driver may partially solve the problem of helping users use tests in a heterogenous system. However, further changes may be required in the validation suite in order to run tests in heterogenous system, especially if data representation on the systems does not allow for exact conversion. Please refer to section 11.3 for details in the -build option of the mpitest driver and section 17.0 for details on porting the validation suite to a other platforms.

1. Fortran 77 with the extensions specified in section 6.2.3

6.2 Software

6.2.1 UNIX or POSIX compatible Operating Environment

The test driver consists of make files and shell scripts. Therefore, the underlying operating system on the platform being used to run the test driver must be a UNIX or POSIX compatible shell providing standard UNIX / POSIX versions of command utilities (sh, make, grep, awk, cp, mv, ls, *etc*).

6.2.2 Process I/O

Each MPI process must be able to write standard output or error strings to a file which is accessible to the test driver. This is required for two reasons. MPI Standard v1.1 does not specify an I/O interface. More importantly, because the state of MPI is undefined after an error, and most of the I/O is generated by test cases in response to an unexpected error, an independent means of reporting results is required.

By default, the tests will write to standard output (stdout) though -stdoutfile of the mpitest driver allows I/O of each rank to be written to its own file. Please refer to section 11.4 for more details. All I/O routines are handled by the test libraries (in MPITEST_message()), so this can be easily modified by the user.

6.2.3 Fortran and C compilers

Fortran and C compilers which support run time options of the GNU compilers semantically and syntactically equivalently are required. The following list some of the options used:

<u>C & Fortran compiler options:</u>

-c - Compile the source but do not link.

-o - Place output in specified file.

-I - Specify all search directories for search include files.
This must work both for #include and Fortran INCLUDE.

-L - Specify directories to be search for -l.

-l - Link with library.

The C compiler must be capable of compiling ANSI C source and the Fortran compiler must be capable of compiling Fortran 77 source.

In addition, Fortran has several requirements outside standard Fortran 77:

- Like MPI, routine names must be able to posses up to 32 characters.
- Like MPI, you must be able to call subroutines with varying data types (For example, an integer array or complex array).
- It must support INLCUDE files.

The Fortran compiler must also supply C-preprocessor functionality, or you must use another preprocessor to convert file.F files to file.f (such as the GNU pre-processor).

6.2.4 File system

The file system where the validation suite resides must be capable of handling hierachical file structures.

6.2.5 Other

In addition, there might be more hardware / software requirement(s) depending on which MPI implementation / system is used. Please refer to the documentation of your MPI library and system for more details.

7.0 Package Contents

7.1 User's Guide

A guide detailing how to use the validation suite (this document)

7.2 MPI Test driver and environment

MPI test driver and environments for the Intel Paragon™ supercomputer running Paragon OS Release R1.4 and cluster of Sun 4 workstations running SunOS 4.1.4 using MPICH 1.0.12 are provided. A test result gathering utility is also included in the package

7.3 C and Fortran test suites

Functional and error test programs written in ANSI C and Fortran 77[1] for the various areas of the MPI implementation to be verified.

7.4 Common library routines in C and Fortran

Common library routines written in ANSI C and Fortran 77[1] used by for the test programs.

7.5 Run script and Makefile template

Run script and makefile template for adding new tests.

8.0 Installation

8.1 Space requirement

The package contains 2 files: the Validation Suite User's Guide and the compressed tar file for the Validation Suite. The Validation suite User's Guide is in postscript format (mpitest_guide.ps) and requires about 607 kilobytes of file space. and the uncompressed tar file (mpitest.tar) for the validation suite requires 21 megabytes of file space. Additional space will be required for compiling and running the validation suites.

1. Fortran 77 with the extensions specified in section 6.2.3

8.2 Unpacking the compressed postscript and tar file

The file mpitest.tar.Z is a compressed tar file. It needs to be uncompressed and untared.

To uncompress:

```
uncompress mpitest_guide.ps.Z
uncompress mpitest.tar.Z
```

To untar:

```
tar xvf mpitest.tar
```

8.3 Setting environment variables and search path

After the compressed tar file has been uncompressed and untarred, there are a number of environment variables that need to be set before any test can be executed using the driver. It is recommended that these be set up in an environment file or in a user's login set up file (e.g. .cshrc) MPITEST_HOME tells the driver where the validation suite resides while MPITEST_ARCH tells the test driver what architecture / platform the tests will be run on. After these environment variables have been set up properly, $MPITEST_HOME/bin needs to be added to user's search path in order for the system to successfully find the utility executables (e.g. mpitest). For convenience, this set up should be done automatically in a standard setup file (e.g. .cshrc).

Assuming the Validation suite has been installed in /home/foobar/MPITEST, setting up MPITEST_HOME in csh would be:

```
setenv MPITEST_HOME /home/foobar/MPITEST
```

The driver environment for the Intel Paragon™ Supercomputer running Paragon OS R1.4 and the Sun 4 workstation running SunOS 4.1.4 using MPICH comes with the package.

Setting MPITEST_ARCH is optional. You can either set the environment variable or you can use the command line options of the mpitest driver. Please refer to section 18.0 for details on command line options of the mpitest driver.

To set up the installed validation suite to be run on Intel Paragon™ Supercomputer using Paragon native MPI library:

set the environment variable MPITEST_ARCH to be INTEL_paragon_nx

(in csh, `setenv MPITEST_ARCH INTEL_paragon_nx`)

To set up the installed validation suite to be run on Sun 4 clustered network workstations using MPICH MPI implementation:

set the environment variable MPITEST_ARCH to be MPICH_sun4_ch_p4

(in csh, `setenv MPITEST_ARCH MPICH_sun4_ch_p4`)

For details on porting the driver environment to other platforms, please refer to section 17.0.

Next, you need to add $MPITEST_HOME/bin to the user's search path.

(in csh, `set path = ($MPITEST_HOME/bin $path )`)

If you are installing the driver environment for the MPICH for the Sun4 implementation, using the provided driver environment for the platform, MPITEST_MACHINE_FILE needs to be set up pointing to a file where all the Sun4 workstations to be used to simulate a multi-computer are defined. Please refer to the MPICH 1.0.12 User's Guide for setting up a machine file.

If the MPI header files (mpi.h, mpif.h, *etc*) for the MPI implementation to be used are not in a standard place where the host language compiler could automatically find it, an additional environment variable needs to be set up. Currently, the mpitest driver expects to find both the C and Fortran header files at the same place as defined by this environment variable.

Assuming the include files are in $MPITEST_HOME/include, in csh,

```
setenv MPITEST_MPI_INCLUDE_DIR $MPITEST_HOME/include
```

The MPI Validation suite is now successfully installed.

9.0 Quick Start

To quickly start a single test:

Step 1: Go to a test

```
cd $MPITEST_HOME/Test/c/collective/functional/MPI_Bcast
```

Step 2: Run the test with 4 ranks

mpitest -np 4

The following output from the driver environment should be seen in the test run:

```
09/11/96 12:05:04 hostname Making mpitest C library
09/11/96 12:05:06 hostname Complete making mpitest C library
09/11/96 12:05:07 hostname Making mpitest Fortran library
09/11/96 12:05:09 hostname Complete making mpitest Fortran library
09/11/96 12:05:12 hostname Test: MPI_Bcast
09/11/96 12:05:13 hostname Using default configuration
09/11/96 12:05:14 hostname Running MPI_Bcast
09/11/96 12:05:48 hostname Completed running 1 test(s)
```

The test run is now completed. The test output can be seen in the test log file MPI_Bcast.${MPITEST_ARCH}_${MPITEST_NPROCS}_L and the result (Pass / Fail) can be seen in the test result file MPI_Bcast.${MPITEST_ARCH}_${MPITEST_NPROCS}_R, where ${MPITEST_NPROCS} was defined by the -np parameter.

Please refer to section 11.0 for more details on compiling and running a list of tests and section 12.0 for further details on interpreting and analyzing test results.

10.0 Validation Suite Anatomy

This section describes the MPITEST directory hierachy. It briefly describes where files are located:

Under $MPITEST_HOME, the following subdirectories should be found:

Test: Where all the test programs reside.

Test/c/blocking - All C Blocking operations tests.

Test/c/collective - All C Collective operations tests.

Test/c/datatype - All C Derived datatype operations tests.

Test/c/env_manager - All C Environment manager tests.

Test/c/grp_ctxt_comm - All C Group, Context, and Comm operations tests.

Test/c/misc - All other C tests.

Test/c/nonblocking - All C Nonblocking operations tests.

Test/c/persist_request - All C Persistent request operations tests.

Test/c/probe_cancel - All C Probe and Cancel operations tests.

Test/c/sendrecv - All C sendrecv operations tests.

Test/c/topo - All C topology operations tests.

Test/fortran/blocking - All Fortran Blocking operations tests.

Test/fortran/collective - All Fortran Collective operations tests.

Test/fortran/datatype - All Fortran Derived datatype tests.

Test/fortran/env_manager - All Fortran Environment manager tests.

Test/fortran/grp_ctxt_comm - All Fortran Group, Context, and Comm tests.

Test/fortran/misc - All other Fortran tests.

Test/fortran/nonblocking - All Fortran Nonblocking operations tests.

Test/fortran/persist_request - All Fortran Persistent request operations tests.

Test/fortran/probe_cancel - All Fortran Probe and Cancel operations tests.

Test/fortran/sendrecv - All Fortran sendrecv operations tests.

Test/fortran/topo - All Fortran topology operations tests.

Each directory is further broken down by:

<u>functional</u>

These tests verify the proper operation of MPI calls when used according to the specification. Any unexpected errors are logged, otherwise the tests will simply report a Start and Passed message.

<u>error</u>

Error tests are designed to test one invalid parameter at a time, with the error returned to caller and verified by the test suite. The ability to run these tests may be limited by the quality of the MPI implementation, since the MPI specification does not specify the state of MPI after an error. All tests call the library routine MPITEST_report(), which as delivered assume that MPI_Allreduce can be successfully used. If this is not the case on the implementation you are testing, the library can be easily modified to use a different algorithm (such as printing status then calling MPI_Abort()).

bin: Where the utility executables, environment files, job submission scripts and related files are located. This is where files that are specific to the platform you are using resides that may need to be modified.

bin/mpitest - MPI test driver.

bin/mpitest.INTEL_paragon_nx - Driver environment for Paragon supercomputer.

bin/mpitest.MPICH_sun4_ch_p4 - Driver environment for Sun4/MPICH.

bin/Makefile.master - Master makefile for the MPI test driver environment.

bin/startjob.INTEL_paragon_nx - Job submission script for the Paragon supercomputer.

bin/startjob.MPICH_sun4_ch_p4 - Job submission script for the Sun4/MPICH.

bin/mpiresult - Gathers results of all tests.

include: Where all the include files for the validation suite are located. These may be modified easily by the user.

mpitest_cfg.h - Default C configuration file.

mpitest.h - Global C constants and variables for the test suite

mpitest_cfgf.h - Default Fortran configuration file.

mpitest_def.h - Miscllany C constant file.

mpitestf.h - Global Fortran constants and variables for test suite.

externalf.h - Fortran library external declaration.

foptions.h - Fortran specific options configuration file.

lib: Where the validation suite support libraries reside.

libmpitest.c - C library routines.

libmpitestf.F - Fortran library routines.

template: Where all of the Makefile and run script templates are located for adding new test(s).

Makefile.c - Makefile template for C tests.

Makefile.f - Makefile template for Fortran tests.

run_script.ksh - Korn shell run script template.

run_script.sh - Born shell run script template.

node.c - C test program template.

node.F - Fortran test program template.

11.0 Compiling and Running Tests

Compilation and execution of test is automatic when using the mpitest environment. The rules for compiling and test execution are governed by the rules set in the master Makefile and the local test Makefile. There are still a number of options a user can set to configure test compilation and execution

11.1 -testlist <file>

If not specified, the mpitest driver creates a list of tests to be compiled and / or run in the current directory where mpitest is invoked. It does so by looking through the current directory and its subdirectories. For any directory with a Makefile, the mpitest driver will consider it a test and will compile and execute the test. User may also specify a list of tests to be run by using the -testlist option. The -testlist option takes an argument of an ASCII file which contains test name(s) (relative or absolute pathnames) separated by newlines. This option can also be used with the -build, or -cleanall option. A testlist of all tests is supplied in $MPITEST_HOME/Test/testlist.

Please refer to section 11.3 for more details on -build and section 15.0 for more details on -cleanall.

11.2 -clibonly, -flibonly

Both the C and Fortran common library routines used by the test suite are automatically compiled and updated, if required, every time the mpitest driver is invoked to compile and / or run tests. Users may choose to compile just the C library (-clibonly) or just the Fortran library (-flibonly) in the case when only C or Fortran tests are being run.

11.3 -build

With the -build option of the mpitest driver, user may elect to just compile a list of tests without submitting them to be executed. This may be useful for environments where cross compiling is available.

11.4 -stdoutfile

In a parallel environment, output from each rank can be mixed up with output from other ranks. The -stdoutfile option causes each rank to put their output in a file. All the files created by each rank will be concatenated at the end of the test run ordered by rank number.

This way, the output from each rank is grouped together which may help debugging test failures in some cases. The only requirement of the test river is that output be line buffered; i.e. a new-line terminated string will be printed intact but rank ordering is not required.

12.0 Interpreting or Analyzing Test Result

For each test, the test output is stored in the test log file (<testname>.<platform name>_<number of ranks>_L) residing in each test directory. The test result (Pass / Fail) is in the test result file (<testname>.<platform name>_<number of ranks>_R) residing in each test directory. Compilation problem(s) would be recorded in the make log file (makelog.<platform>_<number of ranks>). The following shows an example of a result file for a test pass and a test failure:

<u>Test pass:</u>

```
TEST_RESULT: P hostname 31 #4 MPI_Bcast
```

<u>Test failure:</u>

```
TEST_RESULT: F (wrong PASSED count, Expected: 1, Actual: 0) hostname 6 #2
MPI_Cancel_err1
```

<u>Test to be manually verified:</u>

```
TEST_RESULT: V mendel 339 #5 MPI_Bsend_perf
```

The test result for each test is described by either P, F, or V in its test result file. A P stands for a test pass, an F stands for a test failure and a V stands for a test with result not capable of being detected automatically and thus needs to be verified manually.

For each test failure, there is always a brief reason for the failure in the result file. It could be “wrong PASSED count” which says it is expecting to find x number of the keyword “PASSED” in the test log file and did not find them, or “found FAILED” which says it found the keyword “FAILED” in the test log file. In either case, further investigation of the test log file and local make log file would need to be done in order to narrow down the cause of the test failure. A test failure may be attributed to bug(s) in the underlying hardware(s) or software(s) or could simply be mistake(s) in configuration. In any case, the test output log should be used to narrow down any run time problem(s) and the makelog file should be used to find any compilation problem(s).

There are 3 additional fields that may be useful to users in each result file - the name of the machine (where the driver ran), the total execution time (in seconds), and the number of ranks used (prefixed by # in the result file). This information follows the letter P for test passes or the failure result (in bracket) for test failures.

12.1 Analyzing test failures

12.1.1 -verbose option

The -verbose option may be helpful for debugging problem(s) using a test. It takes one integer argument of verbosity level (1 or 2). This option causes additional output to be generated in the test log file and make log file at run time and hence may give more details on where the problem may lie. More output will be seen at verbosity level 2 than that of verbosity level 1. Extra care must be taken when using -verbose, especially when using a large number of ranks as the amount of output may be too large to comprehend and may also cause I/O issues (output file too large, system may thrash, *etc*) on some systems. The following is an example of test run with and without the -verbose option:

Without:

Test log file:

MPITEST info (0): Starting MPI_Cancel_err1: Inactive request test

Killed (core dumped)

Make log file:

```
node.c ===> node.INTEL_paragon_nx_bx

MPI_Cancel_err1.sx ===> MPI_Cancel_err1.INTEL_paragon_nx_5_L

MPI_Cancel_err1.INTEL_paragon_nx_5_L ===>
MPI_Cancel_err1.INTEL_paragon_nx_5_R

TEST_RESULT: F (found FAILED) hostname 6 #5 MPI_Cancel_err1

MPI_Cancel_err1.INTEL_paragon_nx_5_R test failed.
```

With -verbose 2:

Test log file:

```
MPITEST info1 (1): Receiving message from source: 0 to 1, tag: 1

MPITEST info  (0): Starting MPI_Cancel_err1: Inactive request test

MPITEST info1 (0): Sending message from source 0 to 1, tag: 1

MPITEST info1 (0): Waiting for send request to be completed

MPITEST info1 (1): Verifying data received

MPITEST info1 (0): Cancelling an inactive request ...

MPITEST info1 (1): Verifying output status object

MPITEST info1 (1): Node results: pass=0, fail=0, verify=0

Killed (core dumped)
```

Make log file:

```
MPI_Cancel_err1.sh ===> MPI_Cancel_err1.sx

node.c ===> node.INTEL_paragon_nx_bx

icc -I. -I/usr/local/MPITEST/include -g  -o node.INTEL_paragon_nx_bx node
.c  -L/usr/local/MPITEST/lib -lmpitest_INTEL_paragon_nx -lmpi   -nx ;  rm
```

```
 -f MPI_Cancel_err1.INTEL_paragon_nx_4_L
MPI_Cancel_err1.sx ===> MPI_Cancel_err1.INTEL_paragon_nx_4_L
MPI_Cancel_err1.INTEL_paragon_nx_4_L ===>
MPI_Cancel_err1.INTEL_paragon_nx_4_R
TEST_RESULT: F (found FAILED) hostname 5 #4 MPI_Cancel_err1
MPI_Cancel_err1.INTEL_paragon_nx_4_R test failed.
```

From the output with -verbose 2, one could find out the core file was generated when the test program (rank 0) is cancelling an inactive request. -verbose may not be supported for Fortran tests since there is no standard way to handle command line arguments in that case. You must modify MPITEST_VERBOSE manually and recompile the test library and the test to use it.

12.1.2 What to look for in test failures

Each test detects incorrect result (such as corrupted or incorrect data or an unexpected error), it will generate a message to the log file describing the problem as best as it can. For most tests, look for the keyword "FAILED", "MPITEST error", or "MPITEST_ERROR". Where these are found should be fairly close to where the failure occurs. It is recommended that the user look at the test log file around these areas and not the area just below or above since in a parallel environment, each unit output from each ranks could be mixed up on some systems.

In a case of a test hang or system crash, the test log file, test result file, and the make log file may not be completely generated.

12.2 Test results gathering utility

In the bin directory of the package, there is a test results gathering utility: mpiresults. This utility gathers information from all the results files from the tests specified, and generates a simple report (results.<platform name>_<number of ranks>) in the current directory where mpiresults is invoked.

Similar to the mpitest driver, it supports -arch, -envfile, -np, -testlist, and a number of other options. Type mpiresults -help for more details.

13.0 Environment and Advanced Configuration

13.1 Changing the environment

13.1.1 Environment file

The platform specific environment resides in $MPITEST_HOME/bin/mpitest.<platform name>.env by default. Users may provide their own environment using the -env option of the mpitest driver. The -env option takes an argument of filename (relative or absolute path name). Please refer to section 17 for details on the environment file.

13.1.2 Manually overriding environment

Users may also elect to override a subset of environment variables by a manual set without changing the environment or providing a separate version of the environment file. For example, if you wish to use a different C compiler for a particular run, you may do the following in csh before invoking the mpitest driver:

setenv MPITEST_CC /usr/local/bin/gcc

This newly set value will apply during the life of the shell where the environment is explicitly overridden so long as the mpitest test driver is invoked there as well. For a more persistent change, users may wish to provide a full environment in a separate file. For more details on the driver environment, please refer to section 17.0.

13.2 Changing test parameters

13.2.1 Configuration file

Some functional test parameters are not in the test but in a configuration file (mpitest_cfg.h for C and mpitest_cfgf.h for Fortran). A configuration file contains information such as lengths of messages to be sent, communicators, datatypes, compiler directives, macros, and some globally defined constants. This is so a user may change a parameter in one place without having to change all the tests if a different set of testing parameters is desired. Users may also wish to create a local configuration file (residing in the test directory) for a particular test. When compiling tests, the make rules of the mpitest utility will first search for a local configuration file. If that does not succeed, it will use the global default configuration file (in $MPITEST_HOME/include).

Users may also find the first time they run the validation suite on a new implementation that there are just too many failures to analyze at once and desire a more systematic approach, or the tests are too big and take too long to run. By modifying the configuration files, many of the functional tests may be made as simple or stressful as desired. In addition, items such as message lengths may be increased or decreased which would change the amount of memory required to run a test. You may also decide to test one datatype at a time, or just MPI_COMM_WORLD, or any other combination without having to modify tests.

IMPORTANT NOTE:

> *Extra care must be taken to provide valid input in the configuration files, as they will be considered correct input to MPI calls.*

There are 2 configuration files. One for the C version ($MPITEST_HOME/include/

mpitest_cfg.h) and one for the Fortran version ($MPITEST_HOME/include/mpitest_cfgf.h). The following describes the configuration arrays in each configuration file.

13.2.1.1 Message lengths

```
int MPITEST_message_lengths[MPITEST_CONFIG_ARRAY_SIZE] =
{
  0,
  MPITEST_MULT_INC, 8, 8000, 10,
  MPITEST_REPEAT, 320, 8,
  48,
  MPITEST_ADD_INC, 1016, 1032, 8,
  65536,
  MPITEST_END_TOKEN
  };
```

The above shows the message length array in a default C configuration file. The array contains values to be used as message lengths (in bytes) for functional tests. The array should always contains at least one message length and be terminated by MPITEST_END_TOKEN or the tests will not be configured correctly. The following shows the format of the array (+ means one of more):

```
int MPITEST_message_lengths[MPITEST_CONFIG_ARRAY_SIZE] =
{
[<msg_len_token>, ]+ MPITEST_END_TOKEN
};
```

where

<msg_len_token> :: <integer> | <msg_len_specifier>, <integer expr> [, <integer expr>]*

That is, the message lengths array contains a non-empty set of message length tokens. Each token can be an integer or it can be a message length specifier followed by its argument(s)

Table 1: Message length specifier

Message length specifier	Argument	Description
MPITEST_ADD_INC	<start value>, <end value>, <increment>	message lengths specified are: <start value>, <start value> + <increment>, <start_value> + 2 * <increment>, ..., <end_value>.

Table 1: Message length specifier

Message length specifier	Argument	Description
MPITEST_MULT_INC	<start value> <end value>, <increment>	message length specified are: <start value>, <start value> * <increment>, <start value> * <increment> * <increment>, ..., <end_value>.
MPITEST_REPEAT	<value>, <repeat count>	message length specified are: <value>, <value>, ..., <value> (<repeat count> <value>s).

Therefore, the above example message lengths array is equivalent to the following:

```
int MPITEST_message_lengths[MPITEST_CONFIG_ARRAY_SIZE] =
{
  0,
  8, 80, 800, 8000
  320, 320, 320, 320, 320, 320, 320, 320,
  48,
  1016, 1024, 1032,
  65536,
  MPITEST_END_TOKEN
  };
```

The message length specifiers are the same for Fortran. The Fortran message lengths array is in the same format except in the Fortran syntax.

13.2.1.2 Datatypes

```
int MPITEST_types[] =
{
MPITEST_int, MPITEST_short_int, MPITEST_long, MPITEST_unsigned_short,
MPITEST_unsigned, MPITEST_unsigned_long, MPITEST_float, MPITEST_double,
MPITEST_char, MPITEST_unsigned_char, MPITEST_long_double, MPITEST_byte,
MPITEST_derived1,MPITEST_derived2,MPITEST_END_TOKEN
};
```

The above shows a datatypes array in the default configuration file. Instead of using the MPI type names directly from the *MPI Standard*, we chose to us a different set so that an MPITEST_END_TOKEN can be added to the list portably. The datatypes array contains datatypes to be used in most functional tests. It needs to be configured so that there is at least one datatype token in the array and be terminated by MPITEST_END_TOKEN or some of the tests will not be configured correctly. The following table shows all the MPI types and their corresponding MPITEST datatype tokens.

