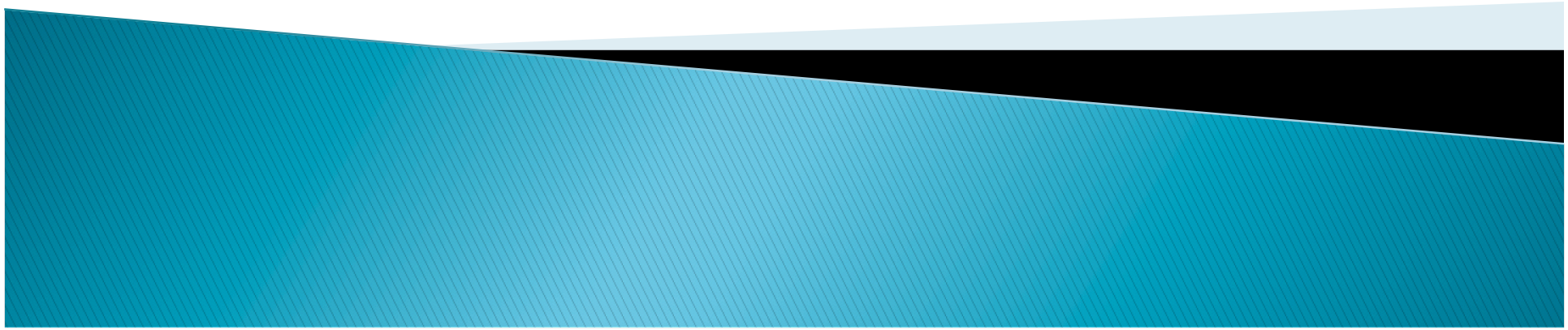
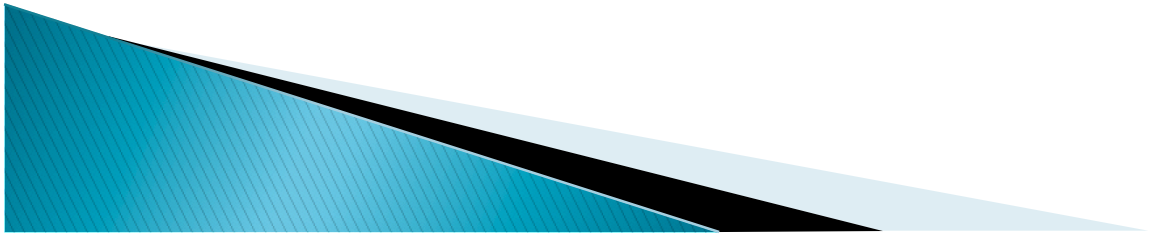
MPI-3 Backwards Compatibility

"All the fun of ABI, but with half the calories"



Suffixes

- Use suffixes for new MPI functions:
  - Tickets current use: MPI_File_write_l (the letter L)
  - …but that suffix can be changed easily
- **QUESTION**: Is "_l" a good suffix?
  - Other possibilities:
    - MPI_File_write_L
    - MPI_File_writel ← this seems problematic
    - MPI_File_writeL
    - …?

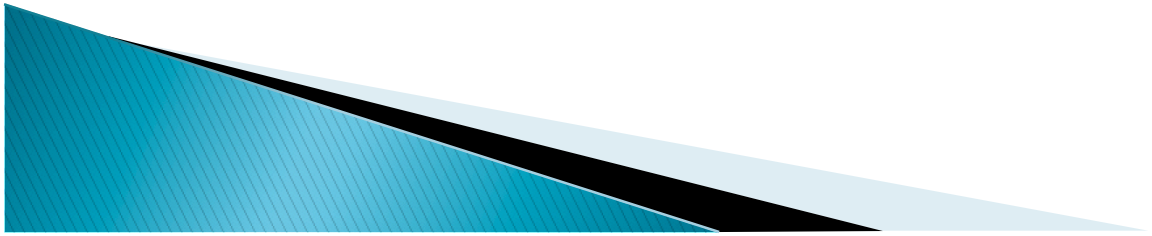


Ticket #117

- https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/117
- Defines an MPI_Count datatype
  - Text is currently relative to MPI-3 trunk HEAD
- Added:
  - MPI_Count items can be used in reductions (like MPI_Offset)
    - MPI_COUNT datatype (and MPI_COUNT_KIND)
  - MPI_GET_ELEMENTS_L
  - MPI_GET_COUNT_L
  - MPI_STATUS_SET_ELEMENTS_L

