```
Index: one-side-2.tex
===================================================================
--- one-side-2.tex	(revision 1767)
+++ one-side-2.tex	(working copy)
@@ -144,7 +144,7 @@
 \cdeclmainindex{MPI\_Win}%
 \cdeclindex{MPI\_Aint}%

-\mpifnewbind{MPI\_Win\_create(base, size, disp\_unit, info, comm,
win, ierror) BIND(C) \fargs TYPE(*), DIMENSION(..), ASYNCHRONOUS ::
base \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: size \\
INTEGER, INTENT(IN) :: disp\_unit \\ TYPE(MPI\_Info), INTENT(IN) ::
info \\ TYPE(MPI\_Comm), INTENT(IN) :: comm \\ TYPE(MPI\_Win),
INTENT(OUT) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_create(base, size, disp\_unit, info, comm,
win, ierror) \fargs TYPE(*), DIMENSION(..), ASYNCHRONOUS :: base \\
INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: size \\ INTEGER,
INTENT(IN) :: disp\_unit \\ TYPE(MPI\_Info), INTENT(IN) :: info \\
TYPE(MPI\_Comm), INTENT(IN) :: comm \\ TYPE(MPI\_Win), INTENT(OUT) ::
win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_CREATE(BASE, SIZE, DISP\_UNIT, INFO, COMM, WIN,
IERROR)\fargs  <type> BASE(*) \\ INTEGER(KIND=MPI\_ADDRESS\_KIND) SIZE
\\INTEGER DISP\_UNIT, INFO, COMM, WIN, IERROR}

 \mpicppemptybind{MPI::Win::Create(const void* base, MPI::Aint size,
int disp\_unit, const MPI::Info\& info, const MPI::Intracomm\& comm)}
{static MPI::Win}
@@ -194,7 +194,7 @@
 \begin{description}
 \item{\infokey{no\_locks}} --- if set to \constskip{true},
 then the implementation may assume that passive target
synchronization (i.e.,
-\mpifunc{MPI\_WIN\_LOCK}, \mpifunc{MPI\_LOCK\_ALL}) will not be used
on
+\mpifunc{MPI\_WIN\_LOCK}, \mpifunc{MPI\_WIN\_LOCK\_ALL}) will not be
used on
 the given window. This implies that this window is not used for 3-
party
 communication, and \RMA/ can be implemented with no (less)
asynchronous
 agent activity at this process.
@@ -300,7 +300,7 @@
 %% views base (baseptr in alloc_mem) as an address-sized integer in
 %% Fortran.  If there is a change in Alloc_mem to use new Fortran
 %% interfaces, this binding should follow the same approach
-\mpifnewbind{MPI\_Win\_allocate(size, disp\_unit, info, comm,
baseptr, win, ierror) BIND(C) \fargs USE, INTRINSIC :: ISO\_C
\_BINDING, ONLY : C\_PTR \\ INTEGER(KIND=MPI\_ADDRESS\_KIND),
INTENT(IN) :: size \\ INTEGER, INTENT(IN) :: disp\_unit \\ TYPE(MPI
\_Info), INTENT(IN) :: info \\ TYPE(MPI\_Comm), INTENT(IN) :: comm \\
TYPE(C\_PTR), INTENT(OUT) :: baseptr \\ TYPE(MPI\_Win), INTENT(OUT) ::
win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_allocate(size, disp\_unit, info, comm,
baseptr, win, ierror) \fargs USE, INTRINSIC :: ISO\_C\_BINDING, ONLY :
C\_PTR \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: size \\
INTEGER, INTENT(IN) :: disp\_unit \\ TYPE(MPI\_Info), INTENT(IN) ::
info \\ TYPE(MPI\_Comm), INTENT(IN) :: comm \\ TYPE(C\_PTR),
INTENT(OUT) :: baseptr \\ TYPE(MPI\_Win), INTENT(OUT) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_ALLOCATE(SIZE, DISP\_UNIT, INFO, COMM, BASEPTR,
WIN, IERROR)\fargs INTEGER DISP\_UNIT, INFO, COMM, WIN, IERROR \\
INTEGER(KIND=MPI\_ADDRESS\_KIND) SIZE, BASEPTR
\mpifoverloadOnlyInAnnex{\>INTERFACE MPI\_WIN\_ALLOCATE \\ \>
\>SUBROUTINE MPI\_WIN\_ALLOCATE\_CPTR(SIZE, DISP\_UNIT, INFO, COMM,
BASEPTR, \& \\ \>\>\>\>WIN, IERROR) \\ \>\>\>USE, INTRINSIC :: ISO\_C
\_BINDING, ONLY : C\_PTR \\ \>\>\>INTEGER :: DISP\_UNIT, INFO, COMM,
WIN, IERROR \\ \>\>\>INTEGER(KIND=MPI\_ADDRESS\_KIND) :: SIZE \\ \>\>
\>TYPE(C\_PTR) :: BASEPTR \\ \>\>END SUBROUTINE \\ \>END INTERFACE}}

 %\mpicppemptybind{MPI::Win::Allocate(MPI::Aint size, int disp\_unit,
%const MPI::Info\& info, const MPI::Intracomm\& comm, void** baseptr)}
{static MPI::Win}
@@ -322,29 +322,38 @@
 \mpiarg{baseptr}.