Table 2: MPITEST C datatype specifier token

MPI C data type	MPITEST C datatype token
MPI_CHAR	MPITEST_char
MPI_SHORT	MPITEST_short_int
MPI_INT	MPITEST_int
MPI_LONG	MPITEST_long
MPI_UNSIGNED_CHAR	MPITEST_unsigned_char
MPI_UNSIGNED_SHORT	MPITEST_unsigned_short
MPI_UNSIGNED	MPITEST_unsigned
MPI_UNSIGNED_LONG	MPITEST_long
MPI_FLOAT	MPITEST_float
MPI_DOUBLE	MPITEST_double
MPI_LONG_DOUBLE	MPITEST_long_double
MI_LONG_LONG_INT	MPITEST_longlong
Derived datatype 1	MPITEST_derived1
Derived datatype 2	MPITEST_derived2

Table 3: MPITEST Fortran datatype specifier token

MPI Fortran data type	MPITEST Fortran datatype token
MPI_INTEGER	MPITEST_INTEGER
MPI_REAL	MPITEST_REAL
MPI_DOUBLE_PRECISION	MPITEST_DOUBLE_PRECISION
MPI_COMPLEX	MPITEST_COMPLEX
MPI_LOGICAL	MPITEST_LOGICAL
MPI_CHARACTER	MPITEST_CHARACTER
MPI_INTEGER1	MPITEST_INTEGER1
MPI_INTEGER2	MPITEST_INTEGER2

Table 3: MPITEST Fortran datatype specifier token

MPI Fortran data type	MPITEST Fortran datatype token
MPI_INTEGER4	MPITEST_INTEGER4
MPI_REAL2	MPITEST_REAL2
MPI_REAL4	MPITEST_REAL4
MPI_REAL8	MPITEST_REAL8
MPI_DOUBLE_COMPLEX	MPITEST_DOUBLE_COMPLEX

13.2.1.3 Communicators

```
int MPITEST_comm_sizes[MPITEST_CONFIG_ARRAY_SIZE] =
{
/*0*/  MPITEST_comm_type_world,

/*1*/  MPITEST_comm_type_self,

/*2*/  MPITEST_comm_type_merge,
/*3*/   MPITEST_comm_type_create,
/*4*/    MPITEST_comm_inc, 1, 4, 2,
/*8*/   MPITEST_comm_type_create,
/*9*/    MPITEST_comm_inc, 2, MPITEST_comm_last_rank, 2,

/*13*/  MPITEST_comm_type_split,
/*14*/   MPITEST_comm_inc, 1, MPITEST_comm_last_rank, 2,

/*18*/   MPITEST_comm_type_split,
/*19*/    MPITEST_node_list, 2, 0, 2,

/*23*/  MPITEST_comm_type_dup,
/*24*/   MPITEST_comm_inc, 0, MPITEST_comm_last_rank, 3,

/*28*/  MPITEST_comm_type_inter,
/*29*/   MPITEST_comm_type_create,
/*30*/    MPITEST_comm_inc, 1, 4, 2,
/*34*/   MPITEST_comm_type_create,
/*35*/    MPITEST_comm_inc, 2, MPITEST_comm_last_rank, 2,
```

```
/*39*/  MPITEST_END_TOKEN
}
```

The above shows a communicator sizes array in the default configuration file. The communicators array contains token(s) to specify the size of communicator(s) to be generated in most tests. It needs to be configured so that there is at least one communicator size token in the array and is terminated by MPITEST_END_TOKEN or some of the tests will not be configured correctly. The following shows the syntax of the communicator sizes array:

```
int MPITEST_comm_sizes[] = {
[<communicator size token>, ]+ MPITEST_END_TOKEN
}
```

where

<communicator size token> ::	MPITEST_comm_type_world |	
	MPITEST_comm_type_self |	
	MPITEST_comm_type_merge	<comm rank specifier>
		<comm rank specifier> |
	MPITEST_comm_type_create	<comm rank specifier> |
	MPITEST_comm_type_split	<comm rank specifier>
		<comm rank specifier> |
	MPITEST_comm_type_dup	<comm rank specifier> |
	MPITEST_comm_type_inter	<comm rank specifier>
		<comm rank specifier>

Table 4: Communicator size specifiers

Communicator specifiers	Argument	Description
MPITEST_comm_type_world	N/A	MPI_COMM_WORLD
MPITEST_comm_type_self	N/A	MPI_COMM_SELF.
MPITEST_comm_type_merge	2 <comm rank specifier>s	Merged communicator (MPI_Comm_merge()).
MPITEST_comm_type_create	<comm rank specifier>	Creating a new communicator using the rank(s) (MPI_Comm_create())
MPITEST_comm_type_split	2 <comm rank specifier>s	Split communicator (MPI_Comm_split()).

Table 4: Communicator size specifiers

Communicator specifiers	Argument	Description
MPITEST_comm_type_dup	<comm rank specifier>	Duped communicator (MPI_Comm_dup()).
MPITEST_comm_type_inter	2 <comm rank specifier>s	Inter-communicator.

Table 5: Communicator rank specifiers

Communicator size specifiers	Argument	Description
MPITEST_comm_last_rank	N/A	Specifies the last rank in the current parent communicator.
MPITEST_comm_inc	<start value>, <end_value>, <increment>	Specifies list of ranks: <start value>, <start value> + <increment>, <start value> + 2 * <increment>, ..., largest value in the sequence <= <end value>
MPITEST_node_list	<n>, <rank1>, <rank2>, ... <rankn>	Specifies list of ranks <rank1>, <rank2>, ..., <rankn>

The communicator size and rank specifiers are the same for Fortran. The format for the communicator sizes array in Fortran is the same except in Fortran syntax.

13.2.2 Compiler directives and MACROs

In the configuration file ($MPITEST_HOME/include/mpitest_cfg.h for C and $MPITEST_HOME/include/foptions.h for Fortran), there are some #define compiler directives and MACRO definitions which are used in some tests. The compiler directives and the MACROs are there to change some of the test behaviors if desired. For example, in the MPI_Type_contiguous() function test, each time after a new type is created, the test will send a message using the newly created type. Some users may wish to turn off the verification code for the output status object from MPI_Recv() since any potential failure may not be directly related to MPI_Type_contiguous() and thus will not affect the total number of failures (Since in this case, if the output status object contains invalid information, it may be related to MPI_Recv() instead of MPI_Type_contiguous() directly and the same failure might have been caught in other MPI_Recv() functional tests). Not all compiler directives and MACROs apply to all tests, however. Please refer to each individual test and the configuration file for more details.

13.2.3 Command line options

Users may also override some of the test parameters using the binary executable specific options.

Please refer to section 18.0 for more details on different options group for the mpitest driver. The following is for C tests only:

```
Usage:  testname [-help] [-verbose n] [-type type_string]
          [-context context_string] [-length length_string]
          [-stdoutdir dirname]

Options (order is not important) :
    -help               generate this message

    -verbose n          Set verbosity flag to n, 0 <= n <= 2

    -length length_string   use message length(s) specified by length_string
                    length_string = length_token[,length_token,...]
                    length_token = length (0<=length) or
                      "mult,start,end,inc" (0<start<=end,1<inc) or
                      "add,start,end,inc" (0<=start<=end,1<=inc) or
                      "repeat,val,repitions" (0<=val, 0<repititions)

    -context context_string use context(s) specified by context_string
                    context_string=context_token[,context_token,...]
                    context_token = context_type,size_token
                    context_type = "world" (no size token to follow)
                      or "self" (no size token to follow)
                      or "split" or "duplicate"
                    size_token = size (1 <= size <= number of proc.)
                      or "inc,start_node,end_node,inc_amount"
                      (1<=start_node<=end_node<=nump,1<=inc_amount)
                      or "list,num_nodes,node1,node2,...,nodelast"
                      (0<=node1,node2,...,nodelast<=nump,1<=numnodes
    -type type_string       use type(s) specified by type string
                    type_string = buffer_type[,buffer_type,...]
                    buffer_type =
                    "int", "short_int", "long", "unsigned_short",
                    "unsigned", "unsigned_long", "float", "double",
                    "char", "unsigned_char", "longlong",
                    "long_double", "byte", "derived[12]"
```

Some examples of using the binary executable specific options:

To run test(s) with only a message of length 1024 bytes:

```
mpitest -np 2 -length 1024
```

To run test(s) with message type integer:

```
mpitest -np 2 -type int
```

To run test(s) with message type integer and float:

```
mpitest -np 2 -type int,float
```

To run test(s) with communicator MPI_COMM_WORLD and MPI_COMM_SELF:

```
mpitest -np 2 -context world,self
```

13.3 C specific configurations

All of the C specific configuration are in $MPITEST_HOME/include/mpitest_def.h

There are a number of things user can configure in this file.

13.3.1 C optional datatypes

Long long is optional C datatypes per *MPI Standard v1.1*. There is a compiler directive that turn on or off the support for the datatype:

Directive	C Optional Datatype
MPITEST_longlong_def	long long

User will need to add MPITEST_longlong, and in the configuration file ($MPITEST_HOME/include/mpitest_cfg.h) in order to configure tests to use the optional type(s) in testing.

The following is an example of the types array configured to run test with all C optional datatypes:

```
int MPITEST_types[] =
{
MPITEST_longlong, MPITEST_long_double, MPITEST_END_TOKEN
};
```

13.3.2 C Configuration arrays size

MPITEST_CONFIG_ARRAY_SIZE is defined to be the maximum number of entries each configuration array). It is set to 128 by default but user can increase or decrease it as desired.

13.4 Fortran specific configurations

All of the Fortran specific configurations are in $MPITEST_HOME/include/foptions.h.

13.4.1 Fortran command line option processing

The validation suite does not require the underlying Fortran compiler used to support Fortran command line options handling. It is set up to handle Fortran command line options if it is available. With command line options, the run time environment is much more friendly. Changing test parameter(s), turning on debugging output, etc could be handled by command line options. Fortran command line options are handled by non-Fortran 77 standard function IARGC() and GETARG() (both available in GNU Fortran compiler) in the mpi test library routine MPITEST_GET_PARAMETER() and it assumes that all command line option parameters be passed to all ranks using these functions. For platforms that offers other ways of handling Fortran command line options processing, users are free to make modification to the routine. In any case, all user can turn on / off the Fortran command line options processing using the compiler directive MPITEST_FCMDLINE, so this will allow all users to use the suite quickly without the need to make a lot of modification at first. User should make sure that MPITEST_FCMDLINE is #defined to turn on Fortran command line options processing. It is off by default for portability.

13.4.2 Fortran optional datatypes

There are a number of Fortran optional datatypes as defined in the MPI Standard but does not have to be supported by all MPI implementations. To turn on support for each optional datatype, make sure they are #defined in $MPITEST_HOME/include/foptions.h:

Table 6: Compiler directive for turning on/off Fortran optional datatype support

Directive	Fortran Optional Datatype
MPITEST_FREAL2_DEF	REAL*2
MPITEST_FREAL4_DEF	REAL*4
MPITEST_FREAL8_DEF	REAL*8
MPITEST_FDOUBLE_COMPLEX_DEF	DOUBLE COMPLEX
MPITEST_FINTEGER1_DEF	INTEGER*1
MPITEST_FINTEGER2_DEF	INTEGER*2
MPITEST_FINTEGER4_DEF	INTEGER*4

Users will need to add MPITEST_REAL4, MPITEST_REAL8, MPITEST_DOUBLE_COMPLEX, MPITEST_INTEGER1, MPITEST_INTEGER2, and / or MPITEST_INTEGER4 to the type array in the configuration file ($MPITEST_include/mpitest_cfgf.h) to configure each test to use the optional datatypes

The following is an example of the types array configured to run test with all Fortran optional datatypes:

```
      DATA MPITEST_TYPES /
     & MPITEST_INTEGER1, MPITEST_INTEGER2, MPITEST_INTEGER4,
```

```
& MPITEST_REAL2
&  MPITEST_REAL4, MPITEST_REAL8,
& MPITEST_DOUBLE_COMPLEX,
& MPITEST_END_TOKEN, 0, 0,  ... ,0 /
```

13.4.3 Display format for different datatypes

The following are the display formats for various Fortran datatypes. They are also defined in $MPITEST_HOME/include/foptions.h.

MACRO	Default Value
INT_FMT	I10
REAL_FMT	F10.2
COMPLEX_FMT	F10.2, F10.2
DOUBLE_FMT	D20.4
CHAR_FMT	A
LOG_FMT	L1
INT1_FMT	I3
INT2_FMT	I5
INT4_FMT	I10
REAL2_FMT	F10.2
REAL4_FMT	F10.2
REAL8_FMT	F10.2
DCOMPLEX_FMT	D20.4, D20.4

These are created so that output format could be easily changed in various system environments.

13.4.4 Test buffer type

In order for tests to be able to use the same buffer for various MPI datatypes, the buffer is defined to be of type MPITEST_BUF_TYPE in most Fortran tests. MPITEST_BUF_TYPE is a cpp MACRO (in $MPITEST_HOME/include/foptions.h) which by default is defined to be DOUBLE COMPLEX. MPITEST_BUF_EXTENT is defined to be the extent (in bytes) of the Fortran type defined in MPITEST_BUF_TYPE. There is also a Fortran constant MAX_BUFF_SIZE (in $MPITEST_HOME/include/mpitestf.h) which determines the number of MPITEST_BUF_TYPE elements in each buffer. Each buffer used in most tests is considered to be typeless contiguous buffer space. Data stored in the buffer could be packed together regardless of whether the actual datatype used is the same as the value defined in MPITEST_BUF_TYPE. The values of MPITEST_BUF_TYPE and MAX_BUFF_SIZE together determine the total buffer space (in bytes) available for most Fortran tests. Most Fortran library routines use both MAX_BUFF_SIZE and MPITEST_BUF_EXTENT to determine if the buffer space is sufficient for testing with various datatypes. MPITEST_BUF_EXTENT is needed since there is no portable and built in facility in Fortran 77 to calculate the size of a type. In general, MPITEST_BUF_TYPE must be defined to be the largest datatype supported by the Fortran compiler used and MPITEST_BUF_EXTENT must be defined to be the extent of the largest datatype.

13.5 Changing library routines

The MPI Validation suite comes with a set of common library routines. They handle the following tasks:

- Initialization
- All I/O and command line processing
- Parsing of configuration file options (message sizes, data types, communicators)for functional tests
- Most initialization of data areas
- Most verification of data received
- Results reporting

They should suit most purposes but some users may wish to modify it for some system specific support. This section would describe where each feature of the library is / are handled. The following table lists some of the library routines and what they are responsible for. This should serve as a good index for users who may wish to modify or develop new ones to suit their purpose. For more documentation on each of these library routine, please refer to the source code of the library in $MPITEST/lib/libmpitest.c for the C version or $MPITEST_HOME/lib/libmpitestf.F for the Fortran version.

Table 7: Library routines

Routine name	Description
MPITEST_buffer_errors()	Check data received in buffer
MPITEST_buffer_errors_ov()	Make sure receiver does not receive more data than expected
MPITEST_get_buffer()	Allocate a buffer space
MPITEST_init()	Initialize library - all tests call this immediately after MPI_Init()
MPITEST_init_buffer()	Initialize a buffer with data

Table 7: Library routines

Routine name	Description
MPITEST_free_communicator()	Free a communicator created
MPITEST_get_comm_size()	Get the size of an input communicator.
MPITEST_get_communicator()	Get a communicator using input from the configuration file
MPITEST_get_datatype()	Get a datatype using input from configuration file
MPITEST_get_parameter()	Get command line options passed to the test binary executable
MPITEST_get_max_message_length()	Get the maximum message length defined in the configuration file
MPITEST_get_message_length()	Get the current message length defined in the configuration file
MPITEST_message()	Display a message to the preset I/O channel. If the input is MPITEST_FATAL, the test will abort
MPITEST_num_datatypes()	Get the number of datatypes defined in the configuration file
MPITEST_num_message_lengths()	Get the number of message length defined in the configuration file
MPITEST_report()	Report test result(s) and print a message to the preset I/O channel

13.6 Adding platform specific constants

All MPITEST defined constants works correctly with their preset values. In the event a new constant needs to be added, the original preset values of all MPITEST defined constants should not be changed or some of the tests may not work. For example, some tests count on the fact that MPITEST_derived1 is greater than all of the predefined basic types (e.g. MPITEST_int). Changing that will cause some tests to fail unreasonably or skip portion of the intended functional testing.

14.0 Tagging a Test Run

The driver environment offers a way to specially tag a test run with a user provided tag.

The user tag provided will be concatenated into filenames of all the test generated files. This may be useful for test run(s) with a special hardware / software configuration, *etc*. The option -tag of the mpitest driver takes one ascii (no space) string argument. The input tag will be combined into all test generated files to distinguish it from other test run(s) possibly with or without a different user tag.

15.0 Removing Test Generated files

In each test run, the driver environment generates a number of files. This include the make output, the test log file, the test result file, run script, binary executable, object file, mpitest libraries, and some platform specific files. The driver environment offers a convenient way to quickly remove all of the test generated file(s).

mpitest -cleanall

The above command will not submit any test to be run but rather remove both the C and Fortran version of the test libraries and all of the test generated files of tests residing in the current directory. Combining the above command with the -testlist option of the test driver will clean the tests in a specified list of tests in a file.

mpitest -cleanall -testlist /$MPITEST_HOME/testlist

The testlist file should list one test in each line. Each entry could be either a relative or absolute path name.

16.0 Adding New Test(s)

16.1 Test with a single source file

In order for newly added tests to be successfully used by the test driver environment, there are a few compliance precautions. Each new test should reside in its own directory and should contain a run script, a Makefile, and one or more source files. All source file(s) in a single test must contain source of only one language - All in C or all in Fortran. All C test source should be suffixed by .c and Fortran source by suffixed by .F. The name of the directory of the test will be the testname used by the test driver. The name of the run script should be <testname>.sh or <testname>.csh or <testname>.ksh depending on which shell is being used.

The first step in creating a new test is to find a proper location in the MPITEST/Test hierachy to put the test in. Tests are grouped by language (C or Fortran), mpi library functional area (collective, derived datatype, *etc*), and test type (functional or error).

The next step is to find a suitable test name that describe the new test to be written. It will help the test users search your tests easily if the test name describe very briefly what the test is testing. we have used the name of the MPI functions combined with a few word separated with the underscore describing the testing area for functional test and the name of MPI library function and suffixed it with _err<a number> for our error test.

Create a test sub-directory using the name chosen. For each test, a Makefile and a run script will needed to be created. For convenience, $MPITEST_HOME/template contains some templates to be used.

The following shows the Makefile for a test with one source:

```
##
#  Makefile for MPI Test
#

MPITEST_LOCALLIB=
MPITEST_LOCALOBJ=

sources:        ${MPITEST_TESTNAME}.sh node.c
execs:          ${MPITEST_TESTNAME}.sx node.${MPITEST_ARCH}_bx
logs::          ${MPITEST_TESTNAME}.${MPITEST_ARCH}_${MPITEST_NPROCS}_L
results::       ${MPITEST_TESTNAME}.${MPITEST_ARCH}_${MPITEST_NPROCS}_R

node.${MPITEST_ARCH}_bx: ${MPITEST_CONFIG_FILE} ${MPITEST_LOCALOBJ}
```

The make rules are controlled by the master makefile (Please see section 17). The following shows a sample Bourne shell run script:

```
#!/bin/sh
##########################################################################
#
# Run the ${MPITEST_TESTNAME} test
#
# Usage:  ${MPITEST_TESTNAME}.sh
#
# MPI operator test
#
##########################################################################
echo ""
echo "*****  BEGIN ${MPITEST_TESTNAME} TEST  *****"
echo ""

${MPITEST_RM:-rm} -f $MPITEST_SCRATCH_FILE
${MPITEST_TOUCH:-touch} $MPITEST_SCRATCH_FILE

#
#  Check essential environment variables
#
if test -z "$MPITEST_IS_ENV_DEF" > ${MPITEST_DEVNULL:-/dev/null}
then
        echo "MPITEST environment variables are not defined, terminating
test, ${MPITEST_FAIL:-FAILED}"  >> ${MPITEST_SCRATCH_FILE:-scratch}
        ${MPITEST_CAT:-cat} ${MPITEST_SCRATCH_FILE:-scratch}
        exit 4
```

```
fi
. $MPITEST_SHLIB

#
# Some pre processing(s) add here (if needed)
#

#
# Get start time for the test
#
GetStartTime

#
#  Run test
#

$MPITEST_BINDIR/startjob.$MPITEST_ARCH -appname node.${MPITEST_ARCH}_bx
\ $MPITEST_LOCAL_OPTIONS > ${MPITEST_SCRATCH_FILE:-scratch} 2>&1

#
# Get run info
#
GetRunInfo

#
# Capture application output
#
${MPITEST_CAT:-cat} ${MPITEST_SCRATCH_FILE:-scratch}

#
#  Report test results
#

echo ""

expectedPassCnt=1
actualPassCnt=0

if $MPITEST_GREP "$MPITEST_FAIL" ${MPITEST_SCRATCH_FILE:-scratch} > \
$MPITEST_DEVNULL
then
        echo "TEST_RESULT: F (found $MPITEST_FAIL) $runInfo"
else
        actualPassCnt=`$MPITEST_GREP -c "$MPITEST_PASS"
${MPITEST_SCRATCH_FILE:-scratch}`
       if test "$actualPassCnt" != "$expectedPassCnt" > $MPITEST_DEVNULL
        then
              echo "TEST_RESULT: F (wrong $MPITEST_PASS count, Expected:
$expectedPassCnt, Actual: $actualPassCnt) $runInfo"
        else
                echo "TEST_RESULT: P $runInfo"
```

```
        fi
fi

echo ""
echo "*****  END ${MPITEST_TESTNAME} TEST  *****"
echo ""
```

This run script is executed by the mpitest driver. It sets up the required environment and submits the test binary executable using the job submission script for the platform being tested. After the test is completed, the run script will check for errors and report test pass or failure. The way the test script reports errors is to first find any "FAILED" string in the test log file, which will cause the test to fail. Then it counts the occurrences of the "PASSED" string, which in this case must be 1 or the test will fail. In most case, the sample run script can be used as is unless something special needs to be done.

The last step is to create test program file(s). For C tests, the main test program (with main()) must be name node.c. For Fortran tests, the main test program must be named node.f or node.F depending on whether cpp directive(s) or macro(s) is / are used in the source.

For tests that require linking with another library (e.g. the C math library), the MPITEST_LOCALLIB in the Makefile would need to be edited. The following makefile shows a test that requires linking with the C math library:

```
#
#  Makefile for MPI Test
#


MPITEST_LOCALLIB=-lm

MPITEST_LOCALOBJ=


sources:        ${MPITEST_TESTNAME}.sh node.c


execs:          ${MPITEST_TESTNAME}.sx
node.${MPITEST_ARCH}${MPITEST_USERTAG}_bx

logs::
${MPITEST_TESTNAME}.${MPITEST_ARCH}_${MPITEST_NPROCS}${MPITEST_USERTAG}
_L


results::
${MPITEST_TESTNAME}.${MPITEST_ARCH}_${MPITEST_NPROCS}${MPITEST_USERTAG}
_R


node.${MPITEST_ARCH}_bx: ${MPITEST_CONFIG_FILE} ${MPITEST_LOCALOBJ}
```

16.2 Test with multiple source files

The mpitest driver environment supports tests with multiple source files so long as they are all using one language (all C or all Fortran). With multiple source files, one should follow all the above steps for creating a test with a single source file. Some changes to the Makefile of the test would then need to be made:

The following shows a Makefile for a test with 2 source files: node.c and reduce.c

```
#
#  Makefile for MPI Test
#

MPITEST_LOCALLIB=
MPITEST_LOCALOBJ= reduce.${MPITEST_ARCH}${MPITEST_USERTAG}_o

sources:        ${MPITEST_TESTNAME}.sh node.c reduce.c

execs:          ${MPITEST_TESTNAME}.sx
node.${MPITEST_ARCH}${MPITEST_USERTAG}_bx
logs::
${MPITEST_TESTNAME}.${MPITEST_ARCH}_${MPITEST_NPROCS}${MPITEST_USERTAG}
_L


results::
${MPITEST_TESTNAME}.${MPITEST_ARCH}_${MPITEST_NPROCS}${MPITEST_USERTAG}
_R


node.$(MPITEST_ARCH)${MPITEST_USERTAG}_bx: $(MPITEST_CONFIG_FILE) \
$(MPITEST_LOCALOBJ)
```

Except for the main source program source (node.c, node.f, or node.F), all the other source file(s) would be listed as object file(s) in MPITEST_LOCALOBJ and MPITEST_LOCALOBJ will be listed as a dependency for making the main test binary executable (node.$(MPITEST_ARCH)$(MPITEST_USERTAG)_bx). Hence, all the source file(s) except the main source program will first be compiled into the corresponding object file(s). The main test program will then be compiled and linked with all the dependent object file(s) producing the final test binary executable.

Please refer to section 12.0 for details on interpreting and analyzing test result.

17.0 Porting the Driver Environment to Other Platforms

Before porting the driver environment to a different platform, a unique name for the platform needs to be decided. For the platforms that come with the package, they loosely follow a naming convention:

<Vendor / MPI implementation name>_<system name>_<MPI implementation specific tag>

e.g. INTEL_paragon_nx

<Vendor name>_<system name>_<nx implementation>

e.g. MPICH_sun4_ch_p4

<MPI implementation name>_<system name>_<ch_p4 device>

Different platforms (architecture / MPI implementation) are set up differently. In order to have a portable test driver environment, all the platform specific environments have been factored into 2 platform specific files: an environment file and a job submission script. For each new platform, a new environment file and a new job submission script will need to be created in order for the validation suite to be used on the new platform.

The names of the environment files and the job submission scripts that comes with the package follow the following naming convention:

Environment files:

mpitest.<unique platform name>.env

e.g. mpitest.INTEL_paragon_nx.env, mpitest.MPICH_sun4_ch_p4.env

job submission script:

startjob.<unique platform name>

e.g. startjob.INTEL_paragon_nx, startjob.MPICH_sun4_ch_p4

These are useful examples of what needs to be implemented.

17.1 Environment file

Each environment file contains 3 basic categories of information specific to a platform: absolute names of each system utilities, system specific environment, and the user defined environment. All the information is stored in environment variables having names prefixed with the word MPITEST.

17.1.1 Name of each system utility

The first section contains the absolute path names of each system utility. An example is the echo command utility:

MPITEST_ECHO=/usr/bin/echo

These environment variables not only specify the names of each system utility but also where the command executable lives. This allow the user to specify a particular one to be used should there be more than one available in a system.

17.1.2 System specific environment

The second section of the environment file is where the other system specific information is stored. An example is where the MPI Validation common routines are:

MPITEST_CLIB_NAME="${MPITEST_CLIB_HOME}/libmpitest_${MPITEST_ARCH}.a"

17.1.3 User defined environment.

This is the section where the user defined environment variables can be added. The standard environment files that come with the package do not have any user defined environment.

17.2 Job submission script

Different systems have different ways to submit a parallel application. Different systems may also offer different run-time system support. These aspects have been factored into the job submission script. Each job submission script is to process any defined options, and submit the application requested with the input command line argument(s).

18.0 Command Line Options

There are a number of command line options to the driver environment. To list all of the available options for the platform set up, type mpitest -help.

Input command line options to the mpi test driver are grouped and ordered into 3 sets of options: test driver specific options, platform specific options, and job specific options. Each input command line option is processed in 3 levels: mpitest driver, job submission script, and the binary executable for each test. Any unrecognized option being processed at any level will cause the current unrecognized option and the rest of the options to be passed to the next level for further processing. Therefore, it is very important the options be inputed and grouped in the correct order. The first set of options is specific to the mpitest driver.

Test driver options:

-arch <platform name>	Platform name
	(e.g. -arch INTEL_paragon_nx)
-build	Only build executable(s), do not run test
-cleanall	Remove all logs, results, make outputs & executables

-config <file>	Configuration file
	(e.g. -config "/usr/local/MPITEST/include/mpitest_cfg.h)
-clibonly	Build or clean mpitest C library only
-envfile <file>	Environment file
	(e.g. -envfile "/usr/local/MPITEST/bin/my.env)
-flibonly	Build, or clean mpitest Fortran library only
-help	Help option, this message
-mmakefile <file>	Alternate master Makefile
	(e.g. -mmakefile /usr/local/MPITEST/bin/Makefile.master)
-mpiclib <lib opt>	MPI C library link option
	(e.g. -mpiclib "-L/usr/local/MPITEST/lib -lmpi)
-mpiflib <lib opt>	MPI Fortran library link option
	(e.g. -mpiclib "-L/usr/local/MPITEST/lib -lmpi)
-np <nprocs>	Number of process(es)
	(e.g. -np 4)
-stdoutfile	Each process write all stdout to a different file
-tag <user_tag>	User tag
	(e.g. -tag mytestrun)
-testclib <lib dir>	C test library directory
	(e.g. -testclib /usr/local/MPITEST/lib)
-testflib <lib dir>	Fortran test library directory
	(e.g. -testclib /usr/local/MPITEST/lib)
-testlist <test list file>	Test(s) to be run
	(e.g. -testlist /usr/local/MPITEST/testlist)
-verbose <level>	Verbose option
-version	Shows version

<u>Platform specific options for Paragon Supercomputer:</u>

-leave_stdout	Do not delete the redirected stdout file(s)
-mkpart	Create partition for test

<u>Platform specific options for Sun4/MPICH</u>

-leave_pg	Do not delete the pg file
-leave_stdout	Do not delete the redirected stdout file(s)
-machinefile <file>	Machine file
	(e.g. -machinefile /usr/local/MPITEST/sun4.list)
-nolocal	Do not start process locally
-p4pg <file>	Process group file
	(e.g. -p4pg /usr/local/MPITEST/p4file)

Test binary executable options: (C only)

-verbose <level>	Verbosity level
	(e.g. -verbose 2)
-length <len str>	Message length (in bytes)
	(e.g. -length 100)
-context <context str>	Context string setting
	(e.g. -context world,self)
-type <type str>	Message datatype
	(e.g. -type int,float)

For example to submit test(s) using 3 ranks and running on a newly created partition on a Paragon supercomputer:

```
mpitest -np 3 -mkpart
```

Or, to submit test using 10 ranks, compiling only the C library, not starting process locally using message length of 100 byte in a Sun4/MPICH environment:

```
mpitest -np 10 -clibonly -nolocal -length 100
```

To avoid confusion, user may also use the option separator --. Thus with the option separator, the above 2 examples become:

```
mpitest -np 3 -- -mkpart
mpitest -np 10 -clibonly -- -nolocal -- -length 100
```

19.0 Driver Environment Variables

The following is a listing of all driver environment variables. The definitions are in $MPITEST_HOME/bin/mpitest.<platform name>.env. The shown assigned values are the default driver environment for the Intel Paragon™ Supercomputer system. Each value can be dynamically overridden at run time by an explicit environment set (e.g. setenv in csh)

Table 8: System utilities

System Utilities	Value
MPITEST_AR	ar860
MPITEST_AWK	awk
MPITEST_BASENAME	basename
MPITEST_CC	icc
MPITEST_CFLAGS	-g
MPITEST_CAT	cat
MPITEST_CPP	CC
MPITEST_DATE	date
MPITEST_ECHO	/usr/bin/echo
MPITEST_EXPR	expr
MPITEST_F77	if77
MPITEST_FFLAGS	-g
MPITEST_GREP	grep
MPITEST_HOSTNAME	hostname
MPITEST_LN	ln
MPITEST_LS	ls
MPITEST_MAKE	make
MPITEST_MORE	more
MPITEST_MV	mv
MPITEST_RANLIB	ranlib
MPITEST_RM	rm
MPITEST_RSH	rsh
MPITEST_SED	sed
MPITEST_SYNC	sync
MPITEST_TOUCH	touch
MPITEST_WC	wc

Table 9: System specific environment

System specific environment	Description	Value
MPITEST_FLIB_HOME	Directory where all the common Fortran library routines for the suite live.	${MPITEST_HOME}/lib
MPITEST_FLIB		-L${MPITEST_FLIB_HOME} -lmpitestf_${MPITEST_ARCH}
MPITEST_INCLUDE_DIR	Directory where all the header files for the suite live.	${MPITEST_HOME}/include
MPITEST_MPI_INCLUDE	Directory where all the mpi header for the implementation used live.	
MPITEST_INCLUDE_OPTION	The include options to the compilers.	-I. -I${MPITEST_INCLUDE_DIR} -I${MPITEST_MPI_INCLUDE_DIR}
MPITEST_KSHLIB	ksh library functions.	${MPITEST_HOME}/lib/mpitest_lib.ksh
MPITEST_LOCAL_OPTIONS	Local options to the compiler.	
MPITEST_LOCKFILE	Name of a test lock file.	LOCK.mpitest
MPITEST_MMAKEFILE	Name of the master makefile.	${MPITEST_HOME}/bin/Makefile.master
MPITEST_MPICLIB	C link option for linking with the mpi library used.	-lmpi

Table 9: System specific environment

System specific environment	Description	Value
MPITEST_MPIFLIB	Fortran link option for linking with the mpi library used.	-lmpi
MPITEST_NPROCS	Number of ranks to be used when executing tests.	2
MPITEST_USERTAG	User specified tag	
MPITEST_MAKELOG	Log file when making test.	makelog.${MPITEST_ARCH}_${MPITEST_NPROCS}${MPITEST_USERTAG}
MPITEST_PASS		PASSED
MPITEST_SCRATCH_FILE	Scratch file name for each test (I/O of each ranks is redirected here).	scratch.{MPITEST_ARCH}_${MPITEST_NPROCS}
MPITEST_MAKESHELL	Shell used in make	/bin/sh
MPITEST_SHLIB	sh library functions.	${MPITEST_HOME}/lib/mpitest_lib.sh
MPITEST_SIGUP	Signal number for hang up.	1
MPITEST_SIGINT	Signal number for interrupt.	2
MPITEST_TESTLIST	A file which store all the tests to be executed, cleaned, or built.	
MPITEST_USERLIB	User library for the system.	-nx
MPITEST_VERBOSE	Verbosity level when running test(s) (1, 2, or "NULL")	

A Frequently Asked Questions

General

Some C or Fortran tests cannot be compiled successfully.

Possible Explanation:

A non-ANSI compliant C compiler or non-Fortran 77 compliant compiler is being used.

Possible solution:

Make sure a ANSI compliant C compiler and a Fortran 77 compiler are being used. Please refer to Section 6.2.3 for more details on compilers requirements.

What does a configuration file contain?

Possible solution:

Please refer to section 13.2.1 for more details.

How do I find specific options which are available for a certain platform?

Possible solution

If MPITEST_ARCH is set up to be the platform you wish to find out the run options available, type:

```
mpitest -help
```

Otherwise, type

```
mpitest -arch <put the unique platform name here> -help
```

When running tests in an automounted file system, some test executable are being reported as not found.

Possible Explanation:

If the file system where some of the test executables are located in a soft mounted file system, it may be that the mounting point was dropped owing to insufficient I/O activity.

Possible solution:

Manually open a window to the machine running the test and do a cd to the mounting point where the executables are. This would cause the mounting point to stay busy and hence will not be dropped by the automounter.

How do I configure the test suite to use a particular message length(s), message type(s), or communicator?

Possible solution:

Please refer to section 13.0 for advanced configuration.

When I submit tests to run in the current directory, I see some warnings saying Makefile is not found in some directory.

Possible Explanation:

If mpitest is complaining about some subdirectories that the user created and is not meant to be a test, the warning can be ignored. When the mpitest driver searches for tests to run in a directory, it looks for all subdirectories. If a subdirectory has no Makefile, the above warning will be printed. This is just so that the user will be notified if some tests are missing a Makefile.

Possible solution:

Try not to create any additional subdirectory in the MPITEST/Test/*/*/* hierachy.

I am running tests with -verbose 2. The test is not hanging but is taking a long time to run.

Possible Explanation:

When using -verbose 2, most of the tests takes considerably longer to run than without. The reason behind is because of the additional I/O each mpi process has to do. Some of the tests could take hours or longer to run especially if a large number of ranks is used.

Possible solution:

We recommend, in a normal regression test run, not to use -verbose option and use rerun tests that failed with -verbose 1 or -verbose 2 for tests analysis.

One of the tests is reporting that it is skipping length = 0, or skipping communicator with size < 2, or skipping intercommunicator, is my set up incorrect?

Possible Explanation:

These are informational message only and do not constitute a test failure. For some of the tests, they require non-zero message length, or communicator with at least 2 ranks (for sending and receiving for example), or does not apply to intercommunicators (collective operations).

Fortran specific

How do I configure the Fortran test to use optional Fortran datatype(s)?

Possible solution:

Please refer to section 13.0 for advance configuration.

I run Fortran tests with some command line options which seems to be ignored

Possible Explanation & Solution:

There are a few things to be checked. First, be sure that the Fortran specific options are entered as the last group of options in the mpitest driver. Second, be sure that the option names are typed correctly. Any typo in option names will be ignored and no error will be reported. Lastly, be sure that the Fortran compiler used supports both GETARG() and IARGC() and edit $MPITEST_HOME/MPITEST/include/foptions.h so that MPITEST_FCMDLINE macro is #defined (uncomment it if necessary).

If the Fortran compiler used does not support GETARG() or IARGC() but does provide some other way to retrieve command line options, user may wish to modify the MPITEST_GET_PARAMETER() routine in $MPITEST_HOME/MPITEST/lib/libmpitestf.F in order to turn on Fortran command line processing.

If the Fortran compiler used does not support GETARG() and / or IARGC() and it does not offer any way to retrieve command line options, Fortran command line option processing of the mpitest driver will not be supported. Please comment out the definition of MPITEST_FCMDLINE in $MPITEST_HOME/MPITEST/include/foptions.h for consistency.

Please refer to section 18.0 for details on various command line options of the mpitest driver and section 13.4.1 for more details on Fortran command line options.

I have put a local mpitest_cfgf.h in a local test directory but the test appear to use the default one instead.

Possible Explanation:

The mpi validation suite test driver uses the -I option of the host compiler to specify which include files to be used in compilation of test sources. In some host language compilers, the -I option is applied only for #include but not the Fortran INCLUDE or vice versa.

Possible solution:

Use a GNU-compliant Fortran compiler.

What is MPITEST_BUF_TYPE / MPITEST_BUF_EXTENT?

Possible Explanation:

Please refer to section 13.4.4

B Tests Index

This section briefly explain the testing area of each test in the validation suite. It does not serve as a full test specification for each test but rather an index for locating tests for a particular testing area. For full detailed documentation of each test, please refer to the corresponding test source. The test are grouped by the functional area, test type. and they are alphabetically ordered. User may find that, in some instances, there is no test associated with a certain MPI calls. This is not because there is no test written for it but rather that the call is embedded in other tests.

B.1 C Tests

B.1.1 Blocking Operations

Table 10: C Blocking functional tests

Test path	Description
c/blocking/functional/MPI_Bsend_ator	Calling MPI_Bsend() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Bsend_null	Calling MPI_Bsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/blocking/functional/ MPI_Bsend_overtake	Calling MPI_Bsend() / MPI_Recv() to send a sequence of message and make sure they are received in the order they were sent.
c/blocking/functional/MPI_Bsend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.

Table 10: C Blocking functional tests

Test path	Description
c/blocking/functional/MPI_Bsend_rtoa	Calling MPI_Bsend() / MPI_Recv() and MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Bsend_rtoaq	Calling MPI_Bsend() / MPI_Recv() to send message from root to all other ranks in communicators. Test will retry if the buffer is full. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Recv_comm	Calling MPI_Recv() with various communicators created and make sure the messages are received in the order sent for each communicator (not the order the recvs were posted)
c/blocking/functional/MPI_Recv_null	Calling MPI_Recv() with source of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/blocking/functional/MPI_Recv_pack	Calling MPI_Recv() verifying that every datatype can be sent as MPI_PACKED. Test loops through various communicators, message types, and message lengths.
c/blocking/functional/MPI_Rsend_null	Calling MPI_Rsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.

Table 10: C Blocking functional tests

Test path	Description
c/blocking/functional/MPI_Rsend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/blocking/functional/MPI_Rsend_rtoa	Calling MPI_Rsend() / MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Send_ator	Calling MPI_Send() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Send_ator2	Calling MPI_Send() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Send_fairness	Quality of implementation test testing the fairness of the sending protocol dealing with multiple MPI_Send() from various ranks to the same destination.
c/blocking/functional/MPI_Send_flood	Quality of implementation test testing the behavior of the sending protocol dealing with many MPI_Send() farm various ranks to the same destination.

Table 10: C Blocking functional tests

Test path	Description
c/blocking/functional/MPI_Send_null	Calling MPI_Send() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/blocking/functional/MPI_Send_off	Calling MPI_Send() / MPI_Recv() to send message from all ranks in communicators to a selected root with byte offsets. Test loops through various communicators, message datatypes, message lengths and root' ranks.
c/blocking/functional/ MPI_Send_overtake	Calling MPI_Send() / MPI_Recv() to send a sequence of messages and make sure they are received in the order they were sent.
c/blocking/functional/MPI_Send_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/blocking/functional/MPI_Send_rtoa	Calling MPI_Send() / MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Send_self	Calling MPI_Send() / MPI_Recv() to send message from a rank to itself. Test loops through various communicators.

Table 10: C Blocking functional tests

Test path	Description
c/blocking/functional/MPI_Ssend_ator	Calling MPI_Ssend() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/blocking/functional/MPI_Ssend_null	Calling MPI_Ssend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/blocking/functional/MPI_Ssend_overtake	Calling MPI_Ssend() and MPI_Recv() to send a sequence of messages and make sure they are received in the order they were sent.
c/blocking/functional/MPI_Ssend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/blocking/functional/MPI_Ssend_rtoa	Calling MPI_Ssend() / MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 11: C Blocking error tests

Test path	Description
c/blocking/error/MPI_Bsend_err1	Invalid rank (negative).
c/blocking/error/MPI_Bsend_err2	Invalid rank (too large).

Table 11: C Blocking error tests

Test path	Description
c/blocking/error/MPI_Bsend_err3	Invalid communicator (MPI_COMM_NULL).
c/blocking/error/MPI_Bsend_err4	Freed communicator.
c/blocking/error/MPI_Bsend_err5	Invalid tag (negative).
c/blocking/error/MPI_Bsend_err6	Invalid tag (> MPI_TAG_UB)
c/blocking/error/MPI_Bsend_err7	Invalid count (negative).
c/blocking/error/MPI_Bsend_err8	Invalid datatype (MPI_DATATYPE_NULL).
c/blocking/error/MPI_Bsend_err9	Mismatching types.
c/blocking/error/MPI_Bsend_err10	Message size > attached buffer size.
c/blocking/error/MPI_Bsend_err11	Message with no attached buffer
c/blocking/error/MPI_Buffer_attach_err1	Attach more than 1 buffer to one process.
c/blocking/error/MPI_Buffer_attach_err2	Invalid length (negative).
c/blocking/error/MPI_Buffer_detach_err1	An non-attached buffer.
c/blocking/error/MPI_Recv_err1	Invalid rank (negative).
c/blocking/error/MPI_Recv_err2	Invalid rank (too large).
c/blocking/error/MPI_Recv_err3	Invalid communicator (MPI_COMM_NULL).
c/blocking/error/MPI_Recv_err4	Freed communicator.
c/blocking/error/MPI_Recv_err5	Invalid tag (negative).
c/blocking/error/MPI_Recv_err6	Invalid tag (> MPI_TAG_UB)
c/blocking/error/MPI_Recv_err7	Invalid count (negative).
c/blocking/error/MPI_Recv_err8	Invalid datatype (MPI_DATATYPE_NULL).
c/blocking/error/MPI_Recv_err9	Message size > recv count.
c/blocking/error/MPI_Rsend_err1	Invalid rank (negative).
c/blocking/error/MPI_Rsend_err2	Invalid rank (too large).

Table 11: C Blocking error tests

Test path	Description
c/blocking/error/MPI_Rsend_err3	Invalid communicator (MPI_COMM_NULL).
c/blocking/error/MPI_Rsend_err4	Freed communicator.
c/blocking/error/MPI_Rsend_err5	Invalid tag (negative).
c/blocking/error/MPI_Rsend_err6	Invalid tag (> MPI_TAG_UB)
c/blocking/error/MPI_Rsend_err7	Invalid count (negative).
c/blocking/error/MPI_Rsend_err8	Invalid datatype (MPI_DATATYPE_NULL).
c/blocking/error/MPI_Rsend_err9	Mismatching types.
c/blocking/error/MPI_Send_err1	Invalid rank (negative).
c/blocking/error/MPI_Send_err2	Invalid rank (too large).
c/blocking/error/MPI_Send_err3	Invalid communicator (MPI_COMM_NULL).
c/blocking/error/MPI_Send_err4	Freed communicator.
c/blocking/error/MPI_Send_err5	Invalid tag (negative).
c/blocking/error/MPI_Send_err6	Invalid tag (> MPI_TAG_UB)
c/blocking/error/MPI_Send_err7	Invalid count (negative).
c/blocking/error/MPI_Send_err8	Invalid datatype (MPI_DATATYPE_NULL).
c/blocking/error/MPI_Send_err9	Mismatching types.
c/blocking/error/MPI_Ssend_err1	Invalid rank (negative).
c/blocking/error/MPI_Ssend_err2	Invalid rank (too large).
c/blocking/error/MPI_Ssend_err3	Invalid communicator (MPI_COMM_NULL).
c/blocking/error/MPI_Ssend_err4	Freed communicator.
c/blocking/error/MPI_Ssend_err5	Invalid tag (negative).
c/blocking/error/MPI_Ssend_err6	Invalid tag (> MPI_TAG_UB)
c/blocking/error/MPI_Ssend_err7	Invalid count (negative).

Table 11: C Blocking error tests

Test path	Description
c/blocking/error/MPI_Ssend_err8	Invalid datatype (MPI_DATATYPE_NULL).
c/blocking/error/MPI_Ssend_err9	Mismatching types.

B.1.2 Collective Operations

Table 12: C Collective functional tests

Test path	Description
c/collective/functional/MPI_Allgather	Calling MPI_Allgather() looping between various intracommunicators, various message data types and various message data lengths.
c/collective/functional/MPI_Allgatherv	Calling MPI_Allgatherv() looping between various intracommunicators, various message data types and various message data lengths.
c/collective/functional/MPI_Allreduce	Calling MPI_Allreduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various predefined reduction operations.
c/collective/functional/ MPI_Allreduce_loc	Calling MPI_Allreduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).
c/collective/functional/ MPI_Allreduce_user	Calling MPI_Allreduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations

Table 12: C Collective functional tests

Test path	Description
c/collective/functional/MPI_Alltoall	Calling MPI_Alltoall() looping between various intracommunicators, various message data types and various message data lengths.
c/collective/functional/MPI_Alltoallv	Calling MPI_Alltoallv() looping between various intracommunicators, various message data types and various message data lengths.
c/collective/functional/MPI_Barrier	Calling MPI_Barrier() using various intracommunicators.
c/collective/functional/MPI_Bcast	Calling MPI_Bcast() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
c/collective/functional/MPI_Gather	Calling MPI_Gather() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
c/collective/functional/MPI_Gatherv	Calling MPI_Gatherv() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
c/collective/functional/MPI_Reduce	Calling MPI_Reduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various predefined reduction operations
c/collective/functional/MPI_Reduce_loc	Calling MPI_Reduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).

Table 12: C Collective functional tests

Test path	Description
c/collective/functional/ MPI_Reduce_scatter	Calling MPI_Reduce_scatter() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various pre-defined reduction operations
c/collective/functional/ MPI_Reduce_scatter_loc	Calling MPI_Reduce_scatter() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).
c/collective/functional/ MPI_Reduce_scatter_user	Calling MPI_Reduce_scatter() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations
c/collective/functional/MPI_Reduce_user	Calling MPI_Reduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations
c/collective/functional/MPI_Scan	Calling MPI_Scan() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various pre-defined reduction operations

Table 12: C Collective functional tests

Test path	Description
c/collective/functional/MPI_Scan_loc	Calling MPI_Scan() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).
c/collective/functional/MPI_Scan_user	Calling MPI_Scan() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations
c/collective/functional/MPI_Scatter	Calling MPI_Scatter() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
c/collective/functional/MPI_Scatterv	Calling MPI_Scatterv() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
c/collective/functional/ MPI_collective_message	Verifying that collective operation does not interfere with ongoing message traffic.
c/collective/functional/ MPI_collective_overlap	Testing overlapping collective operation in overlapping communicators.

Table 13: C Collective error tests

Test path	Description
c/collective/error/MPI_Allgather_err1	Mis-matched types
c/collective/error/MPI_Allgather_err2	Invalid type (MPI_DATATYPE_NULL)

Table 13: C Collective error tests

Test path	Description
c/collective/error/MPI_Allgather_err3	Invalid size (-1)
c/collective/error/MPI_Allgather_err4	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Allgather_err5	Freed communicator
c/collective/error/MPI_Allgather_err6	Inter-communicator
c/collective/error/MPI_Allgatherv_err1	Mis-matched types
c/collective/error/MPI_Allgatherv_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Allgatherv_err3	Invalid size (-1)
c/collective/error/MPI_Allgatherv_err4	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Allgatherv_err5	Freed communicator
c/collective/error/MPI_Allgatherv_err6	Inter-communicator
c/collective/error/MPI_Allreduce_err1	Invalid type/operator pair
c/collective/error/MPI_Allreduce_err2	Mis-matched types
c/collective/error/MPI_Allreduce_err3	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Allreduce_err4	Invalid size (-1)
c/collective/error/MPI_Allreduce_err5	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Allreduce_err6	Freed communicator
c/collective/error/MPI_Allreduce_err7	Inter-communicator
c/collective/error/MPI_Alltoall_err1	Mis-matched types
c/collective/error/MPI_Alltoall_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Alltoall_err3	Invalid size (-1)
c/collective/error/MPI_Alltoall_err4	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Alltoall_err5	Freed communicator
c/collective/error/MPI_Alltoall_err6	Inter-communicator

Table 13: C Collective error tests

Test path	Description
c/collective/error/MPI_Alltoallv_err1	Mis-matched types
c/collective/error/MPI_Alltoallv_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Alltoallv_err3	Invalid size (-1)
c/collective/error/MPI_Alltoallv_err4	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Alltoallv_err5	Freed communicator
c/collective/error/MPI_Alltoallv_err6	Inter-communicator
c/collective/error/MPI_Barrier_err1	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Barrier_err2	Freed communicator
c/collective/error/MPI_Barrier_err3	Inter-communicator
c/collective/error/MPI_Bcast_err1	Mis-matched types
c/collective/error/MPI_Bcast_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Bcast_err3	Invalid size (-1)
c/collective/error/MPI_Bcast_err4	Invalid root (-1)
c/collective/error/MPI_Bcast_err5	Invalid root (> number of ranks)
c/collective/error/MPI_Bcast_err6	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Bcast_err7	Freed communicator
c/collective/error/MPI_Bcast_err8	Inter-communicator
c/collective/error/MPI_Gather_err1	Mis-matched types
c/collective/error/MPI_Gather_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Gather_err3	Invalid size (-1)
c/collective/error/MPI_Gather_err4	Invalid root (-1)
c/collective/error/MPI_Gather_err5	Invalid root (> number of ranks)
c/collective/error/MPI_Gather_err6	Invalid communicator (MPI_COMM_NULL)

Table 13: C Collective error tests

Test path	Description
c/collective/error/MPI_Gather_err7	Freed communicator
c/collective/error/MPI_Gather_err8	Inter-communicator
c/collective/error/MPI_Gatherv_err1	Mis-matched types
c/collective/error/MPI_Gatherv_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Gatherv_err3	Invalid size (-1)
c/collective/error/MPI_Gatherv_err4	Invalid root (-1)
c/collective/error/MPI_Gatherv_err5	Invalid root (> number of ranks)
c/collective/error/MPI_Gatherv_err6	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Gatherv_err7	Freed communicator
c/collective/error/MPI_Gatherv_err8	Inter-communicator
c/collective/error/MPI_Op_free_err1	Invalid operator (MPI_OP_NULL)
c/collective/error/MPI_Op_free_err2	Pre-defined operators
c/collective/error/MPI_Reduce_err1	Invalid type/operator pair
c/collective/error/MPI_Reduce_err2	Mis-matched types
c/collective/error/MPI_Reduce_err3	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Reduce_err4	Invalid size (-1)
c/collective/error/MPI_Reduce_err5	Invalid root (-1)
c/collective/error/MPI_Reduce_err6	Invalid root (> number of ranks)
c/collective/error/MPI_Reduce_err7	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Reduce_err8	Freed communicator
c/collective/error/MPI_Reduce_err9	Inter-communicator
c/collective/error/MPI_Reduce_scatter_err1	Invalid type/operator pair
c/collective/error/MPI_Reduce_scatter_err2	Mis-matched types

Table 13: C Collective error tests

Test path	Description
c/collective/error/ MPI_Reduce_scatter_err3	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/ MPI_Reduce_scatter_err4	Invalid size (-1)
c/collective/error/ MPI_Reduce_scatter_err5	Invalid communicator (MPI_COMM_NULL)
c/collective/error/ MPI_Reduce_scatter_err6	Freed communicator
c/collective/error/ MPI_Reduce_scatter_err7	Inter-communicator
c/collective/error/MPI_Scan_err1	Invalid type/operator pair
c/collective/error/MPI_Scan_err2	Mis-matched types
c/collective/error/MPI_Scan_err3	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Scan_err4	Invalid size (-1)
c/collective/error/MPI_Scan_err5	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Scan_err6	Freed communicator
c/collective/error/MPI_Scan_err7	Inter-communicator
c/collective/error/MPI_Scatter_err1	Mis-matched types
c/collective/error/MPI_Scatter_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Scatter_err3	Invalid size (-1)
c/collective/error/MPI_Scatter_err4	Invalid root (-1)
c/collective/error/MPI_Scatter_err5	Invalid root (> number of ranks)
c/collective/error/MPI_Scatter_err6	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Scatter_err7	Freed communicator
c/collective/error/MPI_Scatter_err8	Inter-communicator
c/collective/error/MPI_Scatterv_err1	Mis-matched types

Table 13: C Collective error tests

Test path	Description
c/collective/error/MPI_Scatterv_err2	Invalid type (MPI_DATATYPE_NULL)
c/collective/error/MPI_Scatterv_err3	Invalid size (-1)
c/collective/error/MPI_Scatterv_err4	Invalid root (-1)
c/collective/error/MPI_Scatterv_err5	Invalid root (> number of ranks)
c/collective/error/MPI_Scatterv_err6	Invalid communicator (MPI_COMM_NULL)
c/collective/error/MPI_Scatterv_err7	Freed communicator
c/collective/error/MPI_Scatterv_err8	Inter-communicator

B.1.3 Derived Datatype Operations

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/MPI_Address	Calling MPI_Address() looping over various length buffer address
c/datatype/functional/ MPI_Get_elements_basic_type	Calling MPI_Get_elements_basic_type() looping over various communicators, message pre-defined data types, message data lengths and root ranks
c/datatype/functional/ MPI_Pack_basic_type	Calling MPI_Pack() and MPI_Unpack() using basic types with 0 displacement space between any 2 adjacent types. The test loops over various communicators, message lengths and root ranks.