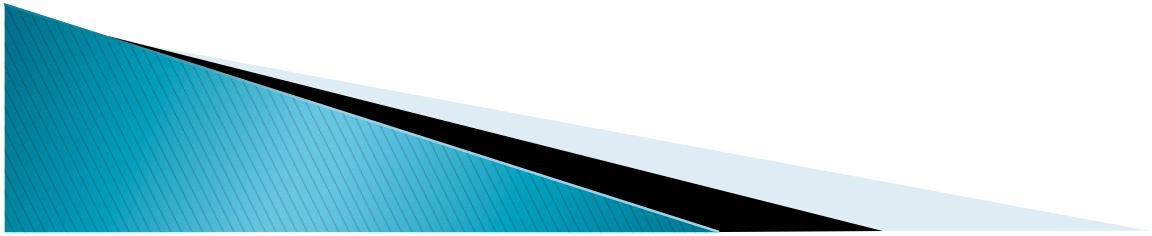
Ticket #117

- Not deprecating anything
  - Can't deprecate MPI_GET_ELEMENTS, etc.
- Not implemented yet; polling Forum for intent first



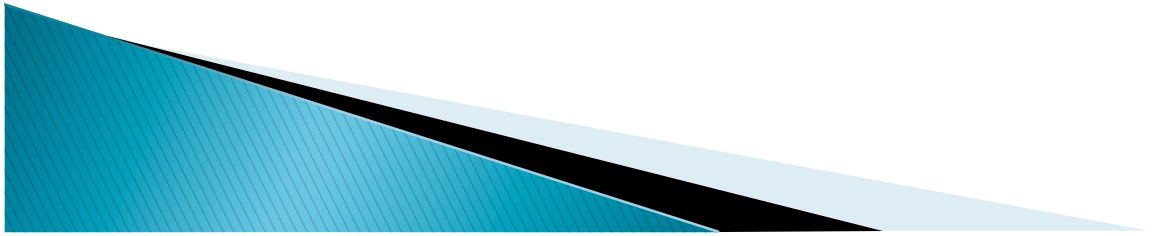
Ticket #117

- QUESTIONS:
  - Ticket changes orientation of text to discuss the "_l" versions of the functions as the primary. Ok?
  - Do we need MPI_UNDEFINED_L?
    - For MPI_GET_COUNT_L
    - We *think* so…
  - What happens if you send 4B elements and call MPI_GET_ELEMENTS?
    - I.e., what is the error case?



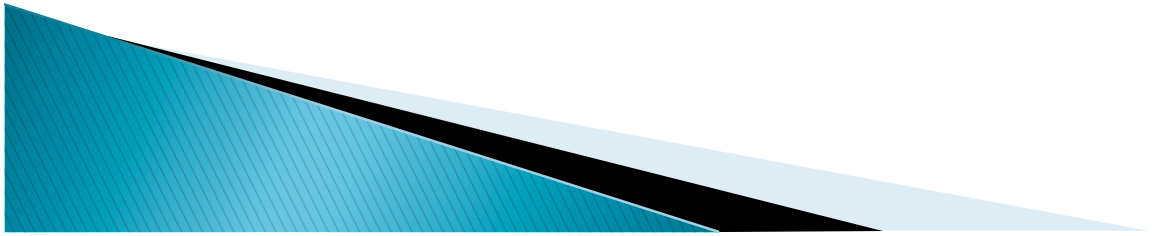
Ticket #220

- https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/220
- Create MPI-IO functions that use MPI_Count
  - Add MPI_File_<foo>_l functions
  - Also add MPI_CONVERSION_FN_NULL_L (etc.)
    - But not MPI_DATAREP_EXTENT_FUNCTION (!)
  - Updated examples to use MPI_Count
- Not implemented yet – polling Forum for intent first



