15. Section 7.5.4 on page 248.
In MPI_CART_RANK: If comm is associated with a zero-dimensional Cartesian topology, coord is not significant and 0 is returned in rank.

16. Section 7.5.4 on page 248.
In MPI_CART_COORDS: If comm is associated with a zero-dimensional Cartesian topology, coords will be unchanged.

17. Section 7.5.5 on page 252.
In MPI_CART_SHIFT: It is erroneous to call MPI_CART_SHIFT with a direction that is either negative or greater than or equal to the number of dimensions in the Cartesian communicator. This implies that it is erroneous to call MPI_CART_SHIFT with a comm that is associated with a zero-dimensional Cartesian topology.

18. Section 7.5.6 on page 254.
In MPI_CART_SUB: If all entries in remain_dims are false or comm is already associated with a zero-dimensional Cartesian topology then newcomm is associated with a zero-dimensional Cartesian topology.

18.1. Section 8.1.1 on page 259.
The subversion number changed from 0 to 1.

19. Section 8.1.2 on page 260.
In MPI_GET_PROCESSOR_NAME: In C, a null character is additionally stored at name[resultlen]. resultlen cannot be larger then MPI_MAX_PROCESSOR_NAME-1. In Fortran, name is padded on the right with blank characters. resultlen cannot be larger then MPI_MAX_PROCESSOR_NAME.

20. Section 8.3 on page 264.
MPI_{COMM,WIN,FILE}_GET_ERRHANDLER behave as if a new error handler object is created. That is, once the error handler is no longer needed, MPI_ERRHANDLER_FREE should be called with the error handler returned from MPI_ERRHANDLER_GET or MPI_{COMM,WIN,FILE}_GET_ERRHANDLER to mark the error handler for deallocation. This provides behavior similar to that of MPI_COMM_GROUP and MPI_GROUP_FREE.

21. Section 8.7 on page 278, see explanations to MPI_FINALIZE.
MPI_FINALIZE is collective over all connected processes. If no processes were spawned, accepted or connected then this means over MPI_COMM_WORLD; otherwise it is collective over the union of all processes that have been and continue to be connected, as explained in Section 10.5.4 on page 318.

22. Section 8.7 on page 278.
About MPI_ABORT:

> *Advice to users.* Whether the errorcode is returned from the executable or from the MPI process startup mechanism (e.g., mpiexec), is an aspect of quality of the MPI library but not mandatory. (*End of advice to users.*)
>
> *Advice to implementors.* Where possible, a high-quality implementation will try to return the errorcode from the MPI process startup mechanism (e.g. mpiexec or singleton init). (*End of advice to implementors.*)