 If the Fortran compiler provides \ftype{TYPE(C\_PTR)},
-then the following interface must be provided in the \code{mpi}
+then the following generic interface must be provided in the
\code{mpi}
 module and should be provided in \code{mpif.h} through overloading,
 i.e., with the same routine name as the
 routine with \ftype{INTEGER(KIND=MPI\_ADDRESS\_KIND) BASEPTR},
-but with a different linker name:
+but with a different specific procedure name:

 %%HEADER
 %%LANG: FORTRAN90
 %%ENDHEADER
 \begin{verbatim}
 INTERFACE MPI_WIN_ALLOCATE
-  SUBROUTINE MPI_WIN_ALLOCATE_CPTR(SIZE, DISP_UNIT, INFO, COMM,
BASEPTR, &
-      WIN, IERROR)
-    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
-    INTEGER :: DISP_UNIT, INFO, COMM, WIN, IERROR
-    INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
-    TYPE(C_PTR) :: BASEPTR
-  END SUBROUTINE
+    SUBROUTINE MPI_WIN_ALLOCATE(SIZE, DISP_UNIT, INFO, COMM, BASEPTR,
&
```

```
+    WIN, IERROR)
+        IMPORT ::  MPI_ADDRESS_KIND
+        INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
+        INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
+    END SUBROUTINE
+    SUBROUTINE MPI_WIN_ALLOCATE_CPTR(SIZE, DISP_UNIT, INFO, COMM,
BASEPTR, &
+    WIN, IERROR)
+        USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
+        IMPORT ::  MPI_ADDRESS_KIND
+        INTEGER ::  DISP_UNIT, INFO, COMM, WIN, IERROR
+        INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE
+        TYPE(C_PTR) ::  BASEPTR
+    END SUBROUTINE
 END INTERFACE
 \end{verbatim}


-The linker name base of this overloaded function is \mpifunc{MPI\_WIN
\_ALLOCATE\_CPTR}. The implied linker names
+The base procedure name of this overloaded function is
+\mpifunc{MPI\_WIN\_ALLOCATE\_CPTR}. The implied specific procedure
+names
 are described in \sectionref{sec:f90:linker-names}.

 \begin{rationale}
@@ -384,7 +393,7 @@
 %% views base (baseptr in alloc_mem) as an address-sized integer in
 %% Fortran.  If there is a change in Alloc_mem to use new Fortran
 %% interfaces, this binding should follow the same approach
-\mpifnewbind{MPI\_Win\_allocate\_shared(size, disp\_unit, info, comm,
baseptr, win, ierror) BIND(C) \fargs USE, INTRINSIC :: ISO\_C
\_BINDING, ONLY : C\_PTR \\ INTEGER(KIND=MPI\_ADDRESS\_KIND),
INTENT(IN) :: size \\ INTEGER, INTENT(IN) :: disp\_unit \\ TYPE(MPI
\_Info), INTENT(IN) :: info \\ TYPE(MPI\_Comm), INTENT(IN) :: comm \\
TYPE(C\_PTR), INTENT(OUT) :: baseptr \\ TYPE(MPI\_Win), INTENT(OUT) ::
win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_allocate\_shared(size, disp\_unit, info, comm,
baseptr, win, ierror) \fargs USE, INTRINSIC :: ISO\_C\_BINDING, ONLY :
C\_PTR \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: size \\
INTEGER, INTENT(IN) :: disp\_unit \\ TYPE(MPI\_Info), INTENT(IN) ::
info \\ TYPE(MPI\_Comm), INTENT(IN) :: comm \\ TYPE(C\_PTR),
INTENT(OUT) :: baseptr \\ TYPE(MPI\_Win), INTENT(OUT) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_ALLOCATE\_SHARED(SIZE, DISP\_UNIT, INFO, COMM,
BASEPTR, WIN, IERROR)\fargs INTEGER DISP\_UNIT, INFO, COMM, WIN,
IERROR \\ INTEGER(KIND=MPI\_ADDRESS\_KIND) SIZE, BASEPTR
\mpifoverloadOnlyInAnnex{\>INTERFACE MPI\_WIN\_ALLOCATE\_SHARED \\ \>
\>SUBROUTINE MPI\_WIN\_ALLOCATE\_SHARED\_CPTR(SIZE, DISP\_UNIT, INFO,
COMM, \& \\ \>\>\>\>BASEPTR, WIN, IERROR) \\ \>\>\>USE, INTRINSIC ::
```

```
ISO\_C\_BINDING, ONLY : C\_PTR \\ \>\>\>INTEGER :: DISP\_UNIT, INFO,
COMM, WIN, IERROR \\ \>\>\>INTEGER(KIND=MPI\_ADDRESS\_KIND) :: SIZE \\
\>\>\>TYPE(C\_PTR) :: BASEPTR \\ \>\>END SUBROUTINE \\ \>END
INTERFACE}}

 %\mpicppemptybind{MPI::Win::Allocate(MPI::Aint size, int disp\_unit,
%const MPI::Info\& info, const MPI::Intracomm\& comm, void* baseptr)}
{static MPI::Win}
@@ -418,29 +427,36 @@
 calculate remote address offsets with local information only.

 If the Fortran compiler provides \ftype{TYPE(C\_PTR)},
-then the following interface must be provided in the \code{mpi}
+then the following generic interface must be provided in the
\code{mpi}
 module and should be provided in \code{mpif.h} through overloading,
 i.e., with the same routine name as the
 routine with \ftype{INTEGER(KIND=MPI\_ADDRESS\_KIND) BASEPTR},
-but with a different linker name:
+but with a different specific procedure name:

 %%HEADER
 %%LANG: FORTRAN90
 %%ENDHEADER
 \begin{verbatim}
 INTERFACE MPI_WIN_ALLOCATE_SHARED
-  SUBROUTINE MPI_WIN_ALLOCATE_SHARED_CPTR(SIZE, DISP_UNIT, INFO,
COMM, &
-      BASEPTR, WIN, IERROR)
-    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
-    INTEGER :: DISP_UNIT, INFO, COMM, WIN, IERROR
-    INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
-    TYPE(C_PTR) :: BASEPTR
-  END SUBROUTINE
+    SUBROUTINE MPI_WIN_ALLOCATE_SHARED(SIZE, DISP_UNIT, INFO, COMM, &
+    BASEPTR, WIN, IERROR)
+        IMPORT ::  MPI_ADDRESS_KIND
+        INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
+        INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
+    END SUBROUTINE
+    SUBROUTINE MPI_WIN_ALLOCATE_SHARED_CPTR(SIZE, DISP_UNIT, INFO,
COMM, &
+    BASEPTR, WIN, IERROR)
+        USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
+        IMPORT ::  MPI_ADDRESS_KIND
+        INTEGER ::  DISP_UNIT, INFO, COMM, WIN, IERROR
+        INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE
+        TYPE(C_PTR) ::  BASEPTR
+    END SUBROUTINE
 END INTERFACE
 \end{verbatim}

-The linker name base of this overloaded function is\flushline % fix
for margin
-\mpifunc{MPI\_WIN\_ALLOCATE\_SHARED\_CPTR}. The implied linker names
+The base procedure name of this overloaded function is\flushline %
fix for margin
+\mpifunc{MPI\_WIN\_ALLOCATE\_SHARED\_CPTR}. The implied specific
procedure names
 are described in \sectionref{sec:f90:linker-names}.

 The \mpiarg{info} argument can be used to specify hints
@@ -487,7 +503,7 @@

 \mpibind{MPI\_Win\_shared\_query(MPI\_Win win, int rank, MPI\_Aint
*size, int~*disp\_unit, void~*baseptr)}

-\mpifnewbind{MPI\_Win\_shared\_query(win, rank, size, disp\_unit,
baseptr, ierror) BIND(C) \fargs USE, INTRINSIC :: ISO\_C\_BINDING,
ONLY : C\_PTR \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
INTENT(IN) :: rank \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(OUT) ::
size \\ INTEGER, INTENT(OUT) :: disp\_unit \\ TYPE(C\_PTR),
INTENT(OUT) :: baseptr \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_shared\_query(win, rank, size, disp\_unit,
baseptr, ierror) \fargs USE, INTRINSIC :: ISO\_C\_BINDING, ONLY : C
\_PTR \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, INTENT(IN) ::
rank \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(OUT) :: size \\
INTEGER, INTENT(OUT) :: disp\_unit \\ TYPE(C\_PTR), INTENT(OUT) ::
baseptr \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_SHARED\_QUERY(WIN, RANK, SIZE, DISP\_UNIT,
BASEPTR, IERROR)\fargs INTEGER WIN, RANK, DISP\_UNIT, IERROR\\INTEGER
(KIND=MPI\_ADDRESS\_KIND) SIZE, BASEPTR
\mpifoverloadOnlyInAnnex{  INTERFACE MPI\_WIN\_SHARED\_QUERY \\ \>
\>SUBROUTINE MPI\_WIN\_SHARED\_QUERY\_CPTR(WIN, RANK, SIZE, DISP
\_UNIT, \&\\ \>\>\>\>BASEPTR, IERROR) \\ \>\>\>USE, INTRINSIC :: ISO
\_C\_BINDING, ONLY : C\_PTR \\ \>\>\>INTEGER :: WIN, RANK, DISP\_UNIT,
IERROR \\ \>\>\>INTEGER(KIND=MPI\_ADDRESS\_KIND) :: SIZE \\ \>\>
\>TYPE(C\_PTR) :: BASEPTR \\ \>\>END SUBROUTINE \\ \>END INTERFACE}}

 This function queries the process-local address for remote memory
segments
@@ -509,25 +525,34 @@
 was called with \mpiarg{size} $= 0$.

 If the Fortran compiler provides \ftype{TYPE(C\_PTR)},
-then the following interface must be provided in the \code{mpi}
+then the following generic interface must be provided in the
\code{mpi}
 module and should be provided in \code{mpif.h} through overloading,
 i.e., with the same routine name as the
 routine with \ftype{INTEGER(KIND=MPI\_ADDRESS\_KIND) BASEPTR},
-but with a different linker name:
+but with a different specific procedure name:

 \begin{verbatim}
 INTERFACE MPI_WIN_SHARED_QUERY
-  SUBROUTINE MPI_WIN_SHARED_QUERY_CPTR(WIN, RANK, SIZE, DISP_UNIT, &
-      BASEPTR, IERROR)
-    USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
-    INTEGER :: WIN, RANK, DISP_UNIT, IERROR
-    INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
-    TYPE(C_PTR) :: BASEPTR
-  END SUBROUTINE
+    SUBROUTINE MPI_WIN_SHARED_QUERY(WIN, RANK, SIZE, DISP_UNIT, &
+    BASEPTR, IERROR)
+        IMPORT :: MPI_ADDRESS_KIND
+        INTEGER WIN, RANK, DISP_UNIT, IERROR
+        INTEGER (KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
+    END SUBROUTINE
+    SUBROUTINE MPI_WIN_SHARED_QUERY_CPTR(WIN, RANK, SIZE, DISP_UNIT,
&
+    BASEPTR, IERROR)
+        USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
+        IMPORT :: MPI_ADDRESS_KIND
+        INTEGER :: WIN, RANK, DISP_UNIT, IERROR
+        INTEGER(KIND=MPI_ADDRESS_KIND) :: SIZE
+        TYPE(C_PTR) :: BASEPTR
+    END SUBROUTINE
 END INTERFACE
 \end{verbatim}

-The linker name base of this overloaded function is \mpifunc{MPI\_WIN
\_SHARED\_QUERY\_CPTR}. The implied linker names
+The base procedure name of this overloaded function is
+\mpifunc{MPI\_WIN\_SHARED\_QUERY\_CPTR}. The implied specific
+procedure names
 are described in \sectionref{sec:f90:linker-names}.

 \subsection{Window of Dynamically Attached Memory}
@@ -564,7 +589,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Win\_create\_dynamic(MPI\_Info info, MPI\_Comm~comm,
MPI\_Win~*win)}

-\mpifnewbind{MPI\_Win\_create\_dynamic(info, comm, win, ierror)
BIND(C) \fargs TYPE(MPI\_Info), INTENT(IN) :: info \\ TYPE(MPI\_Comm),
INTENT(IN) :: comm \\ TYPE(MPI\_Win), INTENT(OUT) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_create\_dynamic(info, comm, win, ierror)
\fargs TYPE(MPI\_Info), INTENT(IN) :: info \\ TYPE(MPI\_Comm),
INTENT(IN) :: comm \\ TYPE(MPI\_Win), INTENT(OUT) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_CREATE\_DYNAMIC(INFO, COMM, WIN, IERROR)\fargs
INTEGER INFO, COMM, WIN, IERROR}

 %\mpicppemptybind{MPI::Win::Create\_dynamic(const MPI::Info\& info,
const MPI::Intracomm\& comm)}{static MPI::Win}
@@ -625,7 +650,7 @@

 \mpibind{MPI\_Win\_attach(MPI\_Win win, void *base, MPI\_Aint size)}

-\mpifnewbind{MPI\_Win\_attach(win, base, size, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: base \\ INTEGER(KIND=MPI\_ADDRESS\_KIND),
INTENT(IN) :: size \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_attach(win, base, size, ierror) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: base \\ INTEGER(KIND=MPI\_ADDRESS\_KIND),
INTENT(IN) :: size \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_ATTACH(WIN, BASE, SIZE, IERROR)\fargs INTEGER
WIN, IERROR\\<type> BASE(*)\\INTEGER (KIND=MPI\_ADDRESS\_KIND) SIZE}

 %\mpicppemptybind{MPI::Win::Register(void *base, MPI::Aint size)
const}{void}
@@ -689,7 +714,7 @@

 \mpibind{MPI\_Win\_detach(MPI\_Win win, const~void *base)}

-\mpifnewbind{MPI\_Win\_detach(win, base, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: base \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_detach(win, base, ierror) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ TYPE(*), DIMENSION(..), ASYNCHRONOUS ::
base \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_DETACH(WIN, BASE, IERROR)\fargs INTEGER WIN,
IERROR\\<type> BASE(*)}

 %\mpicppemptybind{MPI::Win::Detach(void *base, MPI\_Aint size) const}
{void}
@@ -721,7 +746,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_free(MPI\_Win *win)}

-\mpifnewbind{MPI\_Win\_free(win, ierror) BIND(C) \fargs TYPE(MPI
\_Win), INTENT(INOUT) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) ::
ierror}
+\mpifnewbind{MPI\_Win\_free(win, ierror) \fargs TYPE(MPI\_Win),
INTENT(INOUT) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_FREE(WIN, IERROR)\fargs INTEGER WIN, IERROR}
 \mpicppemptybind{MPI::Win::Free()}{void}
@@ -888,7 +913,7 @@
 \cdeclindex{MPI\_Group}%
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_get\_group(MPI\_Win~win, MPI\_Group~*group)}
-\mpifnewbind{MPI\_Win\_get\_group(win, group, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Group), INTENT(OUT) ::
group \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_get\_group(win, group, ierror) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Group), INTENT(OUT) :: group \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_GET\_GROUP(WIN, GROUP, IERROR)\fargs INTEGER WIN,
GROUP, IERROR}
 \mpicppemptybind{MPI::Win::Get\_group() const}{MPI::Group}

@@ -936,7 +961,7 @@
 \cdeclindex{MPI\_Info}%
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_set\_info(MPI\_Win~win, MPI\_Info~info)}
-\mpifnewbind{MPI\_Win\_set\_info(win, info, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Info), INTENT(IN) ::
info \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_set\_info(win, info, ierror) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Info), INTENT(IN) :: info \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_SET\_INFO(WIN, INFO, IERROR)\fargs INTEGER WIN,
INFO, IERROR}

 \mpifunc{MPI\_WIN\_SET\_INFO} sets new values for the hints of the
window associated with \mpiarg{win}.
@@ -959,7 +984,7 @@
 \cdeclindex{MPI\_Info}%
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_get\_info(MPI\_Win~win, MPI\_Info~*info\_used)}
-\mpifnewbind{MPI\_Win\_get\_info(win, info\_used, ierror) BIND(C)
\fargs TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Info),
INTENT(OUT) :: info\_used \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_get\_info(win, info\_used, ierror) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Info), INTENT(OUT) ::
info\_used \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_GET\_INFO(WIN, INFO\_USED, IERROR)\fargs INTEGER
WIN, INFO\_USED, IERROR}

 \mpifunc{MPI\_WIN\_GET\_INFO} returns a new info object containing
the hints of the window associated with \mpiarg{win}.
@@ -1007,7 +1032,8 @@
 after the \RMA/
 call until the operation completes at the origin.

-The outcome of concurrent conflicting accesses to the same memory
```

```
locations is undefined;
+The resulting data values, or outcome, of concurrent conflicting
+accesses to the same memory locations is undefined;
 if a location is updated by a put or accumulate operation, then
 the outcome of loads or other \RMA/ operations is undefined
 until the updating operation has completed at the target.
@@ -1082,7 +1108,7 @@
 \mpibind{MPI\_Put(const void *origin\_addr, int origin\_count, MPI
\_Datatype origin\_datatype, int target\_rank, MPI\_Aint target\_disp,
int target\_count, MPI\_Datatype target\_datatype, MPI\_Win win)}


-\mpifnewbind{MPI\_Put(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
ierror) BIND(C) \fargs TYPE(*), DIMENSION(..), INTENT(IN),
ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
target\_rank, target\_count \\ TYPE(MPI\_Datatype), INTENT(IN) ::
origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Win), INTENT(IN) ::
win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Put(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
ierror) \fargs TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::
origin\_addr \\ INTEGER, INTENT(IN) :: origin\_count, target\_rank,
target\_count \\ TYPE(MPI\_Datatype), INTENT(IN) :: origin\_datatype,
target\_datatype \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) ::
target\_disp \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_PUT(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN\_DATATYPE,
TARGET\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET\_DATATYPE, WIN,
IERROR)\fargs <type> ORIGIN\_ADDR(*) \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND) TARGET\_DISP \\ INTEGER ORIGIN\_COUNT, ORIGIN\_DATATYPE,
TARGET\_RANK, TARGET\_COUNT, TARGET\_DATATYPE,  WIN, IERROR}


@@ -1199,7 +1225,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Get(void *origin\_addr, int origin\_count, MPI
\_Datatype~origin\_datatype, int~target\_rank, MPI\_Aint~target\_disp,
int~target\_count, MPI\_Datatype~target\_datatype, MPI\_Win~win)}

-\mpifnewbind{MPI\_Get(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
ierror) BIND(C) \fargs TYPE(*), DIMENSION(..), ASYNCHRONOUS :: origin
\_addr \\ INTEGER, INTENT(IN) :: origin\_count, target\_rank, target
\_count \\ TYPE(MPI\_Datatype), INTENT(IN) :: origin\_datatype, target
\_datatype \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target
\_disp \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Get(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
ierror) \fargs TYPE(*), DIMENSION(..), ASYNCHRONOUS :: origin\_addr \\
INTEGER, INTENT(IN) :: origin\_count, target\_rank, target\_count \\
TYPE(MPI\_Datatype), INTENT(IN) :: origin\_datatype, target\_datatype
\\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\
TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) ::
ierror}
 \mpifbind{MPI\_GET(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN\_DATATYPE,
TARGET\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET\_DATATYPE, WIN,
IERROR)\fargs <type> ORIGIN\_ADDR(*) \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND) TARGET\_DISP \\ INTEGER ORIGIN\_COUNT, ORIGIN\_DATATYPE,
TARGET\_RANK, TARGET\_COUNT, TARGET\_DATATYPE, WIN, IERROR}

 % \mpicppemptybind{MPI::Win::Get(const void *origin\_addr, int origin
\_count, const MPI::Datatype\& origin\_datatype, int target\_rank,
MPI::Aint target\_disp, int target\_count, const MPI::Datatype\&
target\_datatype) const}{void}
@@ -1410,7 +1436,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Accumulate(const void~*origin\_addr, int~origin\_count,
MPI\_Datatype~origin\_datatype, int~target\_rank, MPI\_Aint~target
\_disp, int~target\_count, MPI\_Datatype~target\_datatype,  MPI
\_Op~op, MPI\_Win~win)}

-\mpifnewbind{MPI\_Accumulate(origin\_addr, origin\_count, origin
\_datatype, target\_rank, target\_disp, target\_count, target
\_datatype, op, win, ierror) BIND(C) \fargs TYPE(*), DIMENSION(..),
INTENT(IN), ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) ::
origin\_count, target\_rank, target\_count \\ TYPE(MPI\_Datatype),
INTENT(IN) :: origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI
\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Op),
INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Accumulate(origin\_addr, origin\_count, origin
\_datatype, target\_rank, target\_disp, target\_count, target
\_datatype, op, win, ierror) \fargs TYPE(*), DIMENSION(..),
INTENT(IN), ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) ::
origin\_count, target\_rank, target\_count \\ TYPE(MPI\_Datatype),
INTENT(IN) :: origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI
\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Op),
INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_ACCUMULATE(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN
\_DATATYPE, TARGET\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET
\_DATATYPE,  OP, WIN, IERROR) \fargs <type> ORIGIN\_ADDR(*) \\
INTEGER(KIND=MPI\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER ORIGIN
\_COUNT, ORIGIN\_DATATYPE,TARGET\_RANK, TARGET\_COUNT, TARGET
\_DATATYPE,  OP, WIN, IERROR}

 \mpicppemptybind{MPI::Win::Accumulate(const void* origin\_addr, int
origin\_count, const MPI::Datatype\& origin\_datatype, int target
\_rank, MPI::Aint target\_disp, int target\_count, const MPI::Datatype
\& target\_datatype, const MPI::Op\& op) const}{void}
@@ -1553,7 +1579,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Get\_accumulate(const~void~*origin\_addr, int~origin
\_count, MPI\_Datatype~origin\_datatype, void~*result\_addr,
int~result\_count, MPI\_Datatype~result\_datatype, int~target\_rank,
MPI\_Aint~target\_disp, int~target\_count, MPI\_Datatype~target
\_datatype, MPI\_Op~op, MPI\_Win~win)}

-\mpifnewbind{MPI\_Get\_accumulate(origin\_addr, origin\_count, origin
\_datatype, result\_addr, result\_count, result\_datatype, target
\_rank, target\_disp, target\_count, target\_datatype, op, win,
ierror) BIND(C) \fargs TYPE(*), DIMENSION(..), INTENT(IN),
ASYNCHRONOUS :: origin\_addr \\ TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: result\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
result\_count, target\_rank, target\_count \\ TYPE(MPI\_Datatype),
INTENT(IN) :: origin\_datatype, target\_datatype, result\_datatype \\
INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\
TYPE(MPI\_Op), INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \
\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Get\_accumulate(origin\_addr, origin\_count, origin
\_datatype, result\_addr, result\_count, result\_datatype, target
\_rank, target\_disp, target\_count, target\_datatype, op, win,
ierror) \fargs TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::
origin\_addr \\ TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result\_addr \
\ INTEGER, INTENT(IN) :: origin\_count, result\_count, target\_rank,
target\_count \\ TYPE(MPI\_Datatype), INTENT(IN) :: origin\_datatype,
target\_datatype, result\_datatype \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Op), INTENT(IN) :: op
\\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
 \mpifbind{MPI\_GET\_ACCUMULATE(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN
\_DATATYPE, RESULT\_ADDR, RESULT\_COUNT, RESULT\_DATATYPE, TARGET
\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET\_DATATYPE, OP, WIN,
IERROR) \fargs <type> ORIGIN\_ADDR(*), RESULT\_ADDR(*) \\
INTEGER(KIND=MPI\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER ORIGIN
\_COUNT, ORIGIN\_DATATYPE, RESULT\_COUNT, RESULT\_DATATYPE, TARGET
\_RANK, TARGET\_COUNT, TARGET\_DATATYPE, OP, WIN, IERROR}

 %\mpicppemptybind{MPI::Win::Get\_accumulate(const void* origin\_addr,
void* result\_addr, const MPI::Datatype\& datatype, int target\_rank,
MPI::Aint target\_disp, const MPI::Op\& op) const}{void}
@@ -1637,7 +1663,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Fetch\_and\_op(const~void~*origin\_addr, void~*result
\_addr, MPI\_Datatype~datatype, int~target\_rank, MPI\_Aint~target
\_disp, MPI\_Op~op, MPI\_Win~win)}
```

```
-\mpifnewbind{MPI\_Fetch\_and\_op(origin\_addr, result\_addr,
datatype, target\_rank, target\_disp, op, win, ierror) BIND(C) \fargs
TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin\_addr \\
TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result\_addr \\ TYPE(MPI
\_Datatype), INTENT(IN) :: datatype \\ INTEGER, INTENT(IN) :: target
\_rank \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp
\\ TYPE(MPI\_Op), INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) ::
win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Fetch\_and\_op(origin\_addr, result\_addr,
datatype, target\_rank, target\_disp, op, win, ierror) \fargs TYPE(*),
DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin\_addr \\ TYPE(*),
DIMENSION(..), ASYNCHRONOUS :: result\_addr \\ TYPE(MPI\_Datatype),
INTENT(IN) :: datatype \\ INTEGER, INTENT(IN) :: target\_rank \\
INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\
TYPE(MPI\_Op), INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \
\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_FETCH\_AND\_OP(ORIGIN\_ADDR, RESULT\_ADDR, DATATYPE,
TARGET\_RANK, TARGET\_DISP, OP, WIN, IERROR) \fargs <type> ORIGIN
\_ADDR(*), RESULT\_ADDR(*) \\ INTEGER(KIND=MPI\_ADDRESS\_KIND) TARGET
\_DISP \\ INTEGER DATATYPE, TARGET\_RANK, OP, WIN, IERROR}

 %\mpicppemptybind{MPI::Win::Fetch\_and\_op(const void* origin\_addr,
void* result\_addr, const MPI::Datatype\& datatype, int target\_rank,
MPI::Aint target\_disp, const MPI::Op\& op) const}{void}
@@ -1683,7 +1709,7 @@

 % compare_addr gets its own declaration to avoid having it spill to
the next
 % line.
-\mpifnewbind{MPI\_Compare\_and\_swap(origin\_addr, compare\_addr,
result\_addr, datatype, target\_rank, target\_disp, win, ierror)
BIND(C) \fargs TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::
origin\_addr \\ TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::
compare\_addr \\TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result\_addr \
\ TYPE(MPI\_Datatype), INTENT(IN) :: datatype \\ INTEGER,
INTENT(IN) :: target\_rank \\ INTEGER(KIND=MPI\_ADDRESS\_KIND),
INTENT(IN) :: target\_disp \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Compare\_and\_swap(origin\_addr, compare\_addr,
result\_addr, datatype, target\_rank, target\_disp, win, ierror)
\fargs TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin
\_addr \\ TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: compare
\_addr \\TYPE(*), DIMENSION(..), ASYNCHRONOUS :: result\_addr \\
TYPE(MPI\_Datatype), INTENT(IN) :: datatype \\ INTEGER, INTENT(IN) ::
target\_rank \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target
\_disp \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
 \mpifbind{MPI\_COMPARE\_AND\_SWAP(ORIGIN\_ADDR, COMPARE\_ADDR, RESULT
\_ADDR, DATATYPE, TARGET\_RANK, TARGET\_DISP, WIN, IERROR) \fargs
<type> ORIGIN\_ADDR(*), COMPARE\_ADDR(*), RESULT\_ADDR(*) \\
INTEGER(KIND=MPI\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER DATATYPE,
TARGET\_RANK, WIN, IERROR}

 %\mpicppemptybind{MPI::Win::Compare\_and\_swap(const void* origin
\_addr, const void* compare\_addr, void* result\_addr, const
MPI::Datatype\& datatype, int target\_rank, MPI::Aint target\_disp)
const}{void}
@@ -1763,7 +1789,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Rput(const~void *origin\_addr, int origin\_count, MPI
\_Datatype~origin\_datatype, int~target\_rank, MPI\_Aint~target\_disp,
int~target\_count, MPI\_Datatype~target\_datatype, MPI\_Win~win, MPI
\_Request~*request)}

-\mpifnewbind{MPI\_Rput(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
request, ierror) BIND(C) \fargs TYPE(*), DIMENSION(..), INTENT(IN),
ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
target\_rank, target\_count \\ TYPE(MPI\_Datatype), INTENT(IN) ::
origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Win), INTENT(IN) ::
win \\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Rput(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
request, ierror) \fargs TYPE(*), DIMENSION(..), INTENT(IN),
ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
target\_rank, target\_count \\ TYPE(MPI\_Datatype), INTENT(IN) ::
origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Win), INTENT(IN) ::
win \\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_RPUT(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN\_DATATYPE,
TARGET\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET\_DATATYPE, WIN,
REQUEST, IERROR)\fargs <type> ORIGIN\_ADDR(*)\\ INTEGER(KIND=MPI
\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER ORIGIN\_COUNT, ORIGIN
\_DATATYPE, TARGET\_RANK, TARGET\_COUNT, TARGET\_DATATYPE,  WIN,
REQUEST, IERROR}

 \mpifunc{MPI\_RPUT} is similar to \mpifunc{MPI\_PUT}
@@ -1804,7 +1830,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Rget(void *origin\_addr, int origin\_count, MPI
\_Datatype~origin\_datatype, int~target\_rank, MPI\_Aint~target\_disp,
int~target\_count, MPI\_Datatype~target\_datatype, MPI\_Win~win, MPI
\_Request~*request)}

-\mpifnewbind{MPI\_Rget(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
request, ierror) BIND(C) \fargs TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
target\_rank, target\_count \\ TYPE(MPI\_Datatype), INTENT(IN) ::
origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI\_ADDRESS
\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Win), INTENT(IN) ::
win \\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Rget(origin\_addr, origin\_count, origin\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, win,
request, ierror) \fargs TYPE(*), DIMENSION(..), ASYNCHRONOUS :: origin
\_addr \\ INTEGER, INTENT(IN) :: origin\_count, target\_rank, target
\_count \\ TYPE(MPI\_Datatype), INTENT(IN) :: origin\_datatype, target
\_datatype \\ INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target
\_disp \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(MPI\_Request),
INTENT(OUT) :: request \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_RGET(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN\_DATATYPE,
TARGET\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET\_DATATYPE, WIN,
REQUEST, IERROR)\fargs <type> ORIGIN\_ADDR(*) \\ INTEGER(KIND=MPI
\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER ORIGIN\_COUNT, ORIGIN
\_DATATYPE, TARGET\_RANK, TARGET\_COUNT, TARGET\_DATATYPE, WIN,
REQUEST, IERROR}

 \mpifunc{MPI\_RGET} is similar to \mpifunc{MPI\_GET}
@@ -1836,7 +1862,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Raccumulate(const~void~*origin\_addr, int~origin
\_count, MPI\_Datatype~origin\_datatype, int~target\_rank, MPI
\_Aint~target\_disp, int~target\_count, MPI\_Datatype~target
\_datatype,  MPI\_Op~op, MPI\_Win~win, MPI\_Request~*request)}

-\mpifnewbind{MPI\_Raccumulate(origin\_addr, origin\_count, origin
\_datatype, target\_rank, target\_disp, target\_count, target
\_datatype, op, win, request, ierror) BIND(C) \fargs TYPE(*),
DIMENSION(..), INTENT(IN), ASYNCHRONOUS :: origin\_addr \\ INTEGER,
INTENT(IN) :: origin\_count, target\_rank, target\_count \\ TYPE(MPI
\_Datatype), INTENT(IN) :: origin\_datatype, target\_datatype \\
INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\
TYPE(MPI\_Op), INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \
\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Raccumulate(origin\_addr, origin\_count, origin
\_datatype, target\_rank, target\_disp, target\_count, target
\_datatype, op, win, request, ierror) \fargs TYPE(*), DIMENSION(..),
INTENT(IN), ASYNCHRONOUS :: origin\_addr \\ INTEGER, INTENT(IN) ::
origin\_count, target\_rank, target\_count \\ TYPE(MPI\_Datatype),
INTENT(IN) :: origin\_datatype, target\_datatype \\ INTEGER(KIND=MPI
\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\ TYPE(MPI\_Op),
INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ TYPE(MPI
\_Request), INTENT(OUT) :: request \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
 \mpifbind{MPI\_RACCUMULATE(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN
\_DATATYPE, TARGET\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET
\_DATATYPE, OP, WIN, REQUEST, IERROR) \fargs <type> ORIGIN\_ADDR(*) \\
INTEGER(KIND=MPI\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER ORIGIN
\_COUNT, ORIGIN\_DATATYPE, TARGET\_RANK, TARGET\_COUNT, TARGET
\_DATATYPE, OP, WIN, REQUEST, IERROR}

 \mpifunc{MPI\_RACCUMULATE} is similar to \mpifunc{MPI\_ACCUMULATE}
@@ -1872,7 +1898,7 @@
 \cdeclindex{MPI\_Aint}%
 \mpibind{MPI\_Rget\_accumulate(const~void~*origin\_addr, int~origin
\_count, MPI\_Datatype~origin\_datatype, void~*result\_addr,
int~result\_count, MPI\_Datatype~result\_datatype, int~target\_rank,
MPI\_Aint~target\_disp, int~target\_count, MPI\_Datatype~target
\_datatype, MPI\_Op~op, MPI\_Win~win, MPI\_Request~*request)}

-\mpifnewbind{MPI\_Rget\_accumulate(origin\_addr, origin\_count,
origin\_datatype, result\_addr, result\_count, result\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, op, win,
request, ierror) BIND(C) \fargs TYPE(*), DIMENSION(..), INTENT(IN),
ASYNCHRONOUS :: origin\_addr \\ TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: result\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
result\_count, target\_rank, target\_count \\ TYPE(MPI\_Datatype),
INTENT(IN) :: origin\_datatype, target\_datatype, result\_datatype \\
INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\
TYPE(MPI\_Op), INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \
\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Rget\_accumulate(origin\_addr, origin\_count,
origin\_datatype, result\_addr, result\_count, result\_datatype,
target\_rank, target\_disp, target\_count, target\_datatype, op, win,
request, ierror) \fargs TYPE(*), DIMENSION(..), INTENT(IN),
ASYNCHRONOUS :: origin\_addr \\ TYPE(*), DIMENSION(..),
ASYNCHRONOUS :: result\_addr \\ INTEGER, INTENT(IN) :: origin\_count,
result\_count, target\_rank, target\_count \\ TYPE(MPI\_Datatype),
INTENT(IN) :: origin\_datatype, target\_datatype, result\_datatype \\
INTEGER(KIND=MPI\_ADDRESS\_KIND), INTENT(IN) :: target\_disp \\
TYPE(MPI\_Op), INTENT(IN) :: op \\ TYPE(MPI\_Win), INTENT(IN) :: win \
\ TYPE(MPI\_Request), INTENT(OUT) :: request \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
 \mpifbind{MPI\_RGET\_ACCUMULATE(ORIGIN\_ADDR, ORIGIN\_COUNT, ORIGIN
\_DATATYPE, RESULT\_ADDR, RESULT\_COUNT, RESULT\_DATATYPE, TARGET
\_RANK, TARGET\_DISP, TARGET\_COUNT, TARGET\_DATATYPE, OP, WIN,
REQUEST, IERROR) \fargs <type> ORIGIN\_ADDR(*), RESULT\_ADDR(*) \\
INTEGER(KIND=MPI\_ADDRESS\_KIND) TARGET\_DISP \\ INTEGER ORIGIN
\_COUNT, ORIGIN\_DATATYPE, RESULT\_COUNT, RESULT\_DATATYPE, TARGET
\_RANK, TARGET\_COUNT, TARGET\_DATATYPE, OP, WIN, REQUEST, IERROR}

 \mpifunc{MPI\_RGET\_ACCUMULATE} is similar to
@@ -2170,7 +2196,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_fence(int~assert, MPI\_Win~win)}

-\mpifnewbind{MPI\_Win\_fence(assert, win, ierror) BIND(C) \fargs
INTEGER, INTENT(IN) :: assert \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_fence(assert, win, ierror) \fargs INTEGER,
INTENT(IN) :: assert \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_FENCE(ASSERT, WIN, IERROR)\fargs INTEGER ASSERT,
WIN, IERROR}

 \mpicppemptybind{MPI::Win::Fence(int assert) const}{void}
@@ -2233,7 +2259,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_start(MPI\_Group group, int assert, MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_start(group, assert, win, ierror) BIND(C)
\fargs TYPE(MPI\_Group), INTENT(IN) :: group \\ INTEGER, INTENT(IN) ::
assert \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_start(group, assert, win, ierror) \fargs
TYPE(MPI\_Group), INTENT(IN) :: group \\ INTEGER, INTENT(IN) :: assert
\\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_START(GROUP, ASSERT, WIN, IERROR)\fargs INTEGER
GROUP, ASSERT, WