MPI: A Message-Passing Interface Standard

Version 2.2

Message Passing Interface Forum

August 31, 2009

This document describes the Message-Passing Interface (MPI) standard, version 2.2. The MPI standard includes point-to-point message-passing, collective communications, group and communicator concepts, process topologies, environmental management, process creation and management, one-sided communications, extended collective operations, external interfaces, I/O, some miscellaneous topics, and a profiling interface. Language bindings for C, C++ and Fortran are defined.

Technically, this version of the standard is based on "MPI: A Message-Passing Interface Standard, version 2.1, June 23, 2008. The MPI Forum added three new routines and a number of enhancements and clarifications to the standard.

Historically, the evolution of the standards is from MPI-1.0 (June 1994) to MPI-1.1 (June 12, 1995) to MPI-1.2 (July 18, 1997), with several clarifications and additions and published as part of the MPI-2 document, to MPI-2.0 (July 18, 1997), with new functionality, to MPI-1.3 (May 30, 2008), combining for historical reasons the documents 1.1 and 1.2 and some errata documents to one combined document, and to MPI-2.1 (June 23, 2008), combining the previous documents. This version, MPI-2.2, is based on MPI-2.1 and provides additional clarifications and errata corrections as well as a few enhancements.

Version 2.2: September 4, 2009. This document contains mostly corrections and clarifications to the MPI 2.1 document. A few extensions have been added; however all correct MPI 2.1 programs are correct MPI 2.2 programs. New features were adopted only when there were compelling needs for users, open source implementations, and minor impact on existing MPI implementations.

Version 2.1: June 23, 2008. This document combines the previous documents MPI-1.3 (May 30, 2008) and MPI-2.0 (July 18, 1997). Certain parts of MPI-2.0, such as some sections of Chapter 4, Miscellany, and Chapter 7, Extended Collective Operations have been merged into the Chapters of MPI-1.3. Additional errata and clarifications collected by the MPI Forum are also included in this document.

Version 1.3: May 30, 2008. This document combines the previous documents MPI-1.1 (June 12, 1995) and the MPI-1.2 Chapter in MPI-2 (July 18, 1997). Additional errata collected by the MPI Forum referring to MPI-1.1 and MPI-1.2 are also included in this document.

Version 2.0: July 18, 1997. Beginning after the release of MPI-1.1, the MPI Forum began meeting to consider corrections and extensions. MPI-2 has been focused on process creation and management, one-sided communications, extended collective communications, external interfaces and parallel I/O. A miscellany chapter discusses items that don't fit elsewhere, in particular language interoperability.

Version 1.2: July 18, 1997. The MPI-2 Forum introduced MPI-1.2 as Chapter 3 in the standard "MPI-2: Extensions to the Message-Passing Interface", July 18, 1997. This section contains clarifications and minor corrections to Version 1.1 of the MPI Standard. The only new function in MPI-1.2 is one for identifying to which version of the MPI Standard the implementation conforms. There are small differences between MPI-1 and MPI-1.1. There are very few differences between MPI-1.1 and MPI-1.2, but large differences between MPI-1.2 and MPI-2.

Version 1.1: June, 1995. Beginning in March, 1995, the Message-Passing Interface Forum reconvened to correct errors and make clarifications in the MPI document of May 5, 1994, referred to below as Version 1.0. These discussions resulted in Version 1.1, which is this document. The changes from Version 1.0 are minor. A version of this document with all changes marked is available. This paragraph is an example of a change.

Version 1.0: May, 1994. The Message-Passing Interface Forum (MPIF), with participation from over 40 organizations, has been meeting since January 1993 to discuss and define a set of library interface standards for message passing. MPIF is not sanctioned or supported by any official standards organization.

The goal of the Message-Passing Interface, simply stated, is to develop a widely used standard for writing message-passing programs. As such the interface should establish a practical, portable, efficient, and flexible standard for message-passing.

This is the final report, Version 1.0, of the Message-Passing Interface Forum. This document contains all the technical features proposed for the interface. This copy of the draft was processed by LaTeX on May 5, 1994.

Please send comments on MPI to `mpi-comments@mpi-forum.org`. Your comment will be forwarded to MPI Forum committee members who will attempt to respond.

Contents

Acknowledgments viii

1 Introduction to MPI 1
- 1.1 Overview and Goals . . . 1
- 1.2 Background of MPI-1.0 . . . 2
- 1.3 Background of MPI-1.1, MPI-1.2, and MPI-2.0 . . . 3
- 1.4 Background of MPI-1.3 and MPI-2.1 . . . 3
- 1.5 Background of MPI-2.2 . . . 4
- 1.6 Who Should Use This Standard? . . . 4
- 1.7 What Platforms Are Targets For Implementation? . . . 4
- 1.8 What Is Included In The Standard? . . . 5
- 1.9 What Is Not Included In The Standard? . . . 6
- 1.10 Organization of this Document . . . 6

2 MPI Terms and Conventions 9
- 2.1 Document Notation . . . 9
- 2.2 Naming Conventions . . . 9
- 2.3 Procedure Specification . . . 10
- 2.4 Semantic Terms . . . 11
- 2.5 Data Types . . . 12
  - 2.5.1 Opaque Objects . . . 12
  - 2.5.2 Array Arguments . . . 14
  - 2.5.3 State . . . 14
  - 2.5.4 Named Constants . . . 14
  - 2.5.5 Choice . . . 15
  - 2.5.6 Addresses . . . 15
  - 2.5.7 File Offsets . . . 16
- 2.6 Language Binding . . . 16
  - 2.6.1 Deprecated Names and Functions . . . 16
  - 2.6.2 Fortran Binding Issues . . . 17
  - 2.6.3 C Binding Issues . . . 18
  - 2.6.4 C++ Binding Issues . . . 18
  - 2.6.5 Functions and Macros . . . 21
- 2.7 Processes . . . 22
- 2.8 Error Handling . . . 22
- 2.9 Implementation Issues . . . 23
  - 2.9.1 Independence of Basic Runtime Routines . . . 23

2.9.2 Interaction with Signals . . . 24
2.10 Examples . . . 24

3 Point-to-Point Communication 25
3.1 Introduction . . . 25
3.2 Blocking Send and Receive Operations . . . 26
3.2.1 Blocking Send . . . 26
3.2.2 Message Data . . . 27
3.2.3 Message Envelope . . . 29
3.2.4 Blocking Receive . . . 30
3.2.5 Return Status . . . 31
3.2.6 Passing MPI_STATUS_IGNORE for Status . . . 33
3.3 Data Type Matching and Data Conversion . . . 34
3.3.1 Type Matching Rules . . . 34
Type MPI_CHARACTER . . . 36
3.3.2 Data Conversion . . . 37
3.4 Communication Modes . . . 38
3.5 Semantics of Point-to-Point Communication . . . 42
3.6 Buffer Allocation and Usage . . . 45
3.6.1 Model Implementation of Buffered Mode . . . 47
3.7 Nonblocking Communication . . . 48
3.7.1 Communication Request Objects . . . 49
3.7.2 Communication Initiation . . . 50
3.7.3 Communication Completion . . . 53
3.7.4 Semantics of Nonblocking Communications . . . 56
3.7.5 Multiple Completions . . . 57
3.7.6 Non-destructive Test of status . . . 64
3.8 Probe and Cancel . . . 64
3.9 Persistent Communication Requests . . . 69
3.10 Send-Receive . . . 73
3.11 Null Processes . . . 75