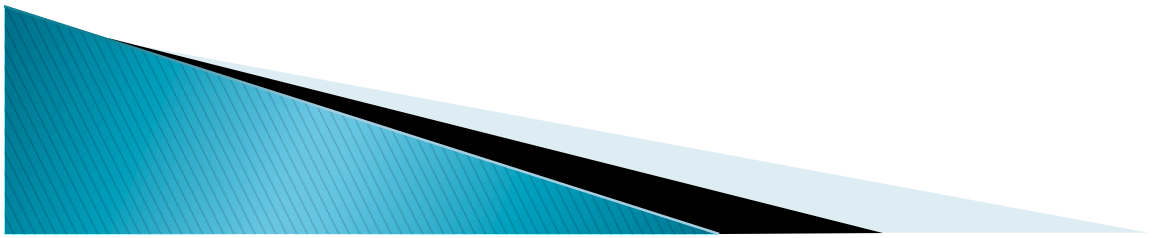
Ticket #220

- QUESTIONS:
  - Ticket changes orientation of text to discuss the "_l" versions of the functions as the primary. Ok?
  - Deprecate the non-_l functions?
    - No text has been added yet about deprecation
    - Will need to list them all in the deprecated chapter if we actually deprecate them
  - MPI_SET_STATUS_ELEMENTS_L: needed if we assume file operations will use generalized requests



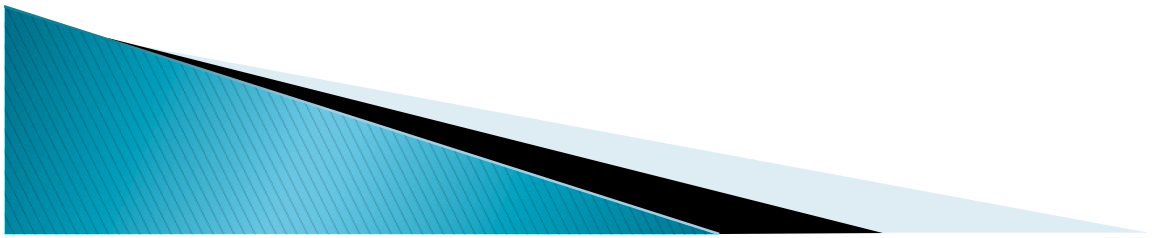
More Questions

- Assuming that we do nothing for C++ bindings for MPI_Count (i.e., no MPI::Count)
- What to do with all new MPI-3 functions?
  - Do they all use MPI_Count?



More Questions

- What about the blocking collectives?
  1. MPI_Bcast, MPI_Ibcast_L(...MPI_Count...)
     - The lack of symmetry seems painful
  2. MPI_Bcast, MPI_Ibcast(...int...)
     - Not using MPI_Count for a new routine seems like a step backward
  3. MPI_Bcast_L(...MPI_Count...), MPI_Ibcast_L (...MPI_Count...)
     - If we introduce _L versions of all the collectives, then why not just do it for all routines…?
  4. MPI_Bcast, MPI_Ibcast(...MPI_Count...)
     - Only using _L extensions when there is a name conflict makes it unclear when a routine takes int vs. MPI_Count.
  - All new MPI-3 functions with a count argument are _L
  - Have all 4 variants: Bcast, Bcast_L, Ibcast, Ibcast_l



- 1. bcast and ibcast both with int
  - ibcast with integer for fortran safety and storage scalability (e.g., gatherv)
- 2. new ticket for bcast_l and ibcast_l
- 3. new ticket(s) for other things with _l
- Point made:
  - Really only need 64 bit queries: get_elements_l and get_count_l
  - So maybe we don't need MPI_Count versions of MPI_File_<foo>_l …etc….?
  - It could be useful to have Datatype constructors that take MPI_Count

More Questions

- If we add _L versions of the collectives…
  - What about MPI_Accumulate?
- If we do MPI_ACCUMULATE_L…
  - What about MPI_GET and MPI_PUT?
- Also note: there's ~15 collectives
  - There's ~15 flavors of send, too
  - So why not do the pt2pt functions?
- WARNING: SLIPPERY SLOPE