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/MPI_Pack_displs	Calling MPI_Pack() and MPI_Unpack() using basic types with non-zero displacement space between any 2 adjacent types. The test loops over various communicators, message lengths and root ranks.
c/datatype/functional/ MPI_Pack_size_basic	Calling MPI_Pack_size() with basic types. The test loops over various communicators, lengths and datatypes.
c/datatype/functional/ MPI_Pack_size_types	Calling MPI_Pack_size() with user defined types. The test loops through various communicators and lengths.
c/datatype/functional/ MPI_Pack_user_type	Calling MPI_Pack() and MPI_Unpack() with user defined types. The test loops through various communicators, and lengths.
c/datatype/functional/ MPI_Type_contiguous_basic	Calling MPI_Type_contiguous() with basic types. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_contiguous_idispls	Calling MPI_Type_contiguous() with user defined types which has non-zero inner displacement between each basic type. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_contiguous_types	Calling MPI_Type_contiguous() with user defined types, Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_extent_MPI_LB_UB	Calling MPI_Type_extent() with MPI_LB and MPI_UB. test loops through various communicators.

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/ MPI_Type_extent_types	Calling MPI_Type_extent() with user defined types. Test loops through various communicators, and lengths.
c/datatype/functional/ MPI_Type_free_pending_msg	Calling MPI_Type_free() with user defined type that is in use by some active data transmission. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_free_types	Calling MPI_Type_free() with user defined type. Test loops through various communicators, lengths, and root's ranks.
c/datatype/functional/ MPI_Type_hindexed_basic	Calling MPI_Type_hindexed() with basic types. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_hindexed_blklen	Calling MPI_Type_hindexed() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_hindexed_displs	Calling MPI_Type_hindexed() with user defined types using non-zero displacement greater than the type size of each entry of the user defined type. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_hindexed_types	Calling MPI_Type_hindexed() with user defined type. test loops through various communicators, lengths, and root's ranks.
c/datatype/functional/ MPI_Type_hvector_basic	Calling MPI_Type_hvector() with basic types. Test loops through various communicators, lengths, and sender's ranks.

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/ MPI_Type_hvector_blklen	Calling MPI_Type_hvector() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_hvector_stride	Calling MPI_Type_hvector() with user defined type with stride larger than the type size of the user defined type. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_hvector_types	Calling MPI_Type_hvector() with user defined type. test loops through various communicators, lengths, and root's ranks.
c/datatype/functional/ MPI_Type_indexed_basic	Calling MPI_Type_indexed() with basic types. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_indexed_blklen	Calling MPI_Type_indexed() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_indexed_displs	Calling MPI_Type_indexed() with user defined types using nonzero displacement greater than the type size of each entry of the user defined type. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_indexed_types	Calling MPI_Type_indexed() with user defined type. Test loops through various communicators, lengths, and root's ranks.

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/ MPI_Type_lb_2MPI_LB	Calling MPI_Type_lb() with 2 user defined types which each contains MPI_LB. Test loops through various communicators.
c/datatype/functional/ MPI_Type_lb_MPI_LB	Calling MPI_Type_lb() with user defined type which contains MPI_LB. Test loops through various communicators.
c/datatype/functional/ MPI_Type_lb_neg_displ	Calling MPI_Type_lb() with user defined type which has negative displacement. Test loops through various communicators.
c/datatype/functional/ MPI_Type_lb_pos_displ	Calling MPI_Type_lb() with user defined type which has positive displacement. Test loops through various communicators and data types.
c/datatype/functional/ MPI_Type_size_MPI_LB_UB	Calling MPI_Type_size() with MPI_LB and MPI_UB. Test loops through various communicators and data types.
c/datatype/functional/ MPI_Type_size_basic	Calling MPI_Type_size() with basic types. Test loops through various communicators,
c/datatype/functional/ MPI_Type_size_types	Calling MPI_Type_size() with user defined type. Test loops through various communicators, and lengths.
c/datatype/functional/ MPI_Type_struct_basic	Calling MPI_Type_struct() with basic types. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_struct_blklen	Calling MPI_Type_struct() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/ MPI_Type_struct_displs	Calling MPI_Type_struct() with user defined types using nonzero displacement greater than the type size of each entry of the user defined type. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_struct_types	Calling MPI_Type_struct() with basic and user defined type. Test loops through various communicators, lengths, and root's ranks.
c/datatype/functional/ MPI_Type_ub_2MPI_UB	Calling MPI_Type_ub() with 2 user defined types which each contains MPI_UB. Test loops through various communicators.
c/datatype/functional/ MPI_Type_ub_MPI_UB	Calling MPI_Type_ub() with user defined type which contains MPI_UB. Test loops through various communicators.
c/datatype/functional/ MPI_Type_ub_neg_displ	Calling MPI_Type_ub() with user defined type which has negative displacement. Test loops through various communicators.
c/datatype/functional/ MPI_Type_ub_pos_displ	Calling MPI_Type_ub() with user defined type which has positive displacement. Test loops through various communicators.
c/datatype/functional/ MPI_Type_vector_basic	Calling MPI_Type_vector() with basic types. Test loops through various communicators, lengths, and sender's ranks.
c/datatype/functional/ MPI_Type_vector_blklen	Calling MPI_Type_vector() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.

Table 14: Derived datatype functional tests

Test path	Description
c/datatype/functional/ MPI_Type_vector_stride	Calling MPI_Type_vector() with user defined type with stride larger than the type size of the user defined type. Test loops through various communicators, lengths and sender's rank.
c/datatype/functional/ MPI_Type_vector_types	Calling MPI_Type_vector() with basic and user defined type. Test loops through various communicators, lengths, and sender's ranks.

Table 15: C Derived datatype error tests

Test path	Description
c/datatype/error/MPI_Get_elements_err1	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Get_elements_err2	Uncommitted datatype
c/datatype/error/MPI_Pack_err1	Negative input count
c/datatype/error/MPI_Pack_err2	Negative outsize
c/datatype/error/MPI_Pack_err3	Invalid communicator (MPI_COMM_NULL)
c/datatype/error/MPI_Pack_err4	Invalid datatype (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Pack_err5	Uncommitted datatype
c/datatype/error/MPI_Pack_err6	outsize < incount with basic type
c/datatype/error/MPI_Pack_size_err1	Negative input incount
c/datatype/error/MPI_Pack_size_err2	Invalid communicator (MPI_COMM_NULL)
c/datatype/error/MPI_Pack_size_err3	Invalid datatype (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Pack_size_err4	Uncommitted datatype

Table 15: C Derived datatype error tests

Test path	Description
c/datatype/error/MPI_Type_commit_err1	Invalid datatype (MPI_DATATYPE_NULL)
c/datatype/error/ MPI_Type_contiguous_err1	Negative input count
c/datatype/error/ MPI_Type_contiguous_err2	Invalid datatype (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_extent_err1	Invalid datatype (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_free_err1	Invalid datatype (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_free_err2	Predefined type (MPI_BYTE)
c/datatype/error/MPI_Type_free_err3	Predefined type (MPI_CHAR)
c/datatype/error/MPI_Type_free_err4	Predefined type (MPI_DOUBLE)
c/datatype/error/MPI_Type_free_err5	Predefined type (MPI_FLOAT)
c/datatype/error/MPI_Type_free_err6	Predefined type (MPI_INT)
c/datatype/error/MPI_Type_free_err7	Predefined type (MPI_LONG)
c/datatype/error/MPI_Type_free_err8	Predefined type (MPI_LONG_DOUBLE)
c/datatype/error/MPI_Type_free_err9	Predefined type (MPI_PACKED)
c/datatype/error/MPI_Type_free_err10	Predefined type (MPI_SHORT)
c/datatype/error/MPI_Type_free_err11	Predefined type (MPI_UNSIGNED_CHAR)
c/datatype/error/MPI_Type_free_err12	Predefined type (MPI_UNSIGNED)
c/datatype/error/MPI_Type_free_err13	Predefined type (MPI_UNSIGNED_LONG)
c/datatype/error/MPI_Type_free_err14	Predefined type (MPI_UNSIGNED_SHORT)
c/datatype/error/MPI_Type_free_err15	Predefined type (MPI_LB)
c/datatype/error/MPI_Type_free_err16	Predefined type (MPI_UB)

Table 15: C Derived datatype error tests

Test path	Description
c/datatype/error/ MPI_Type_hindexed_err1	Negative input count
c/datatype/error/ MPI_Type_hindexed_err2	Negative blocklength
c/datatype/error/ MPI_Type_hindexed_err3	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_hvector_err1	Negative input count
c/datatype/error/MPI_Type_hvector_err2	Negative blocklength
c/datatype/error/MPI_Type_hvector_err3	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_indexed_err1	Negative input count
c/datatype/error/MPI_Type_indexed_err2	Negative blocklength
c/datatype/error/MPI_Type_indexed_err3	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_lb_err1	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_size_err1	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_struct_err1	Negative input count
c/datatype/error/MPI_Type_struct_err2	Negative blocklength
c/datatype/error/MPI_Type_struct_err3	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_ub_err1	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Type_vector_err1	Negative input count
c/datatype/error/MPI_Type_vector_err2	Negative blocklength
c/datatype/error/MPI_Type_vector_err3	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Unpack_err1	Negative insize
c/datatype/error/MPI_Unpack_err2	Negative outcount

Table 15: C Derived datatype error tests

Test path	Description
c/datatype/error/MPI_Unpack_err3	Invalid communicator (MPI_COMM_NULL)
c/datatype/error/MPI_Unpack_err4	Invalid type (MPI_DATATYPE_NULL)
c/datatype/error/MPI_Unpack_err5	Uncommitted datatype
c/datatype/error/MPI_Unpack_err6	outcount < insize

B.1.4 Environment Manager

Table 16: Environment Manager functional tests

Test path	Description
c/env_manager/functional/MPI_Abort	Calling MPI_Abort() from rank 0 in MPI_COMM_WORLD.
c/env_manager/functional/MPI_Errhandler_fatal	Setting MPI_ERRORS_ARE_FATAL using MPI_Errhandler_set().
c/env_manager/functional/MPI_Errhandler_free	Freeing user defined error handler using MPI_Errhandler_free().
c/env_manager/functional/MPI_Errhandler_get	Setting user defined error handler and retrieving it using MPI_Errhandler_get().
c/env_manager/functional/MPI_Errhandler_set	Setting user defined error handler using MPI_Errhandler_set().
c/env_manager/functional/MPI_Error_string	Calling MPI_Error_string for every standard error class.
c/env_manager/functional/MPI_Finalize	Calling MPI_Finalize() from all ranks.
c/env_manager/functional/MPI_Get_processor_name	Calling MPI_Get_processor_name() from each rank.

Table 16: Environment Manager functional tests

Test path	Description
c/env_manager/functional/MPI_Init_attr	Calling MPI_Attr_get() to get all the predefined attributes (MPI_TAG_UB, MPI_HOST, MPI_WTIME_IS_GLOBAL, MPI_IO).
c/env_manager/functional/ MPI_Initialized	Calling MPI_Initialized() before and after MPI_Init().
c/env_manager/functional/MPI_Pcontrol	Calling MPI_Pcontrol() with various levels.
c/env_manager/functional/MPI_Wtime	Calling MPI_Wtime() and MPI_Wtick() from all ranks in MPI_COMM_WORLD.

Table 17: Environment Manager functional tests

Test path	Description
c/env_manager/error/MPI_Abort_err1	Invalid communicator (MPI_COMM_NULL)
c/env_manager/error/ MPI_Errhandler_free_err1	Invalid error handler (MPI_ERRHANDLER_NULL)
c/env_manager/error/ MPI_Errhandler_get_err1	Invalid communicator (MPI_COMM_NULL)
c/env_manager/error/ MPI_Errhandler_get_err2	Freed communicator
c/env_manager/error/ MPI_Errhandler_set_err1	Invalid communicator (MPI_COMM_NULL)
c/env_manager/error/ MPI_Errhandler_set_err2	Freed communicator
c/env_manager/error/ MPI_Errhandler_set_err3	Invalid error handler (MPI_ERRHANDLER_NULL)
c/env_manager/error/MPI_Init_err1	Calling MPI_Init() twice

B.1.5 Group, Context, and Communicator Operations

Table 18: Group, Context, and Communicator functional tests

Test path	Description
c/grp_ctxt_comm/functional/ MPI_Comm_compare	Using MPI_Comm_compare() with MPI_IDENT on various user created communicators and pre-defined communicators (MPI_COMM_WORLD, MPI_COMM_SELF).
c/grp_ctxt_comm/functional/ MPI_Comm_create	Calling MPI_Comm_create() to create communicators from arbitrary group of ranks.
c/grp_ctxt_comm/functional/ MPI_Comm_dup	Calling MPI_Comm_dup() to duplicate various communicators making sure they really are separate communicators.
c/grp_ctxt_comm/functional/ MPI_Comm_group	Calling MPI_Comm_group() to retrieve groups from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Comm_split1	Calling MPI_Comm_split() to split various intracommunicators into communicators of size one.
c/grp_ctxt_comm/functional/ MPI_Comm_split2	Calling MPI_Comm_split() to split various intracommunicators into communicators the same as the original one.
c/grp_ctxt_comm/functional/ MPI_Comm_split3	Calling MPI_Comm_split() to split various intracommunicators into communicators of half of the original size.
c/grp_ctxt_comm/functional/ MPI_Comm_split4	Calling MPI_Comm_split() to split various intracommunicators into 2 communicators with some non-zero set of ranks not being in either.
c/grp_ctxt_comm/functional/ MPI_Group_compare	Calling MPI_Group_compare() with various groups.

Table 18: Group, Context, and Communicator functional tests

Test path	Description
c/grp_ctxt_comm/functional/ MPI_Group_difference1	Calling MPI_Group_difference() with arbitrary groups and (MPI_EMPTY_GROUP, or MPI_COMM_WORLD).
c/grp_ctxt_comm/functional/ MPI_Group_difference2	Calling MPI_Group_difference() with arbitrary non-overlapping groups of MPI_COMM_WORLD.
c/grp_ctxt_comm/functional/ MPI_Group_difference3	Calling MPI_Group_difference() with arbitrary overlapping groups of MPI_COMM_WORLD.
c/grp_ctxt_comm/functional/ MPI_Group_excl1	Calling MPI_Group_excl() to create groups with various ranks from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_excl2	Calling MPI_Group_excl() to create groups of size one from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_incl1	Calling MPI_Group_incl() to create groups with various ranks from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_incl2	Calling MPI_Group_incl() to create groups of size one from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_intersection1	Calling MPI_Group_intersection() with arbitrary groups and (MPI_EMPTY_GROUP, or MPI_COMM_WORLD).
c/grp_ctxt_comm/functional/ MPI_Group_intersection2	Calling MPI_Group_intersection() with arbitrary non-overlapping groups of MPI_COMM_WORLD.
c/grp_ctxt_comm/functional/ MPI_Group_intersection3	Calling MPI_Group_intersection() with arbitrary overlapping groups of MPI_COMM_WORLD.

Table 18: Group, Context, and Communicator functional tests

Test path	Description
c/grp_ctxt_comm/functional/ MPI_Group_range_excl1	Calling MPI_Group_range_excl() to create groups with various ranks from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_range_excl2	Calling MPI_Group_range_excl() to create groups of size one from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_range_excl3	Calling MPI_Group_range_excl() to create groups excluding every other rank from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_range_incl1	Calling MPI_Group_range_incl() to create groups with various ranks from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_range_incl2	Calling MPI_Group_range_incl() to create groups of size one from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_range_incl3	Calling MPI_Group_range_incl() to create groups excluding every other rank from arbitrary communicators.
c/grp_ctxt_comm/functional/ MPI_Group_trans_ranks	Calling MPI_Group_compare() with groups from various communicators.
c/grp_ctxt_comm/functional/ MPI_Group_union1	Calling MPI_Group_union() with arbitrary groups and (MPI_EMPTY_GROUP, or MPI_COMM_WORLD).
c/grp_ctxt_comm/functional/ MPI_Group_union2	Calling MPI_Group_union() with various groups.
c/grp_ctxt_comm/functional/ MPI_Group_union3	Calling MPI_Group_intersection() with arbitrary overlapping groups of MPI_COMM_WORLD.

Table 18: Group, Context, and Communicator functional tests

Test path	Description
c/grp_ctxt_comm/functional/ MPI_Intercomm_create1	Calling MPI_Comm_split() to split various communicators (either half or 1/3) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators.
c/grp_ctxt_comm/functional/ MPI_Intercomm_create2	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators.
c/grp_ctxt_comm/functional/ MPI_Intercomm_merge1	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators. Then call MPI_Intercomm_merge()
c/grp_ctxt_comm/functional/ MPI_Intercomm_merge2	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators. Then call MPI_Intercomm_merge()
c/grp_ctxt_comm/functional/ MPI_Intercomm_merge3	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators. Then call MPI_Intercomm_merge()
c/grp_ctxt_comm/functional/ MPI_Keyval1	Creating MPI Keyvals and assigning attributes in various communicators.
c/grp_ctxt_comm/functional/ MPI_Keyval2	Verifying attribute values can be changed.

Table 18: Group, Context, and Communicator functional tests

Test path	Description
c/grp_ctxt_comm/functional/ MPI_Keyval3	Verifying errors from copy and delete functions are returned to an application.

Table 19: Group, Context, and Communicator error tests

Test path	Description
c/grp_ctxt_comm/error/ MPI_Attr_delete_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Attr_delete_err2	Invalid keyval (MPI_KEYVAL_INVALID)
c/grp_ctxt_comm/error/ MPI_Attr_get_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Attr_get_err2	Invalid keyval (MPI_KEYVAL_INVALID)
c/grp_ctxt_comm/error/ MPI_Attr_put_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Attr_put_err2	Invalid keyval (MPI_KEYVAL_INVALID)
c/grp_ctxt_comm/error/ MPI_Comm_compare_err1	Invalid communicator (comm1=MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_compare_err2	Invalid communicator (comm2=MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_create_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_create_err2	Invalid Group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_dup_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_free_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_free_err2	Invalid communicator (MPI_COMM_WORLD)

Table 19: Group, Context, and Communicator error tests

Test path	Description
c/grp_ctxt_comm/error/ MPI_Comm_free_err3	Invalid communicator (MPI_COMM_SELF)
c/grp_ctxt_comm/error/ MPI_Comm_group_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_rank_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_remote_group_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_remote_group_err2	Invalid communicator (MPI_COMM_WORLD)
c/grp_ctxt_comm/error/ MPI_Comm_remote_size_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_remote_size_err2	Invalid communicator (MPI_COMM_WORLD)
c/grp_ctxt_comm/error/ MPI_Comm_size_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_split_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Comm_test_inter_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Group_compare_err1	Invalid group (group1=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_compare_err2	Invalid group (group2=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_difference_err1	Invalid group (group1=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_difference_err2	Invalid group (group2=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_excl_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_excl_err2	Negative n

Table 19: Group, Context, and Communicator error tests

Test path	Description
c/grp_ctxt_comm/error/ MPI_Group_excl_err3	n > size of group
c/grp_ctxt_comm/error/ MPI_Group_excl_err4	Negative rank
c/grp_ctxt_comm/error/ MPI_Group_excl_err5	rank >= size of group
c/grp_ctxt_comm/error/ MPI_Group_excl_err6	Duplicated ranks
c/grp_ctxt_comm/error/ MPI_Group_free_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_incl_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_incl_err2	Negative n
c/grp_ctxt_comm/error/ MPI_Group_incl_err3	n > size of group
c/grp_ctxt_comm/error/ MPI_Group_incl_err4	Negative rank
c/grp_ctxt_comm/error/ MPI_Group_incl_err5	rank >= size of group
c/grp_ctxt_comm/error/ MPI_Group_incl_err6	Duplicated ranks
c/grp_ctxt_comm/error/ MPI_Group_intersection_err1	Invalid group (group1=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_intersection_err2	Invalid group (group2=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_range_excl_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_range_excl_err2	Negative n
c/grp_ctxt_comm/error/ MPI_Group_range_excl_err3	ranges[] implies negative ranks

Table 19: Group, Context, and Communicator error tests

Test path	Description
c/grp_ctxt_comm/error/MPI_Group_range_excl_err4	ranges[] implies rank >= size of group
c/grp_ctxt_comm/error/MPI_Group_range_excl_err5	Negative stride and first < last
c/grp_ctxt_comm/error/MPI_Group_range_excl_err6	Positive stride and first > last
c/grp_ctxt_comm/error/MPI_Group_range_excl_err7	stride = 0
c/grp_ctxt_comm/error/MPI_Group_range_incl_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/MPI_Group_range_incl_err2	Negative n
c/grp_ctxt_comm/error/MPI_Group_range_incl_err3	ranges[] implies negative ranks
c/grp_ctxt_comm/error/MPI_Group_range_incl_err4	ranges[] implies rank >= size of group
c/grp_ctxt_comm/error/MPI_Group_range_incl_err5	Negative stride and first < last
c/grp_ctxt_comm/error/MPI_Group_range_incl_err6	Positive stride and first > last
c/grp_ctxt_comm/error/MPI_Group_range_incl_err7	stride = 0
c/grp_ctxt_comm/error/MPI_Group_rank_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/MPI_Group_size_err1	Invalid group (MPI_GROUP_NULL)
c/grp_ctxt_comm/error/MPI_Group_trans_ranks_err1	Invalid group (group1=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/MPI_Group_trans_ranks_err2	Invalid group (group2=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/MPI_Group_trans_ranks_err3	Negative n

Table 19: Group, Context, and Communicator error tests

Test path	Description
c/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err4	Negative argument in ranks1
c/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err5	rank >= size of group in ranks1
c/grp_ctxt_comm/error/ MPI_Group_union_err1	Invalid group (group1=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Group_union_err2	Invalid group (group2=MPI_GROUP_NULL)
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err1	Invalid communicator (local_comm= MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err2	Invalid communicator (peer_comm= MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err3	local_comm is intercommunicator
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err5	Local and remote leaders not in peer_comm
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err7	local_leader = MPI_ANY_SOURCE
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err8	remote_leader = MPI_ANY_SOURCE
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err9	local_leader not a valid rank in local_comm
c/grp_ctxt_comm/error/ MPI_Intercomm_create_err10	remote_leader not a valid rank in peer_comm
c/grp_ctxt_comm/error/ MPI_Intercomm_merge_err1	Invalid communicator (MPI_COMM_NULL)
c/grp_ctxt_comm/error/ MPI_Intercomm_merge_err2	Invalid communicator (intercommunicator)
c/grp_ctxt_comm/error/ MPI_Intercomm_merge_err3	Different values of high within a single group
c/grp_ctxt_comm/error/ MPI_Keyval_free_err1	Invalid keyval (MPI_KEYVAL_INVALID)

B.1.6 Miscllany Operations

Table 20: Miscllany functional tests

Test path	Description
c/misc/functional/MPI_defs	Test to verify all required MPI constants and routines are defined.
c/misc/functional/MPI_pdefs	Test to verify all required MPI constants and routines are defined correctly for profiling.

B.1.7 Nonblocking Operations

Table 21: Nonblocking functional tests

Test path	Description
c/nonblocking/functional/ MPI_Ibsend_ator	Calling MPI_Ibsend() / MPI_Irecv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/nonblocking/functional/ MPI_Ibsend_null	Calling MPI_Ibsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/nonblocking/functional/ MPI_Ibsend_overtake	Calling MPI_Ibsend() and MPI_Irecv() to send a sequence of messages and make sure they are received in the order they were sent.

Table 21: Nonblocking functional tests

Test path	Description
c/nonblocking/functional/ MPI_Ibsend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/nonblocking/functional/ MPI_Ibsend_rtoa	Calling MPI_Ibsend() / MPI_Irecv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/nonblocking/functional/ MPI_Ibsend_rtoaq	Calling MPI_Ibsend() / MPI_Irecv() to send message from root to all other ranks in communicators. test will retry if the buffer is full. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/nonblocking/functional/ MPI_Irecv_comm	Calling MPI_Irecv() with various communicators created and make sure the messages are received in the order sent for each communicator (not the order the recvs were posted).
c/nonblocking/functional/MPI_Irecv_null	Calling MPI_Irecv() with source of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/nonblocking/functional/ MPI_Irecv_pack	Calling MPI_Irecv() verifying that every datatype can be sent as MPI_PACKED. Test loops through various communicators, message types, and message lengths.

Table 21: Nonblocking functional tests

Test path	Description
c/nonblocking/functional/ MPI_Irsend_null	Calling MPI_Irsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/nonblocking/functional/ MPI_Irsend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/nonblocking/functional/ MPI_Irsend_rtoa	Calling MPI_Irsend() / MPI_Irecv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.
c/nonblocking/functional/MPI_Isend_ator	Calling MPI_Isend() / MPI_Irecv() to send message from all other ranks to the root’s rank in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.
c/nonblocking/functional/ MPI_Isend_ator2	Calling MPI_Isend() / MPI_Irecv() to send message from all other ranks to the root’s rank in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.
c/nonblocking/functional/ MPI_Isend_fairness	Quality of implementation test testing the fairness of the sending protocol dealing with multiple MPI_Isend() from various ranks to the same destination.

Table 21: Nonblocking functional tests

Test path	Description
c/nonblocking/functional/MPI_Isend_flood	Quality of implementation test testing the behavior of the sending protocol dealing with many MPI_Isend() from various ranks to the same destination.
c/nonblocking/functional/MPI_Isend_null	Calling MPI_Isend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/nonblocking/functional/MPI_Isend_off	Calling MPI_Isend() / MPI_Irecv() to send message from all ranks in communicators to a selected root with byte offsets. Test loops through various communicators, message datatypes, message lengths and root’ ranks.
c/nonblocking/functional/MPI_Isend_overtake	Calling MPI_Isend() / MPI_Irecv() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
c/nonblocking/functional/MPI_Isend_overtake2	Calling MPI_Isend() / MPI_Irecv() using different tags to send a sequence of messages and make sure they are received in the order they were sent.
c/nonblocking/functional/MPI_Isend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/nonblocking/functional/MPI_Isend_rtoa	Calling MPI_Isend() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.

Table 21: Nonblocking functional tests

Test path	Description
c/nonblocking/functional/MPI_Isend_self	Calling MPI_Isend() to send message from a rank to itself. Test loops through various communicators.
c/nonblocking/functional/MPI_Issend_ator	Calling MPI_Issend() / MPI_Irecv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/nonblocking/functional/MPI_Issend_null	Calling MPI_Issend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/nonblocking/functional/MPI_Issend_overtake1	Calling MPI_Issend() / MPI_Irecv() and making sure it does not return until a matching receive has been posted.
c/nonblocking/functional/MPI_Issend_overtake2	Calling MPI_Issend() / MPI_Irecv() and make sure it makes progress when matched by a MPI_Irecv() even though MPI_Wait() is not called until sometime later.
c/nonblocking/functional/MPI_Issend_overtake3	2 ranks calling MPI_Issend() / MPI_Irecv() to each other. They should both make progress when matching MPI_Recv()s have been called.
c/nonblocking/functional/MPI_Issend_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.

Table 21: Nonblocking functional tests

Test path	Description
c/nonblocking/functional/MPI_Issend_rtoa	Calling MPI_Issend() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/nonblocking/functional/MPI_Request_free	Calling MPI_Request_free() on each non-blocking calls. All requests are expected to complete even though they are freed.
c/nonblocking/functional/MPI_Testall	Calling MPI_Testall() on active requests.
c/nonblocking/functional/MPI_Testany	Calling MPI_Testany() on active requests.
c/nonblocking/functional/MPI_Testsome	Calling MPI_Testsome() on active requests.
c/nonblocking/functional/MPI_Waitall	Calling MPI_Waitall() on active requests.
c/nonblocking/functional/MPI_Waitany	Calling MPI_Waitany() on active requests.
c/nonblocking/functional/MPI_Waitsome	Calling MPI_Waitsome() on active requests.
c/nonblocking/functional/async	Stress asynchronous rank-to-rank communication.
c/nonblocking/functional/rings	Stress message sending around rings of ranks.

Table 22: Nonblocking error tests

Test path	Description
c/nonblocking/error/MPI_Ibsend_err1	Negative destination rank
c/nonblocking/error/MPI_Ibsend_err2	Destination rank >= group size
c/nonblocking/error/MPI_Ibsend_err3	Invalid communicator (MPI_COMM_NULL)

Table 22: Nonblocking error tests

Test path	Description
c/nonblocking/error/MPI_Ibsend_err4	Freed communicator
c/nonblocking/error/MPI_Ibsend_err5	tag < 0
c/nonblocking/error/MPI_Ibsend_err6	tag > MPI_TAG_UB
c/nonblocking/error/MPI_Ibsend_err7	Negative count
c/nonblocking/error/MPI_Ibsend_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/nonblocking/error/MPI_Ibsend_err9	Mismatching type at sender and receiver
c/nonblocking/error/MPI_Ibsend_err10	Message size > attached buffer size
c/nonblocking/error/MPI_Ibsend_err11	No buffer attached
c/nonblocking/error/MPI_Irecv_err1	Negative destination rank
c/nonblocking/error/MPI_Irecv_err2	Destination rank >= group size
c/nonblocking/error/MPI_Irecv_err3	Invalid communicator (MPI_COMM_NULL)
c/nonblocking/error/MPI_Irecv_err4	Freed communicator
c/nonblocking/error/MPI_Irecv_err5	tag < 0
c/nonblocking/error/MPI_Irecv_err6	tag > MPI_TAG_UB
c/nonblocking/error/MPI_Irecv_err7	Negative count
c/nonblocking/error/MPI_Irecv_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/nonblocking/error/MPI_Irecv_err9	Receiving message > Irecv count
c/nonblocking/error/MPI_Irsend_err1	Negative destination rank
c/nonblocking/error/MPI_Irsend_err2	Destination rank >= group size
c/nonblocking/error/MPI_Irsend_err3	Invalid communicator (MPI_COMM_NULL)
c/nonblocking/error/MPI_Irsend_err4	Freed communicator
c/nonblocking/error/MPI_Irsend_err5	tag < 0
c/nonblocking/error/MPI_Irsend_err6	tag > MPI_TAG_UB

Table 22: Nonblocking error tests

Test path	Description
c/nonblocking/error/MPI_Irsend_err7	Negative count
c/nonblocking/error/MPI_Irsend_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/nonblocking/error/MPI_Irsend_err9	Mismatching type at sender and receiver
c/nonblocking/error/MPI_Isend_err1	Negative destination rank
c/nonblocking/error/MPI_Isend_err2	Destination rank >= group size
c/nonblocking/error/MPI_Isend_err3	Invalid communicator (MPI_COMM_NULL)
c/nonblocking/error/MPI_Isend_err4	Freed communicator
c/nonblocking/error/MPI_Isend_err5	tag < 0
c/nonblocking/error/MPI_Isend_err6	tag > MPI_TAG_UB
c/nonblocking/error/MPI_Isend_err7	Negative count
c/nonblocking/error/MPI_Isend_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/nonblocking/error/MPI_Isend_err9	Mismatching type at sender and receiver
c/nonblocking/error/MPI_Issend_err1	Negative destination rank
c/nonblocking/error/MPI_Issend_err2	Destination rank >= group size
c/nonblocking/error/MPI_Issend_err3	Invalid communicator (MPI_COMM_NULL)
c/nonblocking/error/MPI_Issend_err4	Freed communicator
c/nonblocking/error/MPI_Issend_err5	tag < 0
c/nonblocking/error/MPI_Issend_err6	tag > MPI_TAG_UB
c/nonblocking/error/MPI_Issend_err7	Negative count
c/nonblocking/error/MPI_Issend_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/nonblocking/error/MPI_Issend_err9	Mismatching type at sender and receiver

B.1.8 Persistent Operations

Table 23: Persistent functional tests

Test path	Description
c/persist_request/functional/ MPI_Bsend_init_null	Calling MPI_Bsend_init() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
c/persist_request/functional/ MPI_Bsend_init_overtake	Calling MPI_Bsend_init() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
c/persist_request/functional/ MPI_Bsend_init_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/persist_request/functional/ MPI_Bsend_init_rtoa	Calling MPI_Bsend_init() and MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/persist_request/functional/ MPI_Recv_init_comm	Calling MPI_Recv_init() and MPI_Bsend_init() with various communicators created and make sure the messages are received in the order sent for each communicator (not the order the recv/s were posted).
c/persist_request/functional/ MPI_Recv_init_null	Calling MPI_Recv_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.

Table 23: Persistent functional tests

Test path	Description
c/persist_request/functional/ MPI_Recv_init_pack	Calling MPI_Recv_init() / MPI_Send() verifying that every datatype can be sent as MPI_PACKED. Test loops through various communicators, message types, and message lengths.
c/persist_request/functional/ MPI_Request_free_p	Calling MPI_Request_free() using request generated from all persistent calls making sure pending request can still be completed even though it has been freed.
c/persist_request/functional/ MPI_Rsend_init_null	Calling MPI_Rsend_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
c/persist_request/functional/ MPI_Rsend_init_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/persist_request/functional/ MPI_Rsend_init_rtoa	Calling MPI_Rsend_init() / MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
c/persist_request/functional/ MPI_Send_init_ator	Calling MPI_Send_init() / MPI_Recv_init() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 23: Persistent functional tests

Test path	Description
c/persist_request/functional/ MPI_Send_init_null	Calling MPI_Send_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
c/persist_request/functional/ MPI_Send_init_off	Calling MPI_Send_init() / MPI_Recv_init() to send message from all ranks in communicators to a selected root with byte offsets. Test loops through various communicators, message datatypes, message lengths and root' ranks.
c/persist_request/functional/ MPI_Send_init_overtake	Calling MPI_Send_init() / MPI_Recv_init() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
c/persist_request/functional/ MPI_Send_init_overtake2	Calling MPI_Send_init() / MPI_Recv_init() and MPI_Recv_init() using different tags to send a sequence of messages and make sure they are received in the order they were sent.
c/persist_request/functional/ MPI_Send_init_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/persist_request/functional/ MPI_Send_init_rtoa	Calling MPI_Send_init() / MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 23: Persistent functional tests

Test path	Description
c/persist_request/functional/ MPI_Send_init_self	Calling MPI_Send_init() / MPI_Recv_init() and MPI_Recv_init() to send message from a rank to itself. Test loops through various communicators.
c/persist_request/functional/ MPI_Ssend_init_ator	Calling MPI_Ssend() / MPI_Recv_init() to send message from all other ranks to the root’s rank in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.
c/persist_request/functional/ MPI_Ssend_init_null	Calling MPI_Ssend_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
c/persist_request/functional/ MPI_Ssend_init_overtake	Calling MPI_Ssend_init() / MPI_Recv_init() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
c/persist_request/functional/ MPI_Ssend_init_perf	Test for recording the latency for sending a message of specified length around a ring in a communicator. Test loops through various communicators, and message lengths.
c/persist_request/functional/ MPI_Ssend_init_rtoa	Calling MPI_Ssend_init() / MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, and message lengths.
c/persist_request/functional/MPI_Startall1	Calling MPI_Bsend_init()/ MPI_Startall() in various communicators.

Table 23: Persistent functional tests

Test path	Description
c/persist_request/functional/MPI_Startall2	Calling MPI_Send_init()/ MPI_Startall() in various communicators.
c/persist_request/functional/MPI_Test_p	Calling MPI_Test() on active requests.
c/persist_request/functional/ MPI_Testall_p	Calling MPI_Testall() on active requests.
c/persist_request/functional/ MPI_Testany_p	Calling MPI_Testany() on active requests.
c/persist_request/functional/ MPI_Testsome_p	Calling MPI_Testsome() on active requests.
c/persist_request/functional/ MPI_Waitall_p	Calling MPI_Waitall() on active requests.
c/persist_request/functional/ MPI_Waitany_p	Calling MPI_Waitany() on active requests.
c/persist_request/functional/ MPI_Waitsome_p	Calling MPI_Waitsome() on active requests.

Table 24: Persistent error tests

Test path	Description
c/persist_request/error/ MPI_Bsend_init_err1	Negative destination rank
c/persist_request/error/ MPI_Bsend_init_err2	Destination rank >= group size
c/persist_request/error/ MPI_Bsend_init_err3	Invalid communicator (MPI_COMM_NULL)
c/persist_request/error/ MPI_Bsend_init_err4	Freed communicator
c/persist_request/error/ MPI_Bsend_init_err5	tag < 0
c/persist_request/error/ MPI_Bsend_init_err6	tag > MPI_TAG_UB

Table 24: Persistent error tests

Test path	Description
c/persist_request/error/ MPI_Bsend_init_err7	Negative count
c/persist_request/error/ MPI_Bsend_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/persist_request/error/ MPI_Bsend_init_err9	Mismatching type at sender and receiver
c/persist_request/error/ MPI_Bsend_init_err10	Message size > attached buffer size
c/persist_request/error/ MPI_Bsend_init_err11	No buffer attached
c/persist_request/error/ MPI_Recv_init_err1	Negative destination rank
c/persist_request/error/ MPI_Recv_init_err2	Destination rank >= group size
c/persist_request/error/ MPI_Recv_init_err3	Invalid communicator (MPI_COMM_NULL)
c/persist_request/error/ MPI_Recv_init_err4	Freed communicator
c/persist_request/error/ MPI_Recv_init_err5	tag < 0
c/persist_request/error/ MPI_Recv_init_err6	tag > MPI_TAG_UB
c/persist_request/error/ MPI_Recv_init_err7	Negative count
c/persist_request/error/ MPI_Recv_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/persist_request/error/ MPI_Recv_init_err9	Message size > Recv_init count
c/persist_request/error/ MPI_Request_free_err1	Freed request
c/persist_request/error/ MPI_Rsend_init_err1	Negative destination rank

Table 24: Persistent error tests

Test path	Description
c/persist_request/error/ MPI_Rsend_init_err2	Destination rank >= group size
c/persist_request/error/ MPI_Rsend_init_err3	Invalid communicator (MPI_COMM_NULL)
c/persist_request/error/ MPI_Rsend_init_err4	Freed communicator
c/persist_request/error/ MPI_Rsend_init_err5	tag < 0
c/persist_request/error/ MPI_Rsend_init_err6	tag > MPI_TAG_UB
c/persist_request/error/ MPI_Rsend_init_err7	Negative count
c/persist_request/error/ MPI_Rsend_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/persist_request/error/ MPI_Rsend_init_err9	Mismatching type at sender and receiver
c/persist_request/error/ MPI_Send_init_err1	Negative destination rank
c/persist_request/error/ MPI_Send_init_err2	Destination rank >= group size
c/persist_request/error/ MPI_Send_init_err3	Invalid communicator (MPI_COMM_NULL)
c/persist_request/error/ MPI_Send_init_err4	Freed communicator
c/persist_request/error/ MPI_Send_init_err5	tag < 0
c/persist_request/error/ MPI_Send_init_err6	tag > MPI_TAG_UB
c/persist_request/error/ MPI_Send_init_err7	Negative count
c/persist_request/error/ MPI_Send_init_err8	Invalid datatype (MPI_DATATYPE_NULL)

Table 24: Persistent error tests

Test path	Description
c/persist_request/error/ MPI_Send_init_err9	Mismatching type at sender and receiver
c/persist_request/error/ MPI_Ssend_init_err1	Negative destination rank
c/persist_request/error/ MPI_Ssend_init_err2	Destination rank >= group size
c/persist_request/error/ MPI_Ssend_init_err3	Invalid communicator (MPI_COMM_NULL)
c/persist_request/error/ MPI_Ssend_init_err4	Freed communicator
c/persist_request/error/ MPI_Ssend_init_err5	tag < 0
c/persist_request/error/ MPI_Ssend_init_err6	tag > MPI_TAG_UB
c/persist_request/error/ MPI_Ssend_init_err7	Negative count
c/persist_request/error/ MPI_Ssend_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/persist_request/error/ MPI_Ssend_init_err9	Mismatching type at sender and receiver
c/persist_request/error/MPI_Start_err1	Already active request
c/persist_request/error/MPI_Start_err2	Freed request
c/persist_request/error/MPI_Startall_err1	Already active request
c/persist_request/error/MPI_Startall_err2	Freed request

B.1.9 Probe and Cancel

Table 25: Probe and Cancel functional tests

Test path	Description
c/probe_cancel/functional/ MPI_Cancel_irecv	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active irecv request(s). Another irecv request will be posted afterward with a matching send request. Looping through various communicators, message lengths, and root’s ranks.
c/probe_cancel/functional/ MPI_Cancel_isend	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active isend request(s). Another isend request will be posted afterward with a matching recv request. Looping through various communicators, message lengths, and root’s ranks.
c/probe_cancel/functional/ MPI_Cancel_issend	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active issend request(s). Another issend request will be posted afterward with a matching recv request. Looping through various communicators, message lengths, and root’s ranks.
c/probe_cancel/functional/ MPI_Cancel_persist_recv	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active persistent receive request(s). Another persistent receive request will be posted afterward with a matching send request. Looping through various communicators, message lengths, and root’s ranks.

Table 25: Probe and Cancel functional tests

Test path	Description
c/probe_cancel/functional/ MPI_Cancel_persist_send	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active persistent send request(s). Another persistent send request will be posted afterward with a matching recv request. Looping through various communicators, message lengths, and root’s ranks.
c/probe_cancel/functional/ MPI_Cancel_some	Calling MPI_Cancel() and MPI_Test_cancelled() on subset of active isend request(s) and letting other active isend request(s) completed (MPI_Wait()).Looping through various communicators, message lengths, and root’s ranks.
c/probe_cancel/functional/ MPI_Iprobe_return	Calling MPI_Iprobe() without any message pending and make sure the call returns. Looping through various communicators, and receiver’s rank.
c/probe_cancel/functional/ MPI_Iprobe_source	Calling MPI_Iprobe() with message sent from various source ranks. Looping through various communicators, message lengths, and receiver’s ranks.
c/probe_cancel/functional/ MPI_Iprobe_tag	Calling MPI_Iprobe() with message sent with various tags. Looping through various communicators, message lengths, and sender’s ranks.
c/probe_cancel/functional/ MPI_Probe_source	Calling MPI_Probe() with message sent from various source ranks. Looping through various communicators, message lengths, and receiver’s ranks.

Table 25: Probe and Cancel functional tests

Test path	Description
c/probe_cancel/functional/MPI_Probe_tag	Calling MPI_Probe() with message sent with various tags. Looping through various communicators, message lengths, and sender's ranks.
c/probe_cancel/functional/ MPI_Test_cancelled_false	Calling MPI_Test_cancelled() with active isend request that has not been cancelled. Looping through various communicators, message lengths, and sender's ranks.

Table 26: Probe and Cancel error tests

Test path	Description
c/probe_cancel/error/MPI_Cancel_err1	Inactive request
c/probe_cancel/error/MPI_Cancel_err2	Freed request
c/probe_cancel/error/MPI_Iprobe_err1	Negative source rank
c/probe_cancel/error/MPI_Iprobe_err2	Negative tag
c/probe_cancel/error/MPI_Iprobe_err3	tag > MPI_TAG_UB
c/probe_cancel/error/MPI_Iprobe_err4	Invalid communicator (MPI_COMM_NULL)
c/probe_cancel/error/MPI_Iprobe_err5	Invalid rank (source rank = comm_size)
c/probe_cancel/error/MPI_Probe_err1	Negative source rank
c/probe_cancel/error/MPI_Probe_err2	Negative tag
c/probe_cancel/error/MPI_Probe_err3	tag > MPI_TAG_UB
c/probe_cancel/error/MPI_Probe_err4	Invalid communicator (MPI_COMM_NULL)
c/probe_cancel/error/MPI_Probe_err5	Invalid rank (source rank = comm_size)

B.1.10 Send and Recv

Table 27: Send and Recv functional tests

Test path	Description
c/sendrecv/functional/MPI_Sendrecv_null	Calling MPI_Sendrecv() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
c/sendrecv/functional/ MPI_Sendrecv_replace_null	Calling MPI_Sendrecv_replace() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
c/sendrecv/functional/ MPI_Sendrecv_replace_ring	Calling MPI_Sendrecv_replace() to send and recv message in a ring style. Test loops through various communicators, message datatypes, and message lengths.
c/sendrecv/functional/ MPI_Sendrecv_replace_rtoa	Calling MPI_Sendrecv_replace() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, and message lengths.
c/sendrecv/functional/ MPI_Sendrecv_replace_self	Calling MPI_Sendrecv_replace() to send message from each rank to itself. Test loops through various communicators, message datatypes, and message lengths.
c/sendrecv/functional/MPI_Sendrecv_ring	Calling MPI_Sendrecv() to send and recv message in a ring style. Test loops through various communicators, message datatypes, and message lengths.
c/sendrecv/functional/MPI_Sendrecv_rtoa	Calling MPI_Sendrecv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, and message lengths.

Table 27: Send and Recv functional tests

Test path	Description
c/sendrecv/functional/MPI_Sendrecv_self	Calling MPI_Sendrecv() to send message from each rank to itself. Test loops through various communicators, message datatypes, and message lengths.

Table 28: Send and Recv error tests

Test path	Description
c/sendrecv/error/MPI_Sendrecv_err1a	Negative Destination rank
c/sendrecv/error/MPI_Sendrecv_err1b	Negative Source rank
c/sendrecv/error/MPI_Sendrecv_err2a	Destination rank >= group size
c/sendrecv/error/MPI_Sendrecv_err2b	Source rank >= group size
c/sendrecv/error/MPI_Sendrecv_err3	Invalid communicator (MPI_COMM_NULL)
c/sendrecv/error/MPI_Sendrecv_err4	Freed communicator
c/sendrecv/error/MPI_Sendrecv_err5a	send tag < 0
c/sendrecv/error/MPI_Sendrecv_err5b	receive tag < 0
c/sendrecv/error/MPI_Sendrecv_err6a	send tag > MPI_TAG_UB
c/sendrecv/error/MPI_Sendrecv_err6b	receive tag > MPI_TAG_UB
c/sendrecv/error/MPI_Sendrecv_err7a	Negative send count
c/sendrecv/error/MPI_Sendrecv_err7b	Negative receive count
c/sendrecv/error/MPI_Sendrecv_err8a	Invalid send type (MPI_DATATYPE_NULL)
c/sendrecv/error/MPI_Sendrecv_err8b	Invalid receive type (MPI_DATATYPE_NULL)
c/sendrecv/error/MPI_Sendrecv_err9	Mismatching type at sender and receiver
c/sendrecv/error/ MPI_Sendrecv_replace_err1a	Negative Destination rank

Table 28: Send and Recv error tests

Test path	Description
c/sendrecv/error/ MPI_Sendrecv_replace_err1b	Negative Source rank
c/sendrecv/error/ MPI_Sendrecv_replace_err2a	Destination rank >= group size
c/sendrecv/error/ MPI_Sendrecv_replace_err2b	Source rank >= group size
c/sendrecv/error/ MPI_Sendrecv_replace_err3	Invalid communicator (MPI_COMM_NULL)
c/sendrecv/error/ MPI_Sendrecv_replace_err4	Freed communicator
c/sendrecv/error/ MPI_Sendrecv_replace_err5a	send tag < 0
c/sendrecv/error/ MPI_Sendrecv_replace_err5b	receive tag < 0
c/sendrecv/error/ MPI_Sendrecv_replace_err6a	send tag > MPI_TAG_UB
c/sendrecv/error/ MPI_Sendrecv_replace_err6b	receive tag > MPI_TAG_UB
c/sendrecv/error/ MPI_Sendrecv_replace_err7	Negative count
c/sendrecv/error/ MPI_Sendrecv_replace_err8	Invalid datatype (MPI_DATATYPE_NULL)
c/sendrecv/error/ MPI_Sendrecv_replace_err9	Mismatching type at sender and receiver

B.1.11 Topology

Table 29: Topology functional tests

Test path	Description
c/topo/functional/MPI_Cart_coords	Calling MPI_Cart_coords for various cartesian communicators created.

Table 29: Topology functional tests

Test path	Description
c/topo/functional/ MPI_Cart_create_nonperiodic	Using MPI_Cart_create() to create various non-periodic cartesian communicators
c/topo/functional/ MPI_Cart_create_periodic	Using MPI_Cart_create() to create various periodic cartesian communicators. Looping through various intracommunicators.
c/topo/functional/MPI_Cart_create_undef	Calling MPI_Cart_create() with some ranks returning MPI_UNDEFINED. Looping through various intracommunicators.
c/topo/functional/MPI_Cart_get	Calling MPI_Cart_get() with various cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Cart_map	Calling MPI_Cart_map() with various cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Cart_rank	Calling MPI_Cart_rank() with various cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/ MPI_Cart_shift_nonperiodic	Calling MPI_Cart_shift() with various non-periodic cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/ MPI_Cart_shift_periodic	Calling MPI_Cart_shift() with various periodic cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Cart_sub	Calling MPI_Cart_sub() with various cartesian communicators created. Looping through various intracommunicators.

Table 29: Topology functional tests

Test path	Description
c/topo/functional/MPI_Cartdim_get	Calling MPI_Cartdim_get() with various cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/ MPI_Dims_create_system	Calling MPI_Dims_create() with all output dimensions set up by MPI. Looping through various intracommunicators.
c/topo/functional/MPI_Dims_create_user	Calling MPI_Dims_create() with some output dimensions set up by MPI and some are set by user. Looping through various intracommunicators.
c/topo/functional/ MPI_Graph_create_noreorder	Calling MPI_Graph_create() with reorder = FALSE. Looping through various intracommunicators.
c/topo/functional/ MPI_Graph_create_reorder	Calling MPI_Graph_create() with reorder = TRUE. Looping through various intracommunicators.
c/topo/functional/ MPI_Graph_create_undef	Calling MPI_Graph_create() such that some ranks will return MPI_UNDEFINED. Looping through various intracommunicators.
c/topo/functional/MPI_Graph_get	Calling MPI_Graph_get() with various graph communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Graph_map	Calling MPI_Graph_map() with various graph communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Graph_neighbors	Calling MPI_Graph_neighbors() with various graph communicators created. Looping through various intracommunicators.

Table 29: Topology functional tests

Test path	Description
c/topo/functional/ MPI_Graph_neighbors_count	Calling MPI_Graph_count() with various graph communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Graphdims_get	Calling MPI_Graphdims_get() with various graph communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Topo_test_cart	Calling MPI_Topo_test() with various cartesian communicators created. Looping through various intracommunicators.
c/topo/functional/MPI_Topo_test_graph	Calling MPI_Topo_test() with various graph communicators created. Looping through various intracommunicators.

Table 30: Topology error tests

Test path	Description
c/topo/error/MPI_Cart_coords_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Cart_coords_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Cart_coords_err3	Invalid communicator (communicator with graph topology)
c/topo/error/MPI_Cart_create_err1	Fewer processes than dimensions
c/topo/error/MPI_Cart_create_err2	Dimensions = 0
c/topo/error/MPI_Cart_create_err3	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Cart_get_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Cart_get_err2	Invalid communicator (MPI_COMM_NULL)

Table 30: Topology error tests

Test path	Description
c/topo/error/MPI_Cart_get_err3	Invalid communicator (communicator with graph topology)
c/topo/error/MPI_Cart_get_err4	Negative dimension
c/topo/error/MPI_Cart_map_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Cart_map_err3	dimensions = 0
c/topo/error/MPI_Cart_map_err5	More ranks than available
c/topo/error/MPI_Cart_rank_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Cart_rank_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Cart_rank_err3	Invalid communicator (communicator with graph topology)
c/topo/error/MPI_Cart_shift_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Cart_shift_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Cart_shift_err3	Invalid communicator (communicator with graph topology)
c/topo/error/MPI_Cart_shift_err4	displacement = 0
c/topo/error/MPI_Cart_shift_err5	Invalid shift direction value
c/topo/error/MPI_Cart_sub_err2	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Cart_sub_err3	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Cart_sub_err4	Invalid communicator (communicator with graph topology)
c/topo/error/MPI_Cartdim_get_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Cartdim_get_err2	Invalid communicator (MPI_COMM_NULL)

Table 30: Topology error tests

Test path	Description
c/topo/error/MPI_Cartdim_get_err3	Invalid communicator (communicator with graph topology)
c/topo/error/MPI_Dims_create_err1	nodes not a multiple of non-zero dimensions
c/topo/error/MPI_Dims_create_err2	Negative dimension
c/topo/error/MPI_Dims_create_err3	Negative nodes
c/topo/error/MPI_Dims_create_err4	Number of dimensions < 0
c/topo/error/MPI_Graph_create_err1	Negative nodes
c/topo/error/MPI_Graph_create_err2	nodes > size of MPI_COMM_WORLD
c/topo/error/MPI_Graph_create_err3	Edge number too large
c/topo/error/MPI_Graph_create_err4	Negative edge number
c/topo/error/MPI_Graph_create_err5	Edge to itself
c/topo/error/MPI_Graph_create_err6	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Graph_get_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Graph_get_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Graph_get_err3	Invalid communicator (communicator with cartesian topology)
c/topo/error/MPI_Graph_map_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Graph_map_err3	Negative nnodes
c/topo/error/MPI_Graph_map_err4	More ranks than available
c/topo/error/MPI_Graph_neighbors_count_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Graph_neighbors_count_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Graph_neighbors_count_err3	Invalid communicator (communicator with cartesian topology)

Table 30: Topology error tests

Test path	Description
c/topo/error/MPI_Graph_neighbors_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Graph_neighbors_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Graph_neighbors_err3	Invalid communicator (communicator with cartesian topology)
c/topo/error/MPI_Graphdims_get_err1	Invalid communicator (MPI_COMM_WORLD)
c/topo/error/MPI_Graphdims_get_err2	Invalid communicator (MPI_COMM_NULL)
c/topo/error/MPI_Graphdims_get_err3	Invalid communicator (communicator with cartesian topology)
c/topo/error/MPI_Topo_test_err1	Invalid communicator (MPI_COMM_NULL)

B.2 Fortran Tests

B.2.1 Blocking Operations

Table 31: C Blocking functional tests

Test path	Description
fortran/blocking/functional/ MPI_Bsend_ator	Calling MPI_Bsend() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Bsend_null	Calling MPI_Bsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.

Table 31: C Blocking functional tests

Test path	Description
fortran/blocking/functional/ MPI_Bsend_overtake	Calling MPI_Bsend() / MPI_Recv() to send a sequence of message and make sure they are received in the order they were sent.
fortran/blocking/functional/ MPI_Bsend_rtoa	Calling MPI_Bsend() / MPI_Recv() and MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Recv_comm	Calling MPI_Recv() with various communicators created and make sure the messages are received in the order sent for each communicator (not the order the recvs were posted)
fortran/blocking/functional/ MPI_Recv_null	Calling MPI_Recv() with source of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/blocking/functional/ MPI_Recv_pack	Calling MPI_Recv() verifying that every datatype can be sent as MPI_PACKED. Test loops through various communicators, message types, and message lengths.
fortran/blocking/functional/ MPI_Rsend_null	Calling MPI_Rsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.

Table 31: C Blocking functional tests

Test path	Description
fortran/blocking/functional/ MPI_Rsend_rtoa	Calling MPI_Rsend() / MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Send_ator	Calling MPI_Send() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Send_ator2	Calling MPI_Send() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Send_null	Calling MPI_Send() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/blocking/functional/ MPI_Send_off	Calling MPI_Send() / MPI_Recv() to send message from all ranks in communicators to a selected root with byte offsets. Test loops through various communicators, message datatypes, message lengths and root' ranks.
fortran/blocking/functional/ MPI_Send_overtake	Calling MPI_Send() / MPI_Recv() to send a sequence of messages and make sure they are received in the order they were sent.

Table 31: C Blocking functional tests

Test path	Description
fortran/blocking/functional/ MPI_Send_rtoa	Calling MPI_Send() / MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Send_self	Calling MPI_Send() / MPI_Recv() to send message from a rank to itself. Test loops through various communicators.
fortran/blocking/functional/ MPI_Ssend_ator	Calling MPI_Ssend() / MPI_Recv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/blocking/functional/ MPI_Ssend_null	Calling MPI_Ssend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/blocking/functional/ MPI_Ssend_overtake	Calling MPI_Ssend() and MPI_Recv() to send a sequence of messages and make sure they are received in the order they were sent.
fortran/blocking/functional/ MPI_Ssend_rtoa	Calling MPI_Ssend() / MPI_Recv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 32: C Blocking error tests

Test path	Description
fortran/blocking/error/MPI_Bsend_err1	Invalid rank (negative).
fortran/blocking/error/MPI_Bsend_err2	Invalid rank (too large).
fortran/blocking/error/MPI_Bsend_err3	Invalid communicator (MPI_COMM_NULL).
fortran/blocking/error/MPI_Bsend_err4	Freed communicator.
fortran/blocking/error/MPI_Bsend_err5	Invalid tag (negative).
fortran/blocking/error/MPI_Bsend_err6	Invalid tag (> MPI_TAG_UB)
fortran/blocking/error/MPI_Bsend_err7	Invalid count (negative).
fortran/blocking/error/MPI_Bsend_err8	Invalid datatype (MPI_DATATYPE_NULL).
fortran/blocking/error/MPI_Bsend_err9	Mismatching types.
fortran/blocking/error/MPI_Bsend_err10	Message size > attached buffer size.
fortran/blocking/error/MPI_Bsend_err11	Message with no attached buffer
fortran/blocking/error/ MPI_Buffer_attach_err1	Attach more than 1 buffer to one process.
fortran/blocking/error/ MPI_Buffer_attach_err2	Invalid length (negative).
fortran/blocking/error/ MPI_Buffer_detach_err1	An non-attached buffer.
fortran/blocking/error/MPI_Recv_err1	Invalid rank (negative).
fortran/blocking/error/MPI_Recv_err2	Invalid rank (too large).
fortran/blocking/error/MPI_Recv_err3	Invalid communicator (MPI_COMM_NULL).
fortran/blocking/error/MPI_Recv_err4	Freed communicator.
fortran/blocking/error/MPI_Recv_err5	Invalid tag (negative).
fortran/blocking/error/MPI_Recv_err6	Invalid tag (> MPI_TAG_UB)
fortran/blocking/error/MPI_Recv_err7	Invalid count (negative).

Table 32: C Blocking error tests

Test path	Description
fortran/blocking/error/MPI_Recv_err8	Invalid datatype (MPI_DATATYPE_NULL).
fortran/blocking/error/MPI_Recv_err9	Message size > recv count.
fortran/blocking/error/MPI_Rsend_err1	Invalid rank (negative).
fortran/blocking/error/MPI_Rsend_err2	Invalid rank (too large).
fortran/blocking/error/MPI_Rsend_err3	Invalid communicator (MPI_COMM_NULL).
fortran/blocking/error/MPI_Rsend_err4	Freed communicator.
fortran/blocking/error/MPI_Rsend_err5	Invalid tag (negative).
fortran/blocking/error/MPI_Rsend_err6	Invalid tag (> MPI_TAG_UB)
fortran/blocking/error/MPI_Rsend_err7	Invalid count (negative).
fortran/blocking/error/MPI_Rsend_err8	Invalid datatype (MPI_DATATYPE_NULL).
fortran/blocking/error/MPI_Rsend_err9	Mismatching types.
fortran/blocking/error/MPI_Send_err1	Invalid rank (negative).
fortran/blocking/error/MPI_Send_err2	Invalid rank (too large).
fortran/blocking/error/MPI_Send_err3	Invalid communicator (MPI_COMM_NULL).
fortran/blocking/error/MPI_Send_err4	Freed communicator.
fortran/blocking/error/MPI_Send_err5	Invalid tag (negative).
fortran/blocking/error/MPI_Send_err6	Invalid tag (> MPI_TAG_UB)
fortran/blocking/error/MPI_Send_err7	Invalid count (negative).
fortran/blocking/error/MPI_Send_err8	Invalid datatype (MPI_DATATYPE_NULL).
fortran/blocking/error/MPI_Send_err9	Mismatching types.
fortran/blocking/error/MPI_Ssend_err1	Invalid rank (negative).
fortran/blocking/error/MPI_Ssend_err2	Invalid rank (too large).
fortran/blocking/error/MPI_Ssend_err3	Invalid communicator (MPI_COMM_NULL).

Table 32: C Blocking error tests

Test path	Description
fortran/blocking/error/MPI_Ssend_err4	Freed communicator.
fortran/blocking/error/MPI_Ssend_err5	Invalid tag (negative).
fortran/blocking/error/MPI_Ssend_err6	Invalid tag (> MPI_TAG_UB)
fortran/blocking/error/MPI_Ssend_err7	Invalid count (negative).
fortran/blocking/error/MPI_Ssend_err8	Invalid datatype (MPI_DATATYPE_NULL).
fortran/blocking/error/MPI_Ssend_err9	Mismatching types.

B.2.2 Collective Operations

Table 33: C Collective functional tests

Test path	Description
fortran/collective/functional/ MPI_Allgather	Calling MPI_Allgather() looping between various intracommunicators, various message data types and various message data lengths.
fortran/collective/functional/ MPI_Allgatherv	Calling MPI_Allgatherv() looping between various intracommunicators, various message data types and various message data lengths.
fortran/collective/functional/ MPI_Allreduce	Calling MPI_Allreduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various predefined reduction operations.
fortran/collective/functional/ MPI_Allreduce_loc	Calling MPI_Allreduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).

Table 33: C Collective functional tests

Test path	Description
fortran/collective/functional/ MPI_Allreduce_user	Calling MPI_Allreduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations
fortran/collective/functional/MPI_Alltoall	Calling MPI_Alltoall() looping between various intracommunicators, various message data types and various message data lengths.
fortran/collective/functional/ MPI_Alltoallv	Calling MPI_Alltoallv() looping between various intracommunicators, various message data types and various message data lengths.
fortran/collective/functional/MPI_Barrier	Calling MPI_Barrier() using various intracommunicators.
fortran/collective/functional/MPI_Bcast	Calling MPI_Bcast() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
fortran/collective/functional/MPI_Gather	Calling MPI_Gather() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
fortran/collective/functional/MPI_Gatherv	Calling MPI_Gatherv() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
fortran/collective/functional/MPI_Reduce	Calling MPI_Reduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various predefined reduction operations

Table 33: C Collective functional tests

Test path	Description
fortran/collective/functional/ MPI_Reduce_loc	Calling MPI_Reduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).
fortran/collective/functional/ MPI_Reduce_scatter	Calling MPI_Reduce_scatter() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various pre-defined reduction operations
fortran/collective/functional/ MPI_Reduce_scatter_loc	Calling MPI_Reduce_scatter() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).
fortran/collective/functional/ MPI_Reduce_scatter_user	Calling MPI_Reduce_scatter() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations
fortran/collective/functional/ MPI_Reduce_user	Calling MPI_Reduce() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations

Table 33: C Collective functional tests

Test path	Description
fortran/collective/functional/MPI_Scan	Calling MPI_Scan() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various predefined reduction operations
fortran/collective/functional/ MPI_Scan_loc	Calling MPI_Scan() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and 2 reduction operations (MPI_MAXLOC & MPI_MINLOC).
fortran/collective/functional/ MPI_Scan_user	Calling MPI_Scan() looping between various intracommunicators, various message data types, various message data lengths, various root ranks and various user defined reduction operations
fortran/collective/functional/MPI_Scatter	Calling MPI_Scatter() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
fortran/collective/functional/ MPI_Scatterv	Calling MPI_Scatterv() looping between various intracommunicators, various message data types, various message data lengths and various root ranks.
fortran/collective/functional/ MPI_collective_message	Verifying that collective operation does not interfere with ongoing message traffic.

Table 34: C Collective error tests

Test path	Description
fortran/collective/error/ MPI_Allgather_err1	Mis-matched types
fortran/collective/error/ MPI_Allgather_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/ MPI_Allgather_err3	Invalid size (-1)
fortran/collective/error/ MPI_Allgather_err4	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/ MPI_Allgather_err5	Freed communicator
fortran/collective/error/ MPI_Allgather_err6	Inter-communicator
fortran/collective/error/ MPI_Allgatherv_err1	Mis-matched types
fortran/collective/error/ MPI_Allgatherv_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/ MPI_Allgatherv_err3	Invalid size (-1)
fortran/collective/error/ MPI_Allgatherv_err4	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/ MPI_Allgatherv_err5	Freed communicator
fortran/collective/error/ MPI_Allgatherv_err6	Inter-communicator
fortran/collective/error/ MPI_Allreduce_err1	Invalid type/operator pair
fortran/collective/error/ MPI_Allreduce_err2	Mis-matched types
fortran/collective/error/ MPI_Allreduce_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/ MPI_Allreduce_err4	Invalid size (-1)

Table 34: C Collective error tests

Test path	Description
fortran/collective/error/MPI_Allreduce_err5	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Allreduce_err6	Freed communicator
fortran/collective/error/MPI_Allreduce_err7	Inter-communicator
fortran/collective/error/MPI_Alltoall_err1	Mis-matched types
fortran/collective/error/MPI_Alltoall_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Alltoall_err3	Invalid size (-1)
fortran/collective/error/MPI_Alltoall_err4	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Alltoall_err5	Freed communicator
fortran/collective/error/MPI_Alltoall_err6	Inter-communicator
fortran/collective/error/MPI_Alltoallv_err1	Mis-matched types
fortran/collective/error/MPI_Alltoallv_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Alltoallv_err3	Invalid size (-1)
fortran/collective/error/MPI_Alltoallv_err4	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Alltoallv_err5	Freed communicator
fortran/collective/error/MPI_Alltoallv_err6	Inter-communicator
fortran/collective/error/MPI_Barrier_err1	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Barrier_err2	Freed communicator
fortran/collective/error/MPI_Barrier_err3	Inter-communicator
fortran/collective/error/MPI_Bcast_err1	Mis-matched types

Table 34: C Collective error tests

Test path	Description
fortran/collective/error/MPI_Bcast_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Bcast_err3	Invalid size (-1)
fortran/collective/error/MPI_Bcast_err4	Invalid root (-1)
fortran/collective/error/MPI_Bcast_err5	Invalid root (> number of ranks)
fortran/collective/error/MPI_Bcast_err6	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Bcast_err7	Freed communicator
fortran/collective/error/MPI_Bcast_err8	Inter-communicator
fortran/collective/error/MPI_Gather_err1	Mis-matched types
fortran/collective/error/MPI_Gather_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Gather_err3	Invalid size (-1)
fortran/collective/error/MPI_Gather_err4	Invalid root (-1)
fortran/collective/error/MPI_Gather_err5	Invalid root (> number of ranks)
fortran/collective/error/MPI_Gather_err6	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Gather_err7	Freed communicator
fortran/collective/error/MPI_Gather_err8	Inter-communicator
fortran/collective/error/MPI_Gatherv_err1	Mis-matched types
fortran/collective/error/MPI_Gatherv_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Gatherv_err3	Invalid size (-1)
fortran/collective/error/MPI_Gatherv_err4	Invalid root (-1)
fortran/collective/error/MPI_Gatherv_err5	Invalid root (> number of ranks)
fortran/collective/error/MPI_Gatherv_err6	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Gatherv_err7	Freed communicator
fortran/collective/error/MPI_Gatherv_err8	Inter-communicator

Table 34: C Collective error tests

Test path	Description
fortran/collective/error/ MPI_Op_free_err1	Invalid operator (MPI_OP_NULL)
fortran/collective/error/ MPI_Op_free_err2	Pre-defined operators
fortran/collective/error/MPI_Reduce_err1	Invalid type/operator pair
fortran/collective/error/MPI_Reduce_err2	Mis-matched types
fortran/collective/error/MPI_Reduce_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Reduce_err4	Invalid size (-1)
fortran/collective/error/MPI_Reduce_err5	Invalid root (-1)
fortran/collective/error/MPI_Reduce_err6	Invalid root (> number of ranks)
fortran/collective/error/MPI_Reduce_err7	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Reduce_err8	Freed communicator
fortran/collective/error/MPI_Reduce_err9	Inter-communicator
fortran/collective/error/ MPI_Reduce_scatter_err1	Invalid type/operator pair
fortran/collective/error/ MPI_Reduce_scatter_err2	Mis-matched types
fortran/collective/error/ MPI_Reduce_scatter_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/ MPI_Reduce_scatter_err4	Invalid size (-1)
fortran/collective/error/ MPI_Reduce_scatter_err5	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/ MPI_Reduce_scatter_err6	Freed communicator
fortran/collective/error/ MPI_Reduce_scatter_err7	Inter-communicator
fortran/collective/error/MPI_Scan_err1	Invalid type/operator pair
fortran/collective/error/MPI_Scan_err2	Mis-matched types

Table 34: C Collective error tests

Test path	Description
fortran/collective/error/MPI_Scan_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Scan_err4	Invalid size (-1)
fortran/collective/error/MPI_Scan_err5	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Scan_err6	Freed communicator
fortran/collective/error/MPI_Scan_err7	Inter-communicator
fortran/collective/error/MPI_Scatter_err1	Mis-matched types
fortran/collective/error/MPI_Scatter_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/MPI_Scatter_err3	Invalid size (-1)
fortran/collective/error/MPI_Scatter_err4	Invalid root (-1)
fortran/collective/error/MPI_Scatter_err5	Invalid root (> number of ranks)
fortran/collective/error/MPI_Scatter_err6	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/MPI_Scatter_err7	Freed communicator
fortran/collective/error/MPI_Scatter_err8	Inter-communicator
fortran/collective/error/ MPI_Scatterv_err1	Mis-matched types
fortran/collective/error/ MPI_Scatterv_err2	Invalid type (MPI_DATATYPE_NULL)
fortran/collective/error/ MPI_Scatterv_err3	Invalid size (-1)
fortran/collective/error/ MPI_Scatterv_err4	Invalid root (-1)
fortran/collective/error/ MPI_Scatterv_err5	Invalid root (> number of ranks)
fortran/collective/error/ MPI_Scatterv_err6	Invalid communicator (MPI_COMM_NULL)
fortran/collective/error/ MPI_Scatterv_err7	Freed communicator

Table 34: C Collective error tests

Test path	Description
fortran/collective/error/ MPI_Scatterv_err8	Inter-communicator

B.2.3 Derived Datatype Operations

Table 35: Derived datatype functional tests

Test path	Description
fortran/datatype/functional/MPI_Address	Calling MPI_Address() looping over various length buffer address
fortran/datatype/functional/ MPI_Get_elements_basic_type	Calling MPI_Get_elements_basic_type() looping over various communicators, message pre-defined data types, message data lengths and root ranks
fortran/datatype/functional/ MPI_Pack_displs	Calling MPI_Pack() and MPI_Unpack() using basic types with non-zero displacement space between any 2 adjacent types. The test loops over various communicators, message lengths and root ranks.
fortran/datatype/functional/ MPI_Pack_size_basic	Calling MPI_Pack_size() with basic types. The test loops over various communicators, lengths and datatypes.
fortran/datatype/functional/ MPI_Pack_size_types	Calling MPI_Pack_size() with user defined types. The test loops through various communicators and lengths.
fortran/datatype/functional/ MPI_Pack_user_type	Calling MPI_Pack() and MPI_Unpack() with user defined types. The test loops through various communicators, and lengths.

Table 35: Derived datatype functional tests

Test path	Description
fortran/datatype/functional/ MPI_Type_contiguous_basic	Calling MPI_Type_contiguous() with basic types. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_contiguous_idispls	Calling MPI_Type_contiguous() with user defined types which has non-zero inner displacement between each basic type. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_contiguous_types	Calling MPI_Type_contiguous() with user defined types, Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_extent_MPI_LB_UB	Calling MPI_Type_extent() with MPI_LB and MPI_UB. test loops through various communicators.
fortran/datatype/functional/ MPI_Type_extent_types	Calling MPI_Type_extent() with user defined types. Test loops through various communicators, and lengths.
fortran/datatype/functional/ MPI_Type_free_pending_msg	Calling MPI_Type_free() with user defined type that is in use by some active data transmission. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_free_types	Calling MPI_Type_free() with user defined type. Test loops through various communicators, lengths, and root's ranks.
fortran/datatype/functional/ MPI_Type_hindexed_basic	Calling MPI_Type_hindexed() with basic types. Test loops through various communicators, lengths, and sender's ranks.

Table 35: Derived datatype functional tests

Test path	Description
fortran/datatype/functional/ MPI_Type_hindexed_blklen	Calling MPI_Type_hindexed() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_hindexed_displs	Calling MPI_Type_hindexed() with user defined types using non-zero displacement greater than the type size of each entry of the user defined type. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_hindexed_types	Calling MPI_Type_hindexed() with user defined type. test loops through various communicators, lengths, and root's ranks.
fortran/datatype/functional/ MPI_Type_hvector_basic	Calling MPI_Type_hvector() with basic types. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_hvector_blklen	Calling MPI_Type_hvector() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_hvector_stride	Calling MPI_Type_hvector() with user defined type with stride larger than the type size of the user defined type. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_hvector_types	Calling MPI_Type_hvector() with user defined type. test loops through various communicators, lengths, and root's ranks.

Table 35: Derived datatype functional tests

Test path	Description
fortran/datatype/functional/ MPI_Type_indexed_basic	Calling MPI_Type_indexed() with basic types. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_indexed_blklen	Calling MPI_Type_indexed() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_indexed_displs	Calling MPI_Type_indexed() with user defined types using nonzero displacement greater than the type size of each entry of the user defined type. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_indexed_types	Calling MPI_Type_indexed() with user defined type. Test loops through various communicators, lengths, and root's ranks.
fortran/datatype/functional/ MPI_Type_lb_2MPI_LB	Calling MPI_Type_lb() with 2 user defined types which each contains MPI_LB. Test loops through various communicators.
fortran/datatype/functional/ MPI_Type_lb_MPI_LB	Calling MPI_Type_lb() with user defined type which contains MPI_LB. Test loops through various communicators.
fortran/datatype/functional/ MPI_Type_lb_neg_displ	Calling MPI_Type_lb() with user defined type which has negative displacement. Test loops through various communicators.
fortran/datatype/functional/ MPI_Type_lb_pos_displ	Calling MPI_Type_lb() with user defined type which has positive displacement. Test loops through various communicators and data types.

Table 35: Derived datatype functional tests

Test path	Description
fortran/datatype/functional/ MPI_Type_size_MPI_LB_UB	Calling MPI_Type_size() with MPI_LB and MPI_UB. Test loops through various communicators and data types.
fortran/datatype/functional/ MPI_Type_size_basic	Calling MPI_Type_size() with basic types. Test loops through various communicators,
fortran/datatype/functional/ MPI_Type_size_types	Calling MPI_Type_size() with user defined type. Test loops through various communicators, and lengths.
fortran/datatype/functional/ MPI_Type_struct_basic	Calling MPI_Type_struct() with basic types. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_struct_blklen	Calling MPI_Type_struct() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_struct_displs	Calling MPI_Type_struct() with user defined types using nonzero displacement greater than the type size of each entry of the user defined type. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_struct_types	Calling MPI_Type_struct() with basic and user defined type. Test loops through various communicators, lengths, and root's ranks.
fortran/datatype/functional/ MPI_Type_ub_2MPI_UB	Calling MPI_Type_ub() with 2 user defined types which each contains MPI_UB. Test loops through various communicators.

Table 35: Derived datatype functional tests

Test path	Description
fortran/datatype/functional/ MPI_Type_ub_MPI_UB	Calling MPI_Type_ub() with user defined type which contains MPI_UB. Test loops through various communicators.
fortran/datatype/functional/ MPI_Type_ub_neg_displ	Calling MPI_Type_ub() with user defined type which has negative displacement. Test loops through various communicators.
fortran/datatype/functional/ MPI_Type_ub_pos_displ	Calling MPI_Type_ub() with user defined type which has positive displacement. Test loops through various communicators.
fortran/datatype/functional/ MPI_Type_vector_basic	Calling MPI_Type_vector() with basic types. Test loops through various communicators, lengths, and sender's ranks.
fortran/datatype/functional/ MPI_Type_vector_blklen	Calling MPI_Type_vector() with user defined types using block length > 1. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_vector_stride	Calling MPI_Type_vector() with user defined type with stride larger than the type size of the user defined type. Test loops through various communicators, lengths and sender's rank.
fortran/datatype/functional/ MPI_Type_vector_types	Calling MPI_Type_vector() with basic and user defined type. Test loops through various communicators, lengths, and sender's ranks.

Table 36: C Derived datatype error tests

Test path	Description
fortran/datatype/error/ MPI_Get_elements_err1	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Get_elements_err2	Uncommitted datatype
fortran/datatype/error/MPI_Pack_err1	Negative input count
fortran/datatype/error/MPI_Pack_err2	Negative outsize
fortran/datatype/error/MPI_Pack_err3	Invalid communicator (MPI_COMM_NULL)
fortran/datatype/error/MPI_Pack_err4	Invalid datatype (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Pack_err5	Uncommitted datatype
fortran/datatype/error/MPI_Pack_err6	outsize < incount with basic type
fortran/datatype/error/ MPI_Pack_size_err1	Negative input incount
fortran/datatype/error/ MPI_Pack_size_err2	Invalid communicator (MPI_COMM_NULL)
fortran/datatype/error/ MPI_Pack_size_err3	Invalid datatype (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Pack_size_err4	Uncommitted datatype
fortran/datatype/error/ MPI_Type_commit_err1	Invalid datatype (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Type_contiguous_err1	Negative input count
fortran/datatype/error/ MPI_Type_contiguous_err2	Invalid datatype (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Type_extent_err1	Invalid datatype (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Type_free_err1	Invalid datatype (MPI_DATATYPE_NULL)

Table 36: C Derived datatype error tests

Test path	Description
fortran/datatype/error/ MPI_Type_free_err2	Predefined type (MPI_BYTE)
fortran/datatype/error/ MPI_Type_free_err3	Predefined type (MPI_CHARACTER)
fortran/datatype/error/ MPI_Type_free_err4	Predefined type (MPI_DOUBLE_PRECISION)
fortran/datatype/error/ MPI_Type_free_err5	Predefined type (MPI_REAL)
fortran/datatype/error/ MPI_Type_free_err6	Predefined type (MPI_INTEGER)
fortran/datatype/error/ MPI_Type_free_err7	Predefined type (MPI_PACKED)
fortran/datatype/error/ MPI_Type_free_err8	Predefined type (MPI_COMPLEX)
fortran/datatype/error/ MPI_Type_free_err9	Predefined type (MPI_DOUBLE_COMPLEX)
fortran/datatype/error/ MPI_Type_free_err10	Predefined type (MPI_LOGICAL)
fortran/datatype/error/ MPI_Type_hindexed_err1	Negative input count
fortran/datatype/error/ MPI_Type_hindexed_err2	Negative blocklength
fortran/datatype/error/ MPI_Type_hindexed_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Type_hvector_err1	Negative input count
fortran/datatype/error/ MPI_Type_hvector_err2	Negative blocklength
fortran/datatype/error/ MPI_Type_hvector_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/ MPI_Type_indexed_err1	Negative input count

Table 36: C Derived datatype error tests

Test path	Description
fortran/datatype/error/MPI_Type_indexed_err2	Negative blocklength
fortran/datatype/error/MPI_Type_indexed_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Type_lb_err1	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Type_size_err1	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Type_struct_err1	Negative input count
fortran/datatype/error/MPI_Type_struct_err2	Negative blocklength
fortran/datatype/error/MPI_Type_struct_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Type_ub_err1	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Type_vector_err1	Negative input count
fortran/datatype/error/MPI_Type_vector_err2	Negative blocklength
fortran/datatype/error/MPI_Type_vector_err3	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Unpack_err1	Negative insize
fortran/datatype/error/MPI_Unpack_err2	Negative outcount
fortran/datatype/error/MPI_Unpack_err3	Invalid communicator (MPI_COMM_NULL)
fortran/datatype/error/MPI_Unpack_err4	Invalid type (MPI_DATATYPE_NULL)
fortran/datatype/error/MPI_Unpack_err5	Uncommitted datatype
fortran/datatype/error/MPI_Unpack_err6	outcount < insize

B.2.4 Environment Manager

Table 37: Environment Manager functional tests

Test path	Description
fortran/env_manager/functional/ MPI_Abort	Calling MPI_Abort() from rank 0 in MPI_COMM_WORLD.
fortran/env_manager/functional/ MPI_Errhandler_fatal	Setting MPI_ERRORS_ARE_FATAL using MPI_Errhandler_set().
fortran/env_manager/functional/ MPI_Errhandler_free	Freeing user defined error handler using MPI_Errhandler_free().
fortran/env_manager/functional/ MPI_Errhandler_get	Setting user defined error handler and retrieving it using MPI_Errhandler_get().
fortran/env_manager/functional/ MPI_Errhandler_set	Setting user defined error handler using MPI_Errhandler_set().
fortran/env_manager/functional/ MPI_Error_string	Calling MPI_Error_string for every standard error class.
fortran/env_manager/functional/ MPI_Finalize	Calling MPI_Finalize() from all ranks.
fortran/env_manager/functional/ MPI_Get_processor_name	Calling MPI_Get_processor_name() from each rank.
fortran/env_manager/functional/ MPI_Init_attr	Calling MPI_Attr_get() to get all the predefined attributes (MPI_TAG_UB, MPI_HOST, MPI_WTIME_IS_GLOBAL, MPI_IO).
fortran/env_manager/functional/ MPI_Initialized	Calling MPI_Initialized() before and after MPI_Init().
fortran/env_manager/functional/ MPI_Pcontrol	Calling MPI_Pcontrol() with various levels.
fortran/env_manager/functional/ MPI_Wtime	Calling MPI_Wtime() and MPI_Wtick() from all ranks in MPI_COMM_WORLD.

Table 38: Environment Manager functional tests

Test path	Description
fortran/env_manager/error/ MPI_Abort_err1	Invalid communicator (MPI_COMM_NULL)
fortran/env_manager/error/ MPI_Errhandler_free_err1	Invalid error handler (MPI_ERRHANDLER_NULL)
fortran/env_manager/error/ MPI_Errhandler_get_err1	Invalid communicator (MPI_COMM_NULL)
fortran/env_manager/error/ MPI_Errhandler_get_err2	Freed communicator
fortran/env_manager/error/ MPI_Errhandler_set_err1	Invalid communicator (MPI_COMM_NULL)
fortran/env_manager/error/ MPI_Errhandler_set_err2	Freed communicator
fortran/env_manager/error/ MPI_Errhandler_set_err3	Invalid error handler (MPI_ERRHANDLER_NULL)
fortran/env_manager/error/MPI_Init_err1	Calling MPI_Init() twice

B.2.5 Group, Context, and Communicator Operations

Table 39: Group, Context, and Communicator functional tests

Test path	Description
fortran/grp_ctxt_comm/functional/ MPI_Comm_compare	Using MPI_Comm_compare() with MPI_IDENT on various user created communicators and pre-defined communicators (MPI_COMM_WORLD, MPI_COMM_SELF).
fortran/grp_ctxt_comm/functional/ MPI_Comm_create	Calling MPI_Comm_create() to create communicators from arbitrary group of ranks.

Table 39: Group, Context, and Communicator functional tests

Test path	Description
fortran/grp_ctxt_comm/functional/ MPI_Comm_dup	Calling MPI_Comm_dup() to duplicate various communicators making sure they really are separate communicators.
fortran/grp_ctxt_comm/functional/ MPI_Comm_group	Calling MPI_Comm_group() to retrieve groups from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Comm_split1	Calling MPI_Comm_split() to split various intracommunicators into communicators of size one.
fortran/grp_ctxt_comm/functional/ MPI_Comm_split2	Calling MPI_Comm_split() to split various intracommunicators into communicators the same as the original one.
fortran/grp_ctxt_comm/functional/ MPI_Comm_split3	Calling MPI_Comm_split() to split various intracommunicators into communicators of half of the original size.
fortran/grp_ctxt_comm/functional/ MPI_Comm_split4	Calling MPI_Comm_split() to split various intracommunicators into 2 communicators with some non-zero set of ranks not being in either.
fortran/grp_ctxt_comm/functional/ MPI_Group_compare	Calling MPI_Group_compare() with various groups.
fortran/grp_ctxt_comm/functional/ MPI_Group_difference1	Calling MPI_Group_difference() with arbitrary groups and (MPI_EMPTY_GROUP, or MPI_COMM_WORLD).
fortran/grp_ctxt_comm/functional/ MPI_Group_difference2	Calling MPI_Group_difference() with arbitrary non-overlapping groups of MPI_COMM_WORLD.
fortran/grp_ctxt_comm/functional/ MPI_Group_difference3	Calling MPI_Group_difference() with arbitrary overlapping groups of MPI_COMM_WORLD.

Table 39: Group, Context, and Communicator functional tests

Test path	Description
fortran/grp_ctxt_comm/functional/ MPI_Group_excl1	Calling MPI_Group_excl() to create groups with various ranks from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_excl2	Calling MPI_Group_excl() to create groups of size one from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_incl1	Calling MPI_Group_incl() to create groups with various ranks from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_incl2	Calling MPI_Group_incl() to create groups of size one from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_intersection1	Calling MPI_Group_intersection() with arbitrary groups and (MPI_EMPTY_GROUP, or MPI_COMM_WORLD).
fortran/grp_ctxt_comm/functional/ MPI_Group_intersection2	Calling MPI_Group_intersection() with arbitrary non-overlapping groups of MPI_COMM_WORLD.
fortran/grp_ctxt_comm/functional/ MPI_Group_intersection3	Calling MPI_Group_intersection() with arbitrary overlapping groups of MPI_COMM_WORLD.
fortran/grp_ctxt_comm/functional/ MPI_Group_range_excl1	Calling MPI_Group_range_excl() to create groups with various ranks from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_range_excl2	Calling MPI_Group_range_excl() to create groups of size one from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_range_excl3	Calling MPI_Group_range_excl() to create groups excluding every other rank from arbitrary communicators.

Table 39: Group, Context, and Communicator functional tests

Test path	Description
fortran/grp_ctxt_comm/functional/ MPI_Group_range_incl1	Calling MPI_Group_range_incl() to create groups with various ranks from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_range_incl2	Calling MPI_Group_range_incl() to create groups of size one from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_range_incl3	Calling MPI_Group_range_incl() to create groups excluding every other rank from arbitrary communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_trans_ranks	Calling MPI_Group_compare() with groups from various communicators.
fortran/grp_ctxt_comm/functional/ MPI_Group_union1	Calling MPI_Group_union() with arbitrary groups and (MPI_EMPTY_GROUP, or MPI_COMM_WORLD).
fortran/grp_ctxt_comm/functional/ MPI_Group_union2	Calling MPI_Group_union() with various groups.
fortran/grp_ctxt_comm/functional/ MPI_Group_union3	Calling MPI_Group_intersection() with arbitrary overlapping groups of MPI_COMM_WORLD.
fortran/grp_ctxt_comm/functional/ MPI_Intercomm_create1	Calling MPI_Comm_split() to split various communicators (either half or 1/3) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators.
fortran/grp_ctxt_comm/functional/ MPI_Intercomm_create2	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators.

Table 39: Group, Context, and Communicator functional tests

Test path	Description
fortran/grp_ctxt_comm/functional/ MPI_Intercomm_merge1	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators. Then call MPI_Intercomm_merge()
fortran/grp_ctxt_comm/functional/ MPI_Intercomm_merge2	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators. Then call MPI_Intercomm_merge()
fortran/grp_ctxt_comm/functional/ MPI_Intercomm_merge3	Calling MPI_Comm_split() to split various communicators (either half or 1/3 split) and calling MPI_Intercomm_create() to create intercommunicators using the split communicators. Then call MPI_Intercomm_merge()
fortran/grp_ctxt_comm/functional/ MPI_Keyval1	Creating MPI Keyvals and assigning attributes in various communicators.
fortran/grp_ctxt_comm/functional/ MPI_Keyval2	Verifying attribute values can be changed.
fortran/grp_ctxt_comm/functional/ MPI_Keyval3	Verifying errors from copy and delete functions are returned to an application.

Table 40: Group, Context, and Communicator error tests

Test path	Description
fortran/grp_ctxt_comm/error/ MPI_Attr_delete_err1	Invalid communicator (MPI_COMM_NULL)

Table 40: Group, Context, and Communicator error tests

Test path	Description
fortran/grp_ctxt_comm/error/ MPI_Attr_delete_err2	Invalid keyval (MPI_KEYVAL_INVALID)
fortran/grp_ctxt_comm/error/ MPI_Attr_get_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Attr_get_err2	Invalid keyval (MPI_KEYVAL_INVALID)
fortran/grp_ctxt_comm/error/ MPI_Attr_put_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Attr_put_err2	Invalid keyval (MPI_KEYVAL_INVALID)
fortran/grp_ctxt_comm/error/ MPI_Comm_compare_err1	Invalid communicator (comm1=MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_compare_err2	Invalid communicator (comm2=MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_create_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_create_err2	Invalid Group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_dup_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_free_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_free_err2	Invalid communicator (MPI_COMM_WORLD)
fortran/grp_ctxt_comm/error/ MPI_Comm_free_err3	Invalid communicator (MPI_COMM_SELF)
fortran/grp_ctxt_comm/error/ MPI_Comm_group_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_rank_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_remote_group_err1	Invalid communicator (MPI_COMM_NULL)

Table 40: Group, Context, and Communicator error tests

Test path	Description
fortran/grp_ctxt_comm/error/ MPI_Comm_remote_group_err2	Invalid communicator (MPI_COMM_WORLD)
fortran/grp_ctxt_comm/error/ MPI_Comm_remote_size_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_remote_size_err2	Invalid communicator (MPI_COMM_WORLD)
fortran/grp_ctxt_comm/error/ MPI_Comm_size_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_split_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Comm_test_inter_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_compare_err1	Invalid group (group1=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_compare_err2	Invalid group (group2=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_difference_err1	Invalid group (group1=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_difference_err2	Invalid group (group2=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_excl_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_excl_err2	Negative n
fortran/grp_ctxt_comm/error/ MPI_Group_excl_err3	n > size of group
fortran/grp_ctxt_comm/error/ MPI_Group_excl_err4	Negative rank
fortran/grp_ctxt_comm/error/ MPI_Group_excl_err5	rank >= size of group
fortran/grp_ctxt_comm/error/ MPI_Group_excl_err6	Duplicated ranks

Table 40: Group, Context, and Communicator error tests

Test path	Description
fortran/grp_ctxt_comm/error/ MPI_Group_free_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_incl_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_incl_err2	Negative n
fortran/grp_ctxt_comm/error/ MPI_Group_incl_err3	n > size of group
fortran/grp_ctxt_comm/error/ MPI_Group_incl_err4	Negative rank
fortran/grp_ctxt_comm/error/ MPI_Group_incl_err5	rank >= size of group
fortran/grp_ctxt_comm/error/ MPI_Group_incl_err6	Duplicated ranks
fortran/grp_ctxt_comm/error/ MPI_Group_intersection_err1	Invalid group (group1=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_intersection_err2	Invalid group (group2=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err2	Negative n
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err3	ranges[] implies negative ranks
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err4	ranges[] implies rank >= size of group
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err5	Negative stride and first < last
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err6	Positive stride and first > last
fortran/grp_ctxt_comm/error/ MPI_Group_range_excl_err7	stride = 0

Table 40: Group, Context, and Communicator error tests

Test path	Description
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err2	Negative n
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err3	ranges[] implies negative ranks
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err4	ranges[] implies rank >= size of group
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err5	Negative stride and first < last
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err6	Positive stride and first > last
fortran/grp_ctxt_comm/error/ MPI_Group_range_incl_err7	stride = 0
fortran/grp_ctxt_comm/error/ MPI_Group_rank_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_size_err1	Invalid group (MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err1	Invalid group (group1=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err2	Invalid group (group2=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err3	Negative n
fortran/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err4	Negative argument in ranks1
fortran/grp_ctxt_comm/error/ MPI_Group_trans_ranks_err5	rank >= size of group in ranks1
fortran/grp_ctxt_comm/error/ MPI_Group_union_err1	Invalid group (group1=MPI_GROUP_NULL)
fortran/grp_ctxt_comm/error/ MPI_Group_union_err2	Invalid group (group2=MPI_GROUP_NULL)

Table 40: Group, Context, and Communicator error tests

Test path	Description
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err1	Invalid communicator (local_comm= MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err2	Invalid communicator (peer_comm= MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err3	local_comm is intercommunicator
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err5	Local and remote leaders not in peer_comm
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err7	local_leader = MPI_ANY_SOURCE
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err8	remote_leader = MPI_ANY_SOURCE
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err9	local_leader not a valid rank in local_comm
fortran/grp_ctxt_comm/error/ MPI_Intercomm_create_err10	remote_leader not a valid rank in peer_comm
fortran/grp_ctxt_comm/error/ MPI_Intercomm_merge_err1	Invalid communicator (MPI_COMM_NULL)
fortran/grp_ctxt_comm/error/ MPI_Intercomm_merge_err2	Invalid communicator (intercom- municator)
fortran/grp_ctxt_comm/error/ MPI_Intercomm_merge_err3	Different values of high within a single group
fortran/grp_ctxt_comm/error/ MPI_Keyval_free_err1	Invalid keyval (MPI_KEYVAL_INVALID)

B.2.6 Miscllany Operations

Table 41: Miscllany functional tests

Test path	Description
fortran/misc/functional/MPI_defs	Test to verify all required MPI constants and routines are defined and all Fortran named constants except MPI_BOTTOM can be used in a PARAMETER statement.
fortran/misc/functional/MPI_defs2	Test to verify all required MPI constants and routines are defined.
fortran/misc/functional/MPI_pdefs	Test to verify all required MPI constants and routines are defined correctly for profiling.

B.2.7 Nonblocking Operations

Table 42: Nonblocking functional tests

Test path	Description
fortran/nonblocking/functional/MPI_Ibsend_null	Calling MPI_Ibsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/nonblocking/functional/MPI_Ibsend_overtake	Calling MPI_Ibsend() and MPI_Irecv() to send a sequence of messages and make sure they are received in the order they were sent.
fortran/nonblocking/functional/MPI_Ibsend_rtoa	Calling MPI_Ibsend() / MPI_Irecv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 42: Nonblocking functional tests

Test path	Description
fortran/nonblocking/functional/ MPI_Irecv_comm	Calling MPI_Irecv() with various communicators created and make sure the messages are received in the order sent for each communicator (not the order the recvs were posted).
fortran/nonblocking/functional/ MPI_Irecv_null	Calling MPI_Irecv() with source of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/nonblocking/functional/ MPI_Irecv_pack	Calling MPI_Irecv() verifying that every datatype can be sent as MPI_PACKED. Test loops through various communicators, message types, and message lengths.
fortran/nonblocking/functional/ MPI_Irsend_null	Calling MPI_Irsend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/nonblocking/functional/ MPI_Irsend_rtoa	Calling MPI_Irsend() / MPI_Irecv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/nonblocking/functional/ MPI_Isend_ator	Calling MPI_Isend() / MPI_Irecv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 42: Nonblocking functional tests

Test path	Description
fortran/nonblocking/functional/ MPI_Isend_ator2	Calling MPI_Isend() / MPI_Irecv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/nonblocking/functional/ MPI_Isend_null	Calling MPI_Isend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/nonblocking/functional/ MPI_Isend_off	Calling MPI_Isend() / MPI_Irecv() to send message from all ranks in communicators to a selected root with byte offsets. Test loops through various communicators, message datatypes, message lengths and root' ranks.
fortran/nonblocking/functional/ MPI_Isend_overtake	Calling MPI_Isend() / MPI_Irecv() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
fortran/nonblocking/functional/ MPI_Isend_rtoa	Calling MPI_Isend() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/nonblocking/functional/ MPI_Isend_self	Calling MPI_Isend() to send message from a rank to itself. Test loops through various communicators.

Table 42: Nonblocking functional tests

Test path	Description
fortran/nonblocking/functional/ MPI_Issend_ator	Calling MPI_Issend() / MPI_Irecv() to send message from all other ranks to the root's rank in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/nonblocking/functional/ MPI_Issend_null	Calling MPI_Issend() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/nonblocking/functional/ MPI_Issend_overtake1	Calling MPI_Issend() / MPI_Irecv() and making sure it does not return until a matching receive has been posted.
fortran/nonblocking/functional/ MPI_Issend_overtake2	Calling MPI_Issend() / MPI_Irecv() and make sure it makes progress when matched by a MPI_Irecv() even though MPI_Wait() is not called until sometime later.
fortran/nonblocking/functional/ MPI_Issend_overtake3	2 ranks calling MPI_Issend() / MPI_Irecv() to each other. They should both make progress when matching MPI_Recv()s have been called.
fortran/nonblocking/functional/ MPI_Issend_rtoa	Calling MPI_Issend() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.
fortran/nonblocking/functional/ MPI_Request_free	Calling MPI_Request_free() on each non-blocking calls. All requests are expected to complete even though they are freed.

Table 42: Nonblocking functional tests

Test path	Description
fortran/nonblocking/functional/MPI_Testall	Calling MPI_Testall() on active requests.
fortran/nonblocking/functional/MPI_Testany	Calling MPI_Testany() on active requests.
fortran/nonblocking/functional/MPI_Testsome	Calling MPI_Testsome() on active requests.
fortran/nonblocking/functional/MPI_Waitall	Calling MPI_Waitall() on active requests.
fortran/nonblocking/functional/MPI_Waitany	Calling MPI_Waitany() on active requests.
fortran/nonblocking/functional/MPI_Waitsome	Calling MPI_Waitsome() on active requests.
fortran/nonblocking/functional/async	Stress asynchronous rank-to-rank communication.
fortran/nonblocking/functional/rings	Stress message sending around rings of ranks.

Table 43: Nonblocking error tests

Test path	Description
fortran/nonblocking/error/MPI_Ibsend_err1	Negative destination rank
fortran/nonblocking/error/MPI_Ibsend_err2	Destination rank >= group size
fortran/nonblocking/error/MPI_Ibsend_err3	Invalid communicator (MPI_COMM_NULL)
fortran/nonblocking/error/MPI_Ibsend_err4	Freed communicator
fortran/nonblocking/error/MPI_Ibsend_err5	tag < 0
fortran/nonblocking/error/MPI_Ibsend_err6	tag > MPI_TAG_UB

Table 43: Nonblocking error tests

Test path	Description
fortran/nonblocking/error/MPI_Ibsend_err7	Negative count
fortran/nonblocking/error/MPI_Ibsend_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/nonblocking/error/MPI_Ibsend_err9	Mismatching type at sender and receiver
fortran/nonblocking/error/MPI_Ibsend_err10	Message size > attached buffer size
fortran/nonblocking/error/MPI_Ibsend_err11	No buffer attached
fortran/nonblocking/error/MPI_Irecv_err1	Negative destination rank
fortran/nonblocking/error/MPI_Irecv_err2	Destination rank >= group size
fortran/nonblocking/error/MPI_Irecv_err3	Invalid communicator (MPI_COMM_NULL)
fortran/nonblocking/error/MPI_Irecv_err4	Freed communicator
fortran/nonblocking/error/MPI_Irecv_err5	tag < 0
fortran/nonblocking/error/MPI_Irecv_err6	tag > MPI_TAG_UB
fortran/nonblocking/error/MPI_Irecv_err7	Negative count
fortran/nonblocking/error/MPI_Irecv_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/nonblocking/error/MPI_Irecv_err9	Receiving message > Irecv count
fortran/nonblocking/error/MPI_Irsend_err1	Negative destination rank
fortran/nonblocking/error/MPI_Irsend_err2	Destination rank >= group size
fortran/nonblocking/error/MPI_Irsend_err3	Invalid communicator (MPI_COMM_NULL)
fortran/nonblocking/error/MPI_Irsend_err4	Freed communicator
fortran/nonblocking/error/MPI_Irsend_err5	tag < 0

Table 43: Nonblocking error tests

Test path	Description
fortran/nonblocking/error/ MPI_Irsend_err6	tag > MPI_TAG_UB
fortran/nonblocking/error/ MPI_Irsend_err7	Negative count
fortran/nonblocking/error/ MPI_Irsend_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/nonblocking/error/ MPI_Irsend_err9	Mismatching type at sender and receiver
fortran/nonblocking/error/ MPI_Isend_err1	Negative destination rank
fortran/nonblocking/error/ MPI_Isend_err2	Destination rank >= group size
fortran/nonblocking/error/ MPI_Isend_err3	Invalid communicator (MPI_COMM_NULL)
fortran/nonblocking/error/ MPI_Isend_err4	Freed communicator
fortran/nonblocking/error/ MPI_Isend_err5	tag < 0
fortran/nonblocking/error/ MPI_Isend_err6	tag > MPI_TAG_UB
fortran/nonblocking/error/ MPI_Isend_err7	Negative count
fortran/nonblocking/error/ MPI_Isend_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/nonblocking/error/ MPI_Isend_err9	Mismatching type at sender and receiver
fortran/nonblocking/error/ MPI_Issend_err1	Negative destination rank
fortran/nonblocking/error/ MPI_Issend_err2	Destination rank >= group size
fortran/nonblocking/error/ MPI_Issend_err3	Invalid communicator (MPI_COMM_NULL)

Table 43: Nonblocking error tests

Test path	Description
fortran/nonblocking/error/ MPI_Issend_err4	Freed communicator
fortran/nonblocking/error/ MPI_Issend_err5	tag < 0
fortran/nonblocking/error/ MPI_Issend_err6	tag > MPI_TAG_UB
fortran/nonblocking/error/ MPI_Issend_err7	Negative count
fortran/nonblocking/error/ MPI_Issend_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/nonblocking/error/ MPI_Issend_err9	Mismatching type at sender and receiver

B.2.8 Persistent Operations

Table 44: Persistent functional tests

Test path	Description
fortran/persist_request/functional/ MPI_Bsend_init_null	Calling MPI_Bsend_init() with destination of MPI_PROC_NULL. Test loops through various communicators, message types, and message lengths.
fortran/persist_request/functional/ MPI_Bsend_init_overtake	Calling MPI_Bsend_init() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
fortran/persist_request/functional/ MPI_Bsend_init_rtoa	Calling MPI_Bsend_init() and MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 44: Persistent functional tests

Test path	Description
fortran/persist_request/functional/ MPI_Recv_init_comm	Calling MPI_Recv_init() and MPI_Bsend_init() with various communicators created and make sure the messages are received in the order sent for each communicator (not the order the recv/s were posted).
fortran/persist_request/functional/ MPI_Recv_init_null	Calling MPI_Recv_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
fortran/persist_request/functional/ MPI_Recv_init_pack	Calling MPI_Recv_init() / MPI_Send() verifying that every datatype can be sent as MPI_PACKED. Test loops through various communicators, message types, and message lengths.
fortran/persist_request/functional/ MPI_Request_free_p	Calling MPI_Request_free() using request generated from all persistent calls making sure pending request can still be completed even though it has been freed.
fortran/persist_request/functional/ MPI_Rsend_init_null	Calling MPI_Rsend_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
fortran/persist_request/functional/ MPI_Rsend_init_rtoa	Calling MPI_Rsend_init() / MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root's ranks.

Table 44: Persistent functional tests

Test path	Description
fortran/persist_request/functional/ MPI_Send_init_null	Calling MPI_Send_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
fortran/persist_request/functional/ MPI_Send_init_off	Calling MPI_Send_init() / MPI_Recv_init() to send message from all ranks in communicators to a selected root with byte offsets. Test loops through various communicators, message datatypes, message lengths and root’ ranks.
fortran/persist_request/functional/ MPI_Send_init_overtake	Calling MPI_Send_init() / MPI_Recv_init() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
fortran/persist_request/functional/ MPI_Send_init_rtoa	Calling MPI_Send_init() / MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.
fortran/persist_request/functional/ MPI_Send_init_self	Calling MPI_Send_init() / MPI_Recv_init() and MPI_Recv_init() to send message from a rank to itself. Test loops through various communicators.
fortran/persist_request/functional/ MPI_Ssend_init_ator	Calling MPI_Ssend() / MPI_Recv_init() to send message from all other ranks to the root’s rank in communicators. Test loops through various communicators, message datatypes, message lengths and root’s ranks.

Table 44: Persistent functional tests

Test path	Description
fortran/persist_request/functional/ MPI_Ssend_init_null	Calling MPI_Ssend_init() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
fortran/persist_request/functional/ MPI_Ssend_init_overtake	Calling MPI_Ssend_init() / MPI_Recv_init() and MPI_Recv_init() to send a sequence of messages and make sure they are received in the order they were sent.
fortran/persist_request/functional/ MPI_Ssend_init_rtoa	Calling MPI_Ssend_init() / MPI_Recv_init() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, and message lengths.
fortran/persist_request/functional/ MPI_Startall1	Calling MPI_Bsend_init()/ MPI_Startall() in various communicators.
fortran/persist_request/functional/ MPI_Startall2	Calling MPI_Send_init()/ MPI_Startall() in various communicators.
fortran/persist_request/functional/ MPI_Test_p	Calling MPI_Test() on active requests.
fortran/persist_request/functional/ MPI_Testall_p	Calling MPI_Testall() on active requests.
fortran/persist_request/functional/ MPI_Testany_p	Calling MPI_Testany() on active requests.
fortran/persist_request/functional/ MPI_Testsome_p	Calling MPI_Testsome() on active requests.
fortran/persist_request/functional/ MPI_Waitall_p	Calling MPI_Waitall() on active requests.
fortran/persist_request/functional/ MPI_Waitany_p	Calling MPI_Waitany() on active requests.

Table 44: Persistent functional tests

Test path	Description
fortran/persist_request/functional/ MPI_Waitsome_p	Calling MPI_Waitsome() on active requests.

Table 45: Persistent error tests

Test path	Description
fortran/persist_request/error/ MPI_Bsend_init_err1	Negative destination rank
fortran/persist_request/error/ MPI_Bsend_init_err2	Destination rank >= group size
fortran/persist_request/error/ MPI_Bsend_init_err3	Invalid communicator (MPI_COMM_NULL)
fortran/persist_request/error/ MPI_Bsend_init_err4	Freed communicator
fortran/persist_request/error/ MPI_Bsend_init_err5	tag < 0
fortran/persist_request/error/ MPI_Bsend_init_err6	tag > MPI_TAG_UB
fortran/persist_request/error/ MPI_Bsend_init_err7	Negative count
fortran/persist_request/error/ MPI_Bsend_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/persist_request/error/ MPI_Bsend_init_err9	Mismatching type at sender and receiver
fortran/persist_request/error/ MPI_Bsend_init_err10	Message size > attached buffer size
fortran/persist_request/error/ MPI_Bsend_init_err11	No buffer attached
fortran/persist_request/error/ MPI_Recv_init_err1	Negative destination rank
fortran/persist_request/error/ MPI_Recv_init_err2	Destination rank >= group size

Table 45: Persistent error tests

Test path	Description
fortran/persist_request/error/ MPI_Recv_init_err3	Invalid communicator (MPI_COMM_NULL)
fortran/persist_request/error/ MPI_Recv_init_err4	Freed communicator
fortran/persist_request/error/ MPI_Recv_init_err5	tag < 0
fortran/persist_request/error/ MPI_Recv_init_err6	tag > MPI_TAG_UB
fortran/persist_request/error/ MPI_Recv_init_err7	Negative count
fortran/persist_request/error/ MPI_Recv_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/persist_request/error/ MPI_Recv_init_err9	Message size > Recv_init count
fortran/persist_request/error/ MPI_Request_free_err1	Freed request
fortran/persist_request/error/ MPI_Rsend_init_err1	Negative destination rank
fortran/persist_request/error/ MPI_Rsend_init_err2	Destination rank >= group size
fortran/persist_request/error/ MPI_Rsend_init_err3	Invalid communicator (MPI_COMM_NULL)
fortran/persist_request/error/ MPI_Rsend_init_err4	Freed communicator
fortran/persist_request/error/ MPI_Rsend_init_err5	tag < 0
fortran/persist_request/error/ MPI_Rsend_init_err6	tag > MPI_TAG_UB
fortran/persist_request/error/ MPI_Rsend_init_err7	Negative count
fortran/persist_request/error/ MPI_Rsend_init_err8	Invalid datatype (MPI_DATATYPE_NULL)

Table 45: Persistent error tests

Test path	Description
fortran/persist_request/error/ MPI_Rsend_init_err9	Mismatching type at sender and receiver
fortran/persist_request/error/ MPI_Send_init_err1	Negative destination rank
fortran/persist_request/error/ MPI_Send_init_err2	Destination rank >= group size
fortran/persist_request/error/ MPI_Send_init_err3	Invalid communicator (MPI_COMM_NULL)
fortran/persist_request/error/ MPI_Send_init_err4	Freed communicator
fortran/persist_request/error/ MPI_Send_init_err5	tag < 0
fortran/persist_request/error/ MPI_Send_init_err6	tag > MPI_TAG_UB
fortran/persist_request/error/ MPI_Send_init_err7	Negative count
fortran/persist_request/error/ MPI_Send_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/persist_request/error/ MPI_Send_init_err9	Mismatching type at sender and receiver
fortran/persist_request/error/ MPI_Ssend_init_err1	Negative destination rank
fortran/persist_request/error/ MPI_Ssend_init_err2	Destination rank >= group size
fortran/persist_request/error/ MPI_Ssend_init_err3	Invalid communicator (MPI_COMM_NULL)
fortran/persist_request/error/ MPI_Ssend_init_err4	Freed communicator
fortran/persist_request/error/ MPI_Ssend_init_err5	tag < 0
fortran/persist_request/error/ MPI_Ssend_init_err6	tag > MPI_TAG_UB

Table 45: Persistent error tests

Test path	Description
fortran/persist_request/error/ MPI_Ssend_init_err7	Negative count
fortran/persist_request/error/ MPI_Ssend_init_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/persist_request/error/ MPI_Ssend_init_err9	Mismatching type at sender and receiver
fortran/persist_request/error/ MPI_Start_err1	Already active request
fortran/persist_request/error/ MPI_Start_err2	Freed request
fortran/persist_request/error/ MPI_Startall_err1	Already active request
fortran/persist_request/error/ MPI_Startall_err2	Freed request

B.2.9 Probe and Cancel

Table 46: Probe and Cancel functional tests

Test path	Description
fortran/probe_cancel/functional/ MPI_Cancel_irecv	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active irecv request(s). Another irecv request will be posted afterward with a matching send request. Looping through various communicators, message lengths, and root's ranks.
fortran/probe_cancel/functional/ MPI_Cancel_isend	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active isend request(s). Another isend request will be posted afterward with a matching recv request. Looping through various communicators, message lengths, and root's ranks.

Table 46: Probe and Cancel functional tests

Test path	Description
fortran/probe_cancel/functional/ MPI_Cancel_issend	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active issend request(s). Another issend request will be posted afterward with a matching recv request. Looping through various communicators, message lengths, and root's ranks.
fortran/probe_cancel/functional/ MPI_Cancel_persist_recv	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active persistent receive request(s). Another persistent receive request will be posted afterward with a matching send request. Looping through various communicators, message lengths, and root's ranks.
fortran/probe_cancel/functional/ MPI_Cancel_persist_send	Calling MPI_Cancel() followed by MPI_Test_cancelled() on active persistent send request(s). Another persistent send request will be posted afterward with a matching recv request. Looping through various communicators, message lengths, and root's ranks.
fortran/probe_cancel/functional/ MPI_Cancel_some	Calling MPI_Cancel() and MPI_Test_cancelled() on subset of active isend request(s) and letting other active isend request(s) completed (MPI_Wait()).Looping through various communicators, message lengths, and root's ranks.
fortran/probe_cancel/functional/ MPI_Iprobe_return	Calling MPI_Iprobe() without any message pending and make sure the call returns. Looping through various communicators, and receiver's rank.

Table 46: Probe and Cancel functional tests

Test path	Description
fortran/probe_cancel/functional/ MPI_Iprobe_source	Calling MPI_Iprobe() with message sent from various source ranks. Looping through various communicators, message lengths, and receiver's ranks.
fortran/probe_cancel/functional/ MPI_Iprobe_tag	Calling MPI_Iprobe() with message sent with various tags. Looping through various communicators, message lengths, and sender's ranks.
fortran/probe_cancel/functional/ MPI_Probe_source	Calling MPI_Probe() with message sent from various source ranks. Looping through various communicators, message lengths, and receiver's ranks.
fortran/probe_cancel/functional/ MPI_Probe_tag	Calling MPI_Probe() with message sent with various tags. Looping through various communicators, message lengths, and sender's ranks.
fortran/probe_cancel/functional/ MPI_Test_cancelled_false	Calling MPI_Test_cancelled() with active isend request that has not been cancelled. Looping through various communicators, message lengths, and sender's ranks.

Table 47: Probe and Cancel error tests

Test path	Description
fortran/probe_cancel/error/ MPI_Cancel_err1	Inactive request
fortran/probe_cancel/error/ MPI_Cancel_err2	Freed request
fortran/probe_cancel/error/ MPI_Iprobe_err1	Negative source rank

Table 47: Probe and Cancel error tests

Test path	Description
fortran/probe_cancel/error/ MPI_Iprobe_err2	Negative tag
fortran/probe_cancel/error/ MPI_Iprobe_err3	tag > MPI_TAG_UB
fortran/probe_cancel/error/ MPI_Iprobe_err4	Invalid communicator (MPI_COMM_NULL)
fortran/probe_cancel/error/ MPI_Iprobe_err5	Invalid rank (source rank = comm_size)
fortran/probe_cancel/error/ MPI_Probe_err1	Negative source rank
fortran/probe_cancel/error/ MPI_Probe_err2	Negative tag
fortran/probe_cancel/error/ MPI_Probe_err3	tag > MPI_TAG_UB
fortran/probe_cancel/error/ MPI_Probe_err4	Invalid communicator (MPI_COMM_NULL)
fortran/probe_cancel/error/ MPI_Probe_err5	Invalid rank (source rank = comm_size)

B.2.10 Send and Recv

Table 48: Send and Recv functional tests

Test path	Description
fortran/sendrecv/functional/ MPI_Sendrecv_null	Calling MPI_Sendrecv() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.
fortran/sendrecv/functional/ MPI_Sendrecv_replace_null	Calling MPI_Sendrecv_replace() using MPI_PROC_NULL. Test loops through various communicators, message datatypes, and message lengths.

Table 48: Send and Recv functional tests

Test path	Description
fortran/sendrecv/functional/ MPI_Sendrecv_replace_ring	Calling MPI_Sendrecv_replace() to send and recv message in a ring style. Test loops through various communicators, message datatypes, and message lengths.
fortran/sendrecv/functional/ MPI_Sendrecv_replace_rtoa	Calling MPI_Sendrecv_replace() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, and message lengths.
fortran/sendrecv/functional/ MPI_Sendrecv_replace_self	Calling MPI_Sendrecv_replace() to send message from each rank to itself. Test loops through various communicators, message datatypes, and message lengths.
fortran/sendrecv/functional/ MPI_Sendrecv_ring	Calling MPI_Sendrecv() to send and recv message in a ring style. Test loops through various communicators, message datatypes, and message lengths.
fortran/sendrecv/functional/ MPI_Sendrecv_rtoa	Calling MPI_Sendrecv() to send message from root to all other ranks in communicators. Test loops through various communicators, message datatypes, and message lengths.
fortran/sendrecv/functional/ MPI_Sendrecv_self	Calling MPI_Sendrecv() to send message from each rank to itself. Test loops through various communicators, message datatypes, and message lengths.

Table 49: Send and Recv error tests

Test path	Description
fortran/sendrecv/error/ MPI_Sendrecv_err1a	Negative Destination rank
fortran/sendrecv/error/ MPI_Sendrecv_err1b	Negative Source rank
fortran/sendrecv/error/ MPI_Sendrecv_err2a	Destination rank >= group size
fortran/sendrecv/error/ MPI_Sendrecv_err2b	Source rank >= group size
fortran/sendrecv/error/ MPI_Sendrecv_err3	Invalid communicator (MPI_COMM_NULL)
fortran/sendrecv/error/ MPI_Sendrecv_err4	Freed communicator
fortran/sendrecv/error/ MPI_Sendrecv_err5a	send tag < 0
fortran/sendrecv/error/ MPI_Sendrecv_err5b	receive tag < 0
fortran/sendrecv/error/ MPI_Sendrecv_err6a	send tag > MPI_TAG_UB
fortran/sendrecv/error/ MPI_Sendrecv_err6b	receive tag > MPI_TAG_UB
fortran/sendrecv/error/ MPI_Sendrecv_err7a	Negative send count
fortran/sendrecv/error/ MPI_Sendrecv_err7b	Negative receive count
fortran/sendrecv/error/ MPI_Sendrecv_err8a	Invalid send type (MPI_DATATYPE_NULL)
fortran/sendrecv/error/ MPI_Sendrecv_err8b	Invalid receive type (MPI_DATATYPE_NULL)
fortran/sendrecv/error/ MPI_Sendrecv_err9	Mismatching type at sender and receiver
fortran/sendrecv/error/ MPI_Sendrecv_replace_err1a	Negative Destination rank

Table 49: Send and Recv error tests

Test path	Description
fortran/sendrecv/error/ MPI_Sendrecv_replace_err1b	Negative Source rank
fortran/sendrecv/error/ MPI_Sendrecv_replace_err2a	Destination rank >= group size
fortran/sendrecv/error/ MPI_Sendrecv_replace_err2b	Source rank >= group size
fortran/sendrecv/error/ MPI_Sendrecv_replace_err3	Invalid communicator (MPI_COMM_NULL)
fortran/sendrecv/error/ MPI_Sendrecv_replace_err4	Freed communicator
fortran/sendrecv/error/ MPI_Sendrecv_replace_err5a	send tag < 0
fortran/sendrecv/error/ MPI_Sendrecv_replace_err5b	receive tag < 0
fortran/sendrecv/error/ MPI_Sendrecv_replace_err6a	send tag > MPI_TAG_UB
fortran/sendrecv/error/ MPI_Sendrecv_replace_err6b	receive tag > MPI_TAG_UB
fortran/sendrecv/error/ MPI_Sendrecv_replace_err7	Negative count
fortran/sendrecv/error/ MPI_Sendrecv_replace_err8	Invalid datatype (MPI_DATATYPE_NULL)
fortran/sendrecv/error/ MPI_Sendrecv_replace_err9	Mismatching type at sender and receiver

B.2.11 Topology

Table 50: Topology functional tests

Test path	Description
fortran/topo/functional/MPI_Cart_coords	Calling MPI_Cart_coords for various cartesian communicators created.

Table 50: Topology functional tests

Test path	Description
fortran/topo/functional/ MPI_Cart_create_nonperiodic	Using MPI_Cart_create() to create various non-periodic cartesian communicators
fortran/topo/functional/ MPI_Cart_create_periodic	Using MPI_Cart_create() to create various periodic cartesian communicators. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Cart_create_undef	Calling MPI_Cart_create() with some ranks returning MPI_UNDEFINED. Looping through various intracommunicators.
fortran/topo/functional/MPI_Cart_get	Calling MPI_Cart_get() with various cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/MPI_Cart_map	Calling MPI_Cart_map() with various cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/MPI_Cart_rank	Calling MPI_Cart_rank() with various cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Cart_shift_nonperiodic	Calling MPI_Cart_shift() with various non-periodic cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Cart_shift_periodic	Calling MPI_Cart_shift() with various periodic cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/MPI_Cart_sub	Calling MPI_Cart_sub() with various cartesian communicators created. Looping through various intracommunicators.

Table 50: Topology functional tests

Test path	Description
fortran/topo/functional/MPI_Cartdim_get	Calling MPI_Cartdim_get() with various cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Dims_create_system	Calling MPI_Dims_create() with all output dimensions set up by MPI. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Dims_create_user	Calling MPI_Dims_create() with some output dimensions set up by MPI and some are set by user. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Graph_create_noreorder	Calling MPI_Graph_create() with reorder = FALSE. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Graph_create_reorder	Calling MPI_Graph_create() with reorder = TRUE. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Graph_create_undef	Calling MPI_Graph_create() such that some ranks will return MPI_UNDEFINED. Looping through various intracommunicators.
fortran/topo/functional/MPI_Graph_get	Calling MPI_Graph_get() with various graph communicators created. Looping through various intracommunicators.
fortran/topo/functional/MPI_Graph_map	Calling MPI_Graph_map() with various graph communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Graph_neighbors	Calling MPI_Graph_neighbors() with various graph communicators created. Looping through various intracommunicators.

Table 50: Topology functional tests

Test path	Description
fortran/topo/functional/ MPI_Graph_neighbors_count	Calling MPI_Graph_count() with various graph communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Graphdims_get	Calling MPI_Graphdims_get() with various graph communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Topo_test_cart	Calling MPI_Topo_test() with various cartesian communicators created. Looping through various intracommunicators.
fortran/topo/functional/ MPI_Topo_test_graph	Calling MPI_Topo_test() with various graph communicators created. Looping through various intracommunicators.

Table 51: Topology error tests

Test path	Description
fortran/topo/error/MPI_Cart_coords_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Cart_coords_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Cart_coords_err3	Invalid communicator (communicator with graph topology)
fortran/topo/error/MPI_Cart_create_err1	Fewer processes than dimensions
fortran/topo/error/MPI_Cart_create_err2	Dimensions = 0
fortran/topo/error/MPI_Cart_create_err3	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Cart_get_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Cart_get_err2	Invalid communicator (MPI_COMM_NULL)

Table 51: Topology error tests

Test path	Description
fortran/topo/error/MPI_Cart_get_err3	Invalid communicator (communicator with graph topology)
fortran/topo/error/MPI_Cart_get_err4	Negative dimension
fortran/topo/error/MPI_Cart_map_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Cart_map_err3	dimensions = 0
fortran/topo/error/MPI_Cart_map_err5	More ranks than available
fortran/topo/error/MPI_Cart_rank_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Cart_rank_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Cart_rank_err3	Invalid communicator (communicator with graph topology)
fortran/topo/error/MPI_Cart_shift_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Cart_shift_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Cart_shift_err3	Invalid communicator (communicator with graph topology)
fortran/topo/error/MPI_Cart_shift_err4	displacement = 0
fortran/topo/error/MPI_Cart_shift_err5	Invalid shift direction value
fortran/topo/error/MPI_Cart_sub_err2	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Cart_sub_err3	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Cart_sub_err4	Invalid communicator (communicator with graph topology)
fortran/topo/error/MPI_Cartdim_get_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Cartdim_get_err2	Invalid communicator (MPI_COMM_NULL)

Table 51: Topology error tests

Test path	Description
fortran/topo/error/MPI_Cartdim_get_err3	Invalid communicator (communicator with graph topology)
fortran/topo/error/MPI_Dims_create_err1	nodes not a multiple of non-zero dimensions
fortran/topo/error/MPI_Dims_create_err2	Negative dimension
fortran/topo/error/MPI_Dims_create_err3	Negative nodes
fortran/topo/error/MPI_Dims_create_err4	Number of dimensions < 0
fortran/topo/error/MPI_Graph_create_err1	Negative nodes
fortran/topo/error/MPI_Graph_create_err2	nodes > size of MPI_COMM_WORLD
fortran/topo/error/MPI_Graph_create_err3	Edge number too large
fortran/topo/error/MPI_Graph_create_err4	Negative edge number
fortran/topo/error/MPI_Graph_create_err5	Edge to itself
fortran/topo/error/MPI_Graph_create_err6	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Graph_get_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/MPI_Graph_get_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Graph_get_err3	Invalid communicator (communicator with cartesian topology)
fortran/topo/error/MPI_Graph_map_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/MPI_Graph_map_err3	Negative nnodes
fortran/topo/error/MPI_Graph_map_err4	More ranks than available
fortran/topo/error/MPI_Graph_neighbors_count_err1	Invalid communicator (MPI_COMM_WORLD)

Table 51: Topology error tests

Test path	Description
fortran/topo/error/ MPI_Graph_neighbors_count_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/ MPI_Graph_neighbors_count_err3	Invalid communicator (communicator with cartesian topology)
fortran/topo/error/ MPI_Graph_neighbors_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/ MPI_Graph_neighbors_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/ MPI_Graph_neighbors_err3	Invalid communicator (communicator with cartesian topology)
fortran/topo/error/ MPI_Graphdims_get_err1	Invalid communicator (MPI_COMM_WORLD)
fortran/topo/error/ MPI_Graphdims_get_err2	Invalid communicator (MPI_COMM_NULL)
fortran/topo/error/ MPI_Graphdims_get_err3	Invalid communicator (communicator with cartesian topology)
fortran/topo/error/MPI_Topo_test_err1	Invalid communicator (MPI_COMM_NULL)