4 Datatypes 77
4.1 Derived Datatypes . . . 77
4.1.1 Type Constructors with Explicit Addresses . . . 79
4.1.2 Datatype Constructors . . . 79
4.1.3 Subarray Datatype Constructor . . . 87
4.1.4 Distributed Array Datatype Constructor . . . 89
4.1.5 Address and Size Functions . . . 94
4.1.6 Lower-Bound and Upper-Bound Markers . . . 96
4.1.7 Extent and Bounds of Datatypes . . . 97
4.1.8 True Extent of Datatypes . . . 98
4.1.9 Commit and Free . . . 99
4.1.10 Duplicating a Datatype . . . 100
4.1.11 Use of General Datatypes in Communication . . . 101
4.1.12 Correct Use of Addresses . . . 104
4.1.13 Decoding a Datatype . . . 104
4.1.14 Examples . . . 112

4.2 Pack and Unpack . . . 121
4.3 Canonical MPI_PACK and MPI_UNPACK . . . 127

5 Collective Communication 131
5.1 Introduction and Overview . . . 131
5.2 Communicator Argument . . . 134
5.2.1 Specifics for Intracommunicator Collective Operations . . . 134
5.2.2 Applying Collective Operations to Intercommunicators . . . 134
5.2.3 Specifics for Intercommunicator Collective Operations . . . 137
5.3 Barrier Synchronization . . . 137
5.4 Broadcast . . . 138
5.4.1 Example using MPI_BCAST . . . 138
5.5 Gather . . . 139
5.5.1 Examples using MPI_GATHER, MPI_GATHERV . . . 142
5.6 Scatter . . . 149
5.6.1 Examples using MPI_SCATTER, MPI_SCATTERV . . . 152
5.7 Gather-to-all . . . 154
5.7.1 Exampleusing MPI_ALLGATHER . . . 156
5.8 All-to-All Scatter/Gather . . . 157
5.9 Global Reduction Operations . . . 162
5.9.1 Reduce . . . 163
5.9.2 Predefined Reduction Operations . . . 164
5.9.3 Signed Characters and Reductions . . . 167
5.9.4 MINLOC and MAXLOC . . . 167
5.9.5 User-Defined Reduction Operations . . . 171
Example of User-defined Reduce . . . 174
5.9.6 All-Reduce . . . 175
5.9.7 Process-local reduction . . . 176
5.10 Reduce-Scatter . . . 177
5.10.1 MPI_REDUCE_SCATTER_BLOCK . . . 178
5.10.2 MPI_REDUCE_SCATTER . . . 179
5.11 Scan . . . 180
5.11.1 Inclusive Scan . . . 180
5.11.2 Exclusive Scan . . . 181
5.11.3 Example using MPI_SCAN . . . 182
5.12 Correctness . . . 183

6 Groups, Contexts, Communicators, and Caching 187
6.1 Introduction . . . 187
6.1.1 Features Needed to Support Libraries . . . 187
6.1.2 MPI's Support for Libraries . . . 188
6.2 Basic Concepts . . . 190
6.2.1 Groups . . . 190
6.2.2 Contexts . . . 190
6.2.3 Intra-Communicators . . . 191
6.2.4 Predefined Intra-Communicators . . . 191
6.3 Group Management . . . 192
6.3.1 Group Accessors . . . 192

6.3.2 Group Constructors . . . 193
6.3.3 Group Destructors . . . 198
6.4 Communicator Management . . . 199
6.4.1 Communicator Accessors . . . 199
6.4.2 Communicator Constructors . . . 200
6.4.3 Communicator Destructors . . . 208
6.5 Motivating Examples . . . 209
6.5.1 Current Practice #1 . . . 209
6.5.2 Current Practice #2 . . . 210
6.5.3 (Approximate) Current Practice #3 . . . 210
6.5.4 Example #4 . . . 211
6.5.5 Library Example #1 . . . 212
6.5.6 Library Example #2 . . . 213
6.6 Inter-Communication . . . 216
6.6.1 Inter-communicator Accessors . . . 217
6.6.2 Inter-communicator Operations . . . 219
6.6.3 Inter-Communication Examples . . . 221
Example 1: Three-Group "Pipeline" . . . 221
Example 2: Three-Group "Ring" . . . 222
6.7 Caching . . . 224
6.7.1 Functionality . . . 225
6.7.2 Communicators . . . 226
6.7.3 Windows . . . 230
6.7.4 Datatypes . . . 233
6.7.5 Error Class for Invalid Keyval . . . 236
6.7.6 Attributes Example . . . 236
6.8 Naming Objects . . . 238
6.9 Formalizing the Loosely Synchronous Model . . . 242
6.9.1 Basic Statements . . . 242
6.9.2 Models of Execution . . . 242
Static communicator allocation . . . 243
Dynamic communicator allocation . . . 243
The General case . . . 243

7 Process Topologies **245**
7.1 Introduction . . . 245
7.2 Virtual Topologies . . . 246
7.3 Embedding in MPI . . . 246
7.4 Overview of the Functions . . . 246
7.5 Topology Constructors . . . 248
7.5.1 Cartesian Constructor . . . 248
7.5.2 Cartesian Convenience Function: MPI_DIMS_CREATE . . . 248
7.5.3 General (Graph) Constructor . . . 250
7.5.4 Distributed (Graph) Constructor . . . 252
7.5.5 Topology Inquiry Functions . . . 257
7.5.6 Cartesian Shift Coordinates . . . 265
7.5.7 Partitioning of Cartesian structures . . . 266
7.5.8 Low-Level Topology Functions . . . 267

7.6 An Application Example . . . 268

8 MPI Environmental Management 271
8.1 Implementation Information . . . 271
8.1.1 Version Inquiries . . . 271
8.1.2 Environmental Inquiries . . . 272
Tag Values . . . 272
Host Rank . . . 272
IO Rank . . . 273
Clock Synchronization . . . 273
8.2 Memory Allocation . . . 274
8.3 Error Handling . . . 276
8.3.1 Error Handlers for Communicators . . . 278
8.3.2 Error Handlers for Windows . . . 280
8.3.3 Error Handlers for Files . . . 281
8.3.4 Freeing Errorhandlers and Retrieving Error Strings . . . 282
8.4 Error Codes and Classes . . . 283
8.5 Error Classes, Error Codes, and Error Handlers . . . 286
8.6 Timers and Synchronization . . . 289
8.7 Startup . . . 290
8.7.1 Allowing User Functions at Process Termination . . . 295
8.7.2 Determining Whether MPI Has Finished . . . 296
8.8 Portable MPI Process Startup . . . 297

9 The Info Object 299

10 Process Creation and Management 305
10.1 Introduction . . . 305
10.2 The Dynamic Process Model . . . 306
10.2.1 Starting Processes . . . 306
10.2.2 The Runtime Environment . . . 306
10.3 Process Manager Interface . . . 308
10.3.1 Processes in MPI . . . 308
10.3.2 Starting Processes and Establishing Communication . . . 308
10.3.3 Starting Multiple Executables and Establishing Communication . . . 313
10.3.4 Reserved Keys . . . 315
10.3.5 Spawn Example . . . 316
Manager-worker Example, Using MPI_COMM_SPAWN. . . . 316
10.4 Establishing Communication . . . 318
10.4.1 Names, Addresses, Ports, and All That . . . 318
10.4.2 Server Routines . . . 320
10.4.3 Client Routines . . . 322
10.4.4 Name Publishing . . . 323
10.4.5 Reserved Key Values . . . 325
10.4.6 Client/Server Examples . . . 325
Simplest Example — Completely Portable. . . . 325
Ocean/Atmosphere - Relies on Name Publishing . . . 326
Simple Client-Server Example. . . . 326

10.5 Other Functionality . . . 328
10.5.1 Universe Size . . . 328
10.5.2 Singleton MPI_INIT . . . 329
10.5.3 MPI_APPNUM . . . 329
10.5.4 Releasing Connections . . . 330
10.5.5 Another Way to Establish MPI Communication . . . 331

11 One-Sided Communications **335**
11.1 Introduction . . . 335
11.2 Initialization . . . 336
11.2.1 Window Creation . . . 336
11.2.2 Window Attributes . . . 338
11.3 Communication Calls . . . 339
11.3.1 Put . . . 340
11.3.2 Get . . . 342
11.3.3 Examples . . . 342
11.3.4 Accumulate Functions . . . 345
11.4 Synchronization Calls . . . 347
11.4.1 Fence . . . 352
11.4.2 General Active Target Synchronization . . . 353
11.4.3 Lock . . . 357
11.4.4 Assertions . . . 358
11.4.5 Miscellaneous Clarifications . . . 360
11.5 Examples . . . 360
11.6 Error Handling . . . 363
11.6.1 Error Handlers . . . 363
11.6.2 Error Classes . . . 363
11.7 Semantics and Correctness . . . 363
11.7.1 Atomicity . . . 369
11.7.2 Progress . . . 369
11.7.3 Registers and Compiler Optimizations . . . 371

12 External Interfaces **373**
12.1 Introduction . . . 373
12.2 Generalized Requests . . . 373
12.2.1 Examples . . . 377
12.3 Associating Information with Status . . . 379
12.4 MPI and Threads . . . 381
12.4.1 General . . . 381
12.4.2 Clarifications . . . 382
12.4.3 Initialization . . . 384

13 I/O **389**
13.1 Introduction . . . 389
13.1.1 Definitions . . . 389
13.2 File Manipulation . . . 391
13.2.1 Opening a File . . . 391
13.2.2 Closing a File . . . 393

13.2.3 Deleting a File . . . 394
13.2.4 Resizing a File . . . 395
13.2.5 Preallocating Space for a File . . . 395
13.2.6 Querying the Size of a File . . . 396
13.2.7 Querying File Parameters . . . 397
13.2.8 File Info . . . 398
Reserved File Hints . . . 399
13.3 File Views . . . 401
13.4 Data Access . . . 404
13.4.1 Data Access Routines . . . 404
Positioning . . . 405
Synchronism . . . 405
Coordination . . . 405
Data Access Conventions . . . 406
13.4.2 Data Access with Explicit Offsets . . . 407
13.4.3 Data Access with Individual File Pointers . . . 410
13.4.4 Data Access with Shared File Pointers . . . 416
Noncollective Operations . . . 417
Collective Operations . . . 419
Seek . . . 420
13.4.5 Split Collective Data Access Routines . . . 421
13.5 File Interoperability . . . 428
13.5.1 Datatypes for File Interoperability . . . 430
13.5.2 External Data Representation: "external32" . . . 431
13.5.3 User-Defined Data Representations . . . 432
Extent Callback . . . 434
Datarep Conversion Functions . . . 435
13.5.4 Matching Data Representations . . . 437
13.6 Consistency and Semantics . . . 437
13.6.1 File Consistency . . . 437
13.6.2 Random Access vs. Sequential Files . . . 440
13.6.3 Progress . . . 441
13.6.4 Collective File Operations . . . 441
13.6.5 Type Matching . . . 441
13.6.6 Miscellaneous Clarifications . . . 441
13.6.7 MPI_Offset Type . . . 442
13.6.8 Logical vs. Physical File Layout . . . 442
13.6.9 File Size . . . 442
13.6.10 Examples . . . 443
Asynchronous I/O . . . 445
13.7 I/O Error Handling . . . 447
13.8 I/O Error Classes . . . 447
13.9 Examples . . . 448
13.9.1 Double Buffering with Split Collective I/O . . . 448
13.9.2 Subarray Filetype Constructor . . . 450

14 Profiling Interface **453**
14.1 Requirements . . . 453
14.2 Discussion . . . 453
14.3 Logic of the Design . . . 454
14.3.1 Miscellaneous Control of Profiling . . . 454
14.4 Examples . . . 455
14.4.1 Profiler Implementation . . . 455
14.4.2 MPI Library Implementation . . . 456
Systems with Weak Symbols . . . 456
Systems Without Weak Symbols . . . 456
14.4.3 Complications . . . 457
Multiple Counting . . . 457
Linker Oddities . . . 457
14.5 Multiple Levels of Interception . . . 458

15 Deprecated Functions **459**
15.1 Deprecated since MPI-2.0 . . . 459
15.2 Deprecated since MPI-2.2 . . . 465

16 Language Bindings **467**
16.1 C++ . . . 467
16.1.1 Overview . . . 467
16.1.2 Design . . . 467
16.1.3 C++ Classes for MPI . . . 468
16.1.4 Class Member Functions for MPI . . . 468
16.1.5 Semantics . . . 469
16.1.6 C++ Datatypes . . . 471
16.1.7 Communicators . . . 474
16.1.8 Exceptions . . . 476
16.1.9 Mixed-Language Operability . . . 477
16.1.10 Profiling . . . 477
16.2 Fortran Support . . . 480
16.2.1 Overview . . . 480
16.2.2 Problems With Fortran Bindings for MPI . . . 481
Problems Due to Strong Typing . . . 482
Problems Due to Data Copying and Sequence Association . . . 482
Special Constants . . . 484
Fortran 90 Derived Types . . . 484
A Problem with Register Optimization . . . 485
16.2.3 Basic Fortran Support . . . 487
16.2.4 Extended Fortran Support . . . 488
The `mpi` Module . . . 488
No Type Mismatch Problems for Subroutines with Choice Arguments 489
16.2.5 Additional Support for Fortran Numeric Intrinsic Types . . . 489
Parameterized Datatypes with Specified Precision and Exponent Range 490
Support for Size-specific MPI Datatypes . . . 494
Communication With Size-specific Types . . . 496
16.3 Language Interoperability . . . 497

16.3.1 Introduction . . . 497
16.3.2 Assumptions . . . 498
16.3.3 Initialization . . . 498
16.3.4 Transfer of Handles . . . 498
16.3.5 Status . . . 502
16.3.6 MPI Opaque Objects . . . 502
Datatypes . . . 503
Callback Functions . . . 504
Error Handlers . . . 504
Reduce Operations . . . 504
Addresses . . . 504
16.3.7 Attributes . . . 505
16.3.8 Extra State . . . 509
16.3.9 Constants . . . 509
16.3.10 Interlanguage Communication . . . 510

A Language Bindings Summary **512**
A.1 Defined Values and Handles . . . 512
A.1.1 Defined Constants . . . 512
A.1.2 Types . . . 523
A.1.3 Prototype definitions . . . 524
A.1.4 Deprecated prototype definitions . . . 527
A.1.5 Info Keys . . . 528
A.1.6 Info Values . . . 528
A.2 C Bindings . . . 530
A.2.1 Point-to-Point Communication C Bindings . . . 530
A.2.2 Datatypes C Bindings . . . 531
A.2.3 Collective Communication C Bindings . . . 533
A.2.4 Groups, Contexts, Communicators, and Caching C Bindings . . . 534
A.2.5 Process Topologies C Bindings . . . 537
A.2.6 MPI Environmenta Management C Bindings . . . 538
A.2.7 The Info Object C Bindings . . . 539
A.2.8 Process Creation and Management C Bindings . . . 539
A.2.9 One-Sided Communications C Bindings . . . 540
A.2.10 External Interfaces C Bindings . . . 540
A.2.11 I/O C Bindings . . . 541
A.2.12 Language Bindings C Bindings . . . 543
A.2.13 Profiling Interface C Bindings . . . 544
A.2.14 Deprecated C Bindings . . . 544
A.3 Fortran Bindings . . . 546
A.3.1 Point-to-Point Communication Fortran Bindings . . . 546
A.3.2 Datatypes Fortran Bindings . . . 548
A.3.3 Collective Communication Fortran Bindings . . . 551
A.3.4 Groups, Contexts, Communicators, and Caching Fortran Bindings . . . 553
A.3.5 Process Topologies Fortran Bindings . . . 557
A.3.6 MPI Environmenta Management Fortran Bindings . . . 558
A.3.7 The Info Object Fortran Bindings . . . 560
A.3.8 Process Creation and Management Fortran Bindings . . . 561

A.3.9 One-Sided Communications Fortran Bindings . . . 562
A.3.10 External Interfaces Fortran Bindings . . . 563
A.3.11 I/O Fortran Bindings . . . 563
A.3.12 Language Bindings Fortran Bindings . . . 567
A.3.13 Profiling Interface Fortran Bindings . . . 568
A.3.14 Deprecated Fortran Bindings . . . 568
A.4 C++ Bindings (deprecated) . . . 570
A.4.1 Point-to-Point Communication C++ Bindings . . . 570
A.4.2 Datatypes C++ Bindings . . . 573
A.4.3 Collective Communication C++ Bindings . . . 575
A.4.4 Groups, Contexts, Communicators, and Caching C++ Bindings . . . 576
A.4.5 Process Topologies C++ Bindings . . . 579
A.4.6 MPI Environmenta Management C++ Bindings . . . 581
A.4.7 The Info Object C++ Bindings . . . 582
A.4.8 Process Creation and Management C++ Bindings . . . 583
A.4.9 One-Sided Communications C++ Bindings . . . 584
A.4.10 External Interfaces C++ Bindings . . . 585
A.4.11 I/O C++ Bindings . . . 585
A.4.12 Language Bindings C++ Bindings . . . 589
A.4.13 Profiling Interface C++ Bindings . . . 589
A.4.14 C++ Bindings on all MPI Classes . . . 590
A.4.15 Construction / Destruction . . . 590
A.4.16 Copy / Assignment . . . 590
A.4.17 Comparison . . . 590
A.4.18 Inter-language Operability . . . 590

B Change-Log **592**
B.1 Changes from Version 2.1 to Version 2.2 . . . 592
B.2 Changes from Version 2.0 to Version 2.1 . . . 595

Bibliography **599**

Examples Index **603**

MPI Constant and Predefined Handle Index **606**

MPI Declarations Index **612**

MPI Callback Function Prototype Index **614**

MPI Function Index **615**

List of Figures

5.1 Collective comminucations, an overview . . . 132
5.2 Intercommunicator allgather . . . 136
5.3 Intercommunicator reduce-scatter . . . 136
5.4 Gather example . . . 143
5.5 Gatherv example with strides . . . 144
5.6 Gatherv example, 2-dimensional . . . 145
5.7 Gatherv example, 2-dimensional, subarrays with different sizes . . . 146
5.8 Gatherv example, 2-dimensional, subarrays with different sizes and strides . 147
5.9 Scatter example . . . 152
5.10 Scatterv example with strides . . . 153
5.11 Scatterv example with different strides and counts . . . 154
5.12 Race conditions with point-to-point and collective communications . . . 185

6.1 Intercommunicator create using MPI_COMM_CREATE . . . 204
6.2 Intercommunicator constructionwith MPI_COMM_SPLIT . . . 207
6.3 Three-group pipeline. . . . 221
6.4 Three-group ring. . . . 223

7.1 Set-up of process structure for two-dimensional parallel Poisson solver. . . . 270

11.1 Active target communication . . . 349
11.2 Active target communication, with weak synchronization . . . 350
11.3 Passive target communication . . . 351
11.4 Active target communication with several processes . . . 355
11.5 Schematic description of window . . . 364
11.6 Symmetric communication . . . 370
11.7 Deadlock situation . . . 370
11.8 No deadlock . . . 370

13.1 Etypes and filetypes . . . 390
13.2 Partitioning a file among parallel processes . . . 390
13.3 Displacements . . . 402
13.4 Example array file layout . . . 450
13.5 Example local array filetype for process 1 . . . 451

List of Tables

2.1 Deprecated constructs 17

3.1 Predefined MPI datatypes corresponding to Fortran datatypes 27
3.2 Predefined MPI datatypes corresponding to C datatypes 28
3.3 Predefined MPI datatypes corresponding to both C and Fortran datatypes 29

4.1 combiner values returned from MPI_TYPE_GET_ENVELOPE 106

6.1 MPI_COMM_* Function Behavior (in Inter-Communication Mode) 218

8.1 Error classes (Part 1) 284
8.2 Error classes (Part 2) 285

11.1 Error classes in one-sided communication routines 363

13.1 Data access routines 404
13.2 "external32" sizes of predefined datatypes 433
13.3 I/O Error Classes 448

16.1 C++ names for the MPI C and C++ predefined datatypes 472
16.2 C++ names for the MPI Fortran predefined datatypes 472
16.3 C++ names for other MPI datatypes 473

Acknowledgments

This document represents the work of many people who have served on the MPI Forum. The meetings have been attended by dozens of people from many parts of the world. It is the hard and dedicated work of this group that has led to the MPI standard.

The technical development was carried out by subgroups, whose work was reviewed by the full committee. During the period of development of the Message-Passing Interface (MPI), many people helped with this effort.

Those who served as primary coordinators in MPI-1.0 and MPI-1.1 are:

- Jack Dongarra, David Walker, Conveners and Meeting Chairs
- Ewing Lusk, Bob Knighten, Minutes
- Marc Snir, William Gropp, Ewing Lusk, Point-to-Point Communication
- Al Geist, Marc Snir, Steve Otto, Collective Communication
- Steve Otto, Editor
- Rolf Hempel, Process Topologies
- Ewing Lusk, Language Binding
- William Gropp, Environmental Management
- James Cownie, Profiling
- Tony Skjellum, Lyndon Clarke, Marc Snir, Richard Littlefield, Mark Sears, Groups, Contexts, and Communicators
- Steven Huss-Lederman, Initial Implementation Subset

The following list includes some of the active participants in the MPI-1.0 and MPI-1.1 process not mentioned above.

Ed Anderson	Robert Babb	Joe Baron	Eric Barszcz
Scott Berryman	Rob Bjornson	Nathan Doss	Anne Elster
Jim Feeney	Vince Fernando	Sam Fineberg	Jon Flower
Daniel Frye	Ian Glendinning	Adam Greenberg	Robert Harrison
Leslie Hart	Tom Haupt	Don Heller	Tom Henderson
Alex Ho	C.T. Howard Ho	Gary Howell	John Kapenga
James Kohl	Susan Krauss	Bob Leary	Arthur Maccabe
Peter Madams	Alan Mainwaring	Oliver McBryan	Phil McKinley
Charles Mosher	Dan Nessett	Peter Pacheco	Howard Palmer
Paul Pierce	Sanjay Ranka	Peter Rigsbee	Arch Robison
Erich Schikuta	Ambuj Singh	Alan Sussman	Robert Tomlinson
Robert G. Voigt	Dennis Weeks	Stephen Wheat	Steve Zenith

The University of Tennessee and Oak Ridge National Laboratory made the draft available by anonymous FTP mail servers and were instrumental in distributing the document.

The work on the MPI-1 standard was supported in part by ARPA and NSF under grant ASC-9310330, the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615, and by the Commission of the European Community through Esprit project P6643 (PPPE).

MPI-1.2 and MPI-2.0:

Those who served as primary coordinators in MPI-1.2 and MPI-2.0 are:

- Ewing Lusk, Convener and Meeting Chair
- Steve Huss-Lederman, Editor
- Ewing Lusk, Miscellany
- Bill Saphir, Process Creation and Management
- Marc Snir, One-Sided Communications
- Bill Gropp and Anthony Skjellum, Extended Collective Operations
- Steve Huss-Lederman, External Interfaces
- Bill Nitzberg, I/O
- Andrew Lumsdaine, Bill Saphir, and Jeff Squyres, Language Bindings
- Anthony Skjellum and Arkady Kanevsky, Real-Time

The following list includes some of the active participants who attended MPI-2 Forum meetings and are not mentioned above.

Greg Astfalk	Robert Babb	Ed Benson	Rajesh Bordawekar
Pete Bradley	Peter Brennan	Ron Brightwell	Maciej Brodowicz
Eric Brunner	Greg Burns	Margaret Cahir	Pang Chen
Ying Chen	Albert Cheng	Yong Cho	Joel Clark
Lyndon Clarke	Laurie Costello	Dennis Cottel	Jim Cownie
Zhenqian Cui	Suresh Damodaran-Kamal		Raja Daoud
Judith Devaney	David DiNucci	Doug Doefler	Jack Dongarra
Terry Dontje	Nathan Doss	Anne Elster	Mark Fallon
Karl Feind	Sam Fineberg	Craig Fischberg	Stephen Fleischman
Ian Foster	Hubertus Franke	Richard Frost	Al Geist
Robert George	David Greenberg	John Hagedorn	Kei Harada
Leslie Hart	Shane Hebert	Rolf Hempel	Tom Henderson
Alex Ho	Hans-Christian Hoppe	Joefon Jann	Terry Jones
Karl Kesselman	Koichi Konishi	Susan Kraus	Steve Kubica
Steve Landherr	Mario Lauria	Mark Law	Juan Leon
Lloyd Lewins	Ziyang Lu	Bob Madahar	Peter Madams

John May, Oliver McBryan, Brian McCandless, Tyce McLarty, Thom McMahon, Harish Nag, Nick Nevin, Jarek Nieplocha, Ron Oldfield, Peter Ossadnik, Steve Otto, Peter Pacheco, Yoonho Park, Perry Partow, Pratap Pattnaik, Elsie Pierce, Paul Pierce, Heidi Poxon, Jean-Pierre Prost, Boris Protopopov, James Pruyve, Rolf Rabenseifner, Joe Rieken, Peter Rigsbee, Tom Robey, Anna Rounbehler, Nobutoshi Sagawa, Arindam Saha, Eric Salo, Darren Sanders, Eric Sharakan, Andrew Sherman, Fred Shirley, Lance Shuler, A. Gordon Smith, Ian Stockdale, David Taylor, Stephen Taylor, Greg Tensa, Rajeev Thakur, Marydell Tholburn, Dick Treumann, Simon Tsang, Manuel Ujaldon, David Walker, Jerrell Watts, Klaus Wolf, Parkson Wong, Dave Wright

The MPI Forum also acknowledges and appreciates the valuable input from people via e-mail and in person.

The following institutions supported the MPI-2 effort through time and travel support for the people listed above.

Argonne National Laboratory
Bolt, Beranek, and Newman
California Institute of Technology
Center for Computing Sciences
Convex Computer Corporation
Cray Research
Digital Equipment Corporation
Dolphin Interconnect Solutions, Inc.
Edinburgh Parallel Computing Centre
General Electric Company
German National Research Center for Information Technology
Hewlett-Packard
Hitachi
Hughes Aircraft Company
Intel Corporation
International Business Machines
Khoral Research
Lawrence Livermore National Laboratory
Los Alamos National Laboratory
MPI Software Techology, Inc.
Mississippi State University
NEC Corporation
National Aeronautics and Space Administration
National Energy Research Scientific Computing Center
National Institute of Standards and Technology
National Oceanic and Atmospheric Adminstration
Oak Ridge National Laboratory
Ohio State University
PALLAS GmbH

Pacific Northwest National Laboratory
Pratt & Whitney
San Diego Supercomputer Center
Sanders, A Lockheed-Martin Company
Sandia National Laboratories
Schlumberger
Scientific Computing Associates, Inc.
Silicon Graphics Incorporated
Sky Computers
Sun Microsystems Computer Corporation
Syracuse University
The MITRE Corporation
Thinking Machines Corporation
United States Navy
University of Colorado
University of Denver
University of Houston
University of Illinois
University of Maryland
University of Notre Dame
University of San Fransisco
University of Stuttgart Computing Center
University of Wisconsin

MPI-2 operated on a very tight budget (in reality, it had no budget when the first meeting was announced). Many institutions helped the MPI-2 effort by supporting the efforts and travel of the members of the MPI Forum. Direct support was given by NSF and DARPA under NSF contract CDA-9115428 for travel by U.S. academic participants and Esprit under project HPC Standards (21111) for European participants.

MPI-1.3 and MPI-2.1:

The editors and organizers of the combined documents have been:

- Richard Graham, Convener and Meeting Chair
- Jack Dongarra, Steering Committee
- Al Geist, Steering Committee
- Bill Gropp, Steering Committee
- Rainer Keller, Merge of MPI-1.3
- Andrew Lumsdaine, Steering Committee
- Ewing Lusk, Steering Committee, MPI-1.1-Errata (Oct. 12, 1998) MPI-2.1-Errata Ballots 1, 2 (May 15, 2002)

- Rolf Rabenseifner, Steering Committee, Merge of MPI-2.1 and MPI-2.1-Errata Ballots 3, 4 (2008)

All chapters have been revisited to achieve a consistent MPI-2.1 text. Those who served as authors for the necessary modifications are:

- Bill Gropp, Frontmatter, Introduction, and Bibliography
- Richard Graham, Point-to-Point Communication
- Adam Moody, Collective Communication
- Richard Treumann, Groups, Contexts, and Communicators
- Jesper Larsson Träff, Process Topologies, Info-Object, and One-Sided Communications
- George Bosilca, Environmental Management
- David Solt, Process Creation and Management
- Bronis R. de Supinski, External Interfaces, and Profiling
- Rajeev Thakur, I/O
- Jeffrey M. Squyres, Language Bindings and MPI 2.1 Secretary
- Rolf Rabenseifner, Deprecated Functions and Annex Change-Log
- Alexander Supalov and Denis Nagorny, Annex Language Bindings

The following list includes some of the active participants who attended MPI-2 Forum meetings and in the e-mail discussions of the errata items and are not mentioned above.

Pavan Balaji	Purushotham V. Bangalore	Brian Barrett	Richard Barrett
Christian Bell	Robert Blackmore	Gil Bloch	Ron Brightwell
Jeffrey Brown	Darius Buntinas	Jonathan Carter	Nathan DeBardeleben
Terry Dontje	Gabor Dozsa	Edric Ellis	Karl Feind
Edgar Gabriel	Patrick Geoffray	David Gingold	Dave Goodell
Erez Haba	Robert Harrison	Thomas Herault	Steve Hodson
Torsten Hoefler	Joshua Hursey	Yann Kalemkarian	Matthew Koop
Quincey Koziol	Sameer Kumar	Miron Livny	Kannan Narasimhan
Mark Pagel	Avneesh Pant	Steve Poole	Howard Pritchard
Craig Rasmussen	Hubert Ritzdorf	Rob Ross	Tony Skjellum
Brian Smith	Vinod Tipparaju	Jesper Larsson Träff	Keith Underwood

The MPI Forum also acknowledges and appreciates the valuable input from people via e-mail and in person.

The following institutions supported the MPI-2 effort through time and travel support for the people listed above.

Argonne National Laboratory
Bull

Cisco Systems, Inc.
Cray Inc.
The HDF Group
Hewlett-Packard
IBM T.J. Watson Research
Indiana University
Institut National de Recherche en Informatique et Automatique (INRIA)
Intel Corporation
Lawrence Berkeley National Laboratory
Lawrence Livermore National Laboratory
Los Alamos National Laboratory
Mathworks
Mellanox Technologies
Microsoft
Myricom
NEC Laboratories Europe, NEC Europe Ltd.
Oak Ridge National Laboratory
Ohio State University
Pacific Northwest National Laboratory
QLogic Corporation
Sandia National Laboratories
SiCortex
Silicon Graphics Incorporated
Sun Microsystems, Inc.
University of Alabama at Birmingham
University of Houston
University of Illinois at Urbana-Champaign
University of Stuttgart, High Performance Computing Center Stuttgart (HLRS)
University of Tennessee, Knoxville
University of Wisconsin

Funding for the MPI Forum meetings was partially supported by award #CCF-0816909 from the National Science Foundation.
In addition, the HDF Group provided travel support for one U.S. academic.

MPI-2.2:

All chapters have been revisited to achieve a consistent MPI-2.2 text. Those who served as authors for the necessary modifications are:

- William Gropp, Frontmatter, Introduction, and Bibliography; MPI 2.2 chair.
- Richard Graham, Point-to-Point Communication and Datatypes
- Adam Moody and Torsten Hoefler, Collective Communication
- Richard Treumann, Groups, Contexts, and Communicators

- Jesper Larsson Träff and Torsten Hoefler, Process Topologies,
- Jesper Larsson Träff, Info-Object and One-Sided Communications
- George Bosilca, Datatypes, Environmental Management
- David Solt, Process Creation and Management
- Bronis R. de Supinski, External Interfaces, and Profiling
- Rajeev Thakur, I/O
- Jeffrey M. Squyres, Language Bindings and MPI 2.2 Secretary
- Rolf Rabenseifner, Deprecated Functions, and Annex Change-Log
- Alexander Supalov and Rolf Rabenseifner, Annex Language Bindings

The following list includes some of the active participants who attended MPI-2 Forum meetings and in the e-mail discussions of the errata items and are not mentioned above.

Pavan Balaji	Purushotham V. Bangalore	Brian Barrett
Richard Barrett	Christian Bell	Robert Blackmore
Gil Bloch	Ron Brightwell	Greg Bronevetsky
Jeff Brown	Darius Buntinas	Jonathan Carter
Nathan DeBardeleben	Terry Dontje	Gabor Dozsa
Edric Ellis	Karl Feind	Edgar Gabriel
Patrick Geoffray	Johann George	David Gingold
David Goodell	Erez Haba	Robert Harrison
Thomas Herault	Marc-Andra Hermanns	Steve Hodson
Joshua Hursey	Yutaka Ishikawa	Bin Jia
Hideyuki Jitsumoto	Terry Jones	Yann Kalemkarian
Ranier Keller	Matthew Koop	Quincey Koziol
Manojkumar Krishnan	Sameer Kumar	Miron Livny
Andrew Lumsdaine	Miao Luo	Ewing Lusk
Timothy I. Mattox	Kannan Narasimhan	Mark Pagel
Avneesh Pant	Steve Poole	Howard Pritchard
Craig Rasmussen	Hubert Ritzdorf	Rob Ross
Martin Schulz	Pavel Shamis	Galen Shipman
Christian Siebert	Anthony Skjellum	Brian Smith
Naoki Sueyasu	Vinod Tipparaju	Keith Underwood
Rolf Vandevaart	Abhinav Vishnu	Weikuan Yu

The MPI Forum also acknowledges and appreciates the valuable input from people via e-mail and in person.

The following institutions supported the MPI-2.2 effort through time and travel support for the people listed above.

Argonne National Laboratory
Auburn University
Bull
Cisco Systems, Inc.

Cray Inc.
Forschungszentrum Jülich
Fujitsu
The HDF Group
Hewlett-Packard
International Business Machines
Indiana University
Institut National de Recherche en Informatique et Automatique (INRIA)
Institute for Advanced Science & Engineering Corporation
Lawrence Berkeley National Laboratory
Lawrence Livermore National Laboratory
Los Alamos National Laboratory
Mathworks
Mellanox Technologies
Microsoft
Myricom
NEC Corporation
Oak Ridge National Laboratory
Ohio State University
Pacific Northwest National Laboratory
QLogic Corporation
Sandia National Laboratories
SiCortex, Inc.
Silicon Graphics Inc.
Sun Microsystems, Inc.
Tokyo Institute of Technology
University of Alabama at Birmingham
University of Houston
University of Illinois at Urbana-Champaign
University of Stuttgart, High Performance Computing Center Stuttgart (HLRS)
University of Tennessee, Knoxville
University of Tokyo
University of Wisconsin

Funding for the MPI Forum meetings was partially supported by award #CCF-0816909 from the National Science Foundation.

In addition, the HDF Group provided travel support for one U.S. academic.

Chapter 1

Introduction to MPI

1.1 Overview and Goals

MPI (Message-Passing Interface) is a *message-passing library interface specification*. All parts of this definition are significant. MPI addresses primarily the message-passing parallel programming model, in which data is moved from the address space of one process to that of another process through cooperative operations on each process. (Extensions to the "classical" message-passing model are provided in collective operations, remote-memory access operations, dynamic process creation, and parallel I/O.) MPI is a *specification*, not an implementation; there are multiple implementations of MPI. This specification is for a *library interface*; MPI is not a language, and all MPI operations are expressed as functions, subroutines, or methods, according to the appropriate language bindings, which for C, C++, Fortran-77, and Fortran-95, are part of the MPI standard. The standard has been defined through an open process by a community of parallel computing vendors, computer scientists, and application developers. The next few sections provide an overview of the history of MPI's development.

The main advantages of establishing a message-passing standard are portability and ease of use. In a distributed memory communication environment in which the higher level routines and/or abstractions are built upon lower level message-passing routines the benefits of standardization are particularly apparent. Furthermore, the definition of a message-passing standard, such as that proposed here, provides vendors with a clearly defined base set of routines that they can implement efficiently, or in some cases provide hardware support for, thereby enhancing scalability.

The goal of the Message-Passing Interface simply stated is to develop a widely used standard for writing message-passing programs. As such the interface should establish a practical, portable, efficient, and flexible standard for message passing.

A complete list of goals follows.

- Design an application programming interface (not necessarily for compilers or a system implementation library).
- Allow efficient communication: Avoid memory-to-memory copying, allow overlap of computation and communication, and offload to communication co-processor, where available.
- Allow for implementations that can be used in a heterogeneous environment.

- Allow convenient C, C++, Fortran-77, and Fortran-95 bindings for the interface.
- Assume a reliable communication interface: the user need not cope with communication failures. Such failures are dealt with by the underlying communication subsystem.
- Define an interface that can be implemented on many vendor's platforms, with no significant changes in the underlying communication and system software.
- Semantics of the interface should be language independent.
- The interface should be designed to allow for thread safety.

1.2 Background of MPI-1.0

MPI sought to make use of the most attractive features of a number of existing message-passing systems, rather than selecting one of them and adopting it as the standard. Thus, MPI was strongly influenced by work at the IBM T. J. Watson Research Center [1, 2], Intel's NX/2 [38], Express [12], nCUBE's Vertex [34], p4 [7, 8], and PARMACS [5, 9]. Other important contributions have come from Zipcode [40, 41], Chimp [16, 17], PVM [4, 14], Chameleon [25], and PICL [24].

The MPI standardization effort involved about 60 people from 40 organizations mainly from the United States and Europe. Most of the major vendors of concurrent computers were involved in MPI, along with researchers from universities, government laboratories, and industry. The standardization process began with the Workshop on Standards for Message-Passing in a Distributed Memory Environment, sponsored by the Center for Research on Parallel Computing, held April 29-30, 1992, in Williamsburg, Virginia [48]. At this workshop the basic features essential to a standard message-passing interface were discussed, and a working group established to continue the standardization process.

A preliminary draft proposal, known as MPI1, was put forward by Dongarra, Hempel, Hey, and Walker in November 1992, and a revised version was completed in February 1993 [15]. MPI1 embodied the main features that were identified at the Williamsburg workshop as being necessary in a message passing standard. Since MPI1 was primarily intended to promote discussion and "get the ball rolling," it focused mainly on point-to-point communications. MPI1 brought to the forefront a number of important standardization issues, but did not include any collective communication routines and was not thread-safe.

In November 1992, a meeting of the MPI working group was held in Minneapolis, at which it was decided to place the standardization process on a more formal footing, and to generally adopt the procedures and organization of the High Performance Fortran Forum. Subcommittees were formed for the major component areas of the standard, and an email discussion service established for each. In addition, the goal of producing a draft MPI standard by the Fall of 1993 was set. To achieve this goal the MPI working group met every 6 weeks for two days throughout the first 9 months of 1993, and presented the draft MPI standard at the Supercomputing 93 conference in November 1993. These meetings and the email discussion together constituted the MPI Forum, membership of which has been open to all members of the high performance computing community.

1.3 Background of MPI-1.1, MPI-1.2, and MPI-2.0

Beginning in March 1995, the MPI Forum began meeting to consider corrections and extensions to the original MPI Standard document [21]. The first product of these deliberations was Version 1.1 of the MPI specification, released in June of 1995 [22] (see `http://www.mpi-forum.org` for official MPI document releases). At that time, effort focused in five areas.

1. Further corrections and clarifications for the MPI-1.1 document.
2. Additions to MPI-1.1 that do not significantly change its types of functionality (new datatype constructors, language interoperability, etc.).
3. Completely new types of functionality (dynamic processes, one-sided communication, parallel I/O, etc.) that are what everyone thinks of as "MPI-2 functionality."
4. Bindings for Fortran 90 and C++. MPI-2 specifies C++ bindings for both MPI-1 and MPI-2 functions, and extensions to the Fortran 77 binding of MPI-1 and MPI-2 to handle Fortran 90 issues.
5. Discussions of areas in which the MPI process and framework seem likely to be useful, but where more discussion and experience are needed before standardization (e.g. zero-copy semantics on shared-memory machines, real-time specifications).

Corrections and clarifications (items of type 1 in the above list) were collected in Chapter 3 of the MPI-2 document: "Version 1.2 of MPI." That chapter also contains the function for identifying the version number. Additions to MPI-1.1 (items of types 2, 3, and 4 in the above list) are in the remaining chapters of the MPI-2 document, and constitute the specification for MPI-2. Items of type 5 in the above list have been moved to a separate document, the "MPI Journal of Development" (JOD), and are not part of the MPI-2 Standard.

This structure makes it easy for users and implementors to understand what level of MPI compliance a given implementation has:

- MPI-1 compliance will mean compliance with MPI-1.3. This is a useful level of compliance. It means that the implementation conforms to the clarifications of MPI-1.1 function behavior given in Chapter 3 of the MPI-2 document. Some implementations may require changes to be MPI-1 compliant.
- MPI-2 compliance will mean compliance with all of MPI-2.1.
- The MPI Journal of Development is not part of the MPI Standard.

It is to be emphasized that forward compatibility is preserved. That is, a valid MPI-1.1 program is both a valid MPI-1.3 program and a valid MPI-2.1 program, and a valid MPI-1.3 program is a valid MPI-2.1 program.

1.4 Background of MPI-1.3 and MPI-2.1

After the release of MPI-2.0, the MPI Forum kept working on errata and clarifications for both standard documents (MPI-1.1 and MPI-2.0). The short document "Errata for MPI-1.1" was released October 12, 1998. On July 5, 2001, a first ballot of errata and clarifications for

MPI-2.0 was released, and a second ballot was voted on May 22, 2002. Both votes were done electronically. Both ballots were combined into one document: "Errata for MPI-2", May 15, 2002. This errata process was then interrupted, but the Forum and its e-mail reflectors kept working on new requests for clarification.

Restarting regular work of the MPI Forum was initiated in three meetings, at EuroPVM/MPI'06 in Bonn, at EuroPVM/MPI'07 in Paris, and at SC'07 in Reno. In December 2007, a steering committee started the organization of new MPI Forum meetings at regular 8-weeks intervals. At the January 14-16, 2008 meeting in Chicago, the MPI Forum decided to combine the existing and future MPI documents to one single document for each version of the MPI standard. For technical and historical reasons, this series was started with MPI-1.3. Additional Ballots 3 and 4 solved old questions from the errata list started in 1995 up to new questions from the last years. After all documents (MPI-1.1, MPI-2, Errata for MPI-1.1 (Oct. 12, 1998), and MPI-2.1 Ballots 1-4) were combined into one draft document, for each chapter, a chapter author and review team were defined. They cleaned up the document to achieve a consistent MPI-2.1 document. The final MPI-2.1 standard document was finished in June 2008, and finally released with a second vote in September 2008 in the meeting at Dublin, just before EuroPVM/MPI'08. The major work of the current MPI Forum is the preparation of MPI-3.

1.5 Background of MPI-2.2

MPI-2.2 is a minor update to the MPI-2.1 standard. This version address additional errors and ambiguities that were not corrected in the MPI-2.1 standard as well as a small number of extensions to MPI-2.1 that met the following criteria:

- Any correct MPI-2.1 program is a correct MPI-2.2 program.
- Any extension must have significant benefit for users.
- Any extension must not require significant implementation effort. To that end, all such changes are accompanied by an open source implementation.

The discussions of MPI-2.2 proceeded concurrently with the MPI-3 discussions; in some cases, extensions were proposed for MPI-2.2 but were later moved to MPI-3.

1.6 Who Should Use This Standard?

This standard is intended for use by all those who want to write portable message-passing programs in Fortran, C and C++. This includes individual application programmers, developers of software designed to run on parallel machines, and creators of environments and tools. In order to be attractive to this wide audience, the standard must provide a simple, easy-to-use interface for the basic user while not semantically precluding the high-performance message-passing operations available on advanced machines.

1.7 What Platforms Are Targets For Implementation?

The attractiveness of the message-passing paradigm at least partially stems from its wide portability. Programs expressed this way may run on distributed-memory multiprocessors,

networks of workstations, and combinations of all of these. In addition, shared-memory implementations, including those for multi-core processors and hybrid architectures, are possible. The paradigm will not be made obsolete by architectures combining the shared- and distributed-memory views, or by increases in network speeds. It thus should be both possible and useful to implement this standard on a great variety of machines, including those "machines" consisting of collections of other machines, parallel or not, connected by a communication network.

The interface is suitable for use by fully general MIMD programs, as well as those written in the more restricted style of SPMD. MPI provides many features intended to improve performance on scalable parallel computers with specialized interprocessor communication hardware. Thus, we expect that native, high-performance implementations of MPI will be provided on such machines. At the same time, implementations of MPI on top of standard Unix interprocessor communication protocols will provide portability to workstation clusters and heterogenous networks of workstations.

1.8 What Is Included In The Standard?

The standard includes:

- Point-to-point communication
- Datatypes
- Collective operations
- Process groups
- Communication contexts
- Process topologies
- Environmental Management and inquiry
- The info object
- Process creation and management
- One-sided communication
- External interfaces
- Parallel file I/O
- Language Bindings for Fortran, C and C++
- Profiling interface

1.9 What Is Not Included In The Standard?

The standard does not specify:

- Operations that require more operating system support than is currently standard; for example, interrupt-driven receives, remote execution, or active messages,
- Program construction tools,
- Debugging facilities.

There are many features that have been considered and not included in this standard. This happened for a number of reasons, one of which is the time constraint that was self-imposed in finishing the standard. Features that are not included can always be offered as extensions by specific implementations. Perhaps future versions of MPI will address some of these issues.

1.10 Organization of this Document

The following is a list of the remaining chapters in this document, along with a brief description of each.

- Chapter 2, MPI Terms and Conventions, explains notational terms and conventions used throughout the MPI document.
- Chapter 3, Point to Point Communication, defines the basic, pairwise communication subset of MPI. *Send* and *receive* are found here, along with many associated functions designed to make basic communication powerful and efficient.
- Chapter 4, Datatypes, defines a method to describe any data layout, e.g., an array of structures in the memory, which can be used as message send or receive buffer.
- Chapter 5, Collective Communications, defines process-group collective communication operations. Well known examples of this are barrier and broadcast over a group of processes (not necessarily all the processes). With MPI-2, the semantics of collective communication was extended to include intercommunicators. It also adds two new collective operations.
- Chapter 6, Groups, Contexts, Communicators, and Caching, shows how groups of processes are formed and manipulated, how unique communication contexts are obtained, and how the two are bound together into a *communicator*.
- Chapter 7, Process Topologies, explains a set of utility functions meant to assist in the mapping of process groups (a linearly ordered set) to richer topological structures such as multi-dimensional grids.
- Chapter 8, MPI Environmental Management, explains how the programmer can manage and make inquiries of the current MPI environment. These functions are needed for the writing of correct, robust programs, and are especially important for the construction of highly-portable message-passing programs.

- Chapter 9, The Info Object, defines an opaque object, that is used as input of several MPI routines.
- Chapter 10, Process Creation and Management, defines routines that allow for creation of processes.
- Chapter 11, One-Sided Communications, defines communication routines that can be completed by a single process. These include shared-memory operations (put/get) and remote accumulate operations.
- Chapter 12, External Interfaces, defines routines designed to allow developers to layer on top of MPI. This includes generalized requests, routines that decode MPI opaque objects, and threads.
- Chapter 13, I/O, defines MPI support for parallel I/O.
- Chapter 14, Profiling Interface, explains a simple name-shifting convention that any MPI implementation must support. One motivation for this is the ability to put performance profiling calls into MPI without the need for access to the MPI source code. The name shift is merely an interface, it says nothing about how the actual profiling should be done and in fact, the name shift can be useful for other purposes.
- Chapter 15, Deprecated Functions, describes routines that are kept for reference. However usage of these functions is discouraged, as they may be deleted in future versions of the standard.
- Chapter 16, Language Bindings, describes the C++ binding, discusses Fortran issues, and describes language interoperability aspects between C, C++, and Fortran.

The Appendices are:

- Annex A, Language Bindings Summary, gives specific syntax in C, C++, and Fortran, for all MPI functions, constants, and types.
- Annex B, Change-Log, summarizes major changes since the previous version of the standard.
- Several Index pages are showing the locations of examples, constants and predefined handles, callback routines' prototypes, and all MPI functions.

MPI provides various interfaces to facilitate interoperability of distinct MPI implementations. Among these are the canonical data representation for MPI I/O and for MPI_PACK_EXTERNAL and MPI_UNPACK_EXTERNAL. The definition of an actual binding of these interfaces that will enable interoperability is outside the scope of this document.

A separate document consists of ideas that were discussed in the MPI Forum and deemed to have value, but are not included in the MPI Standard. They are part of the "Journal of Development" (JOD), lest good ideas be lost and in order to provide a starting point for further work. The chapters in the JOD are

- Chapter 2, Spawning Independent Processes, includes some elements of dynamic process management, in particular management of processes with which the spawning processes do not intend to communicate, that the Forum discussed at length but ultimately decided not to include in the MPI Standard.

- Chapter 3, Threads and MPI, describes some of the expected interaction between an MPI implementation and a thread library in a multi-threaded environment.
- Chapter 4, Communicator ID, describes an approach to providing identifiers for communicators.
- Chapter 5, Miscellany, discusses Miscellaneous topics in the MPI JOD, in particular single-copy routines for use in shared-memory environments and new datatype constructors.
- Chapter 6, Toward a Full Fortran 90 Interface, describes an approach to providing a more elaborate Fortran 90 interface.
- Chapter 7, Split Collective Communication, describes a specification for certain non-blocking collective operations.
- Chapter 8, Real-Time MPI, discusses MPI support for real time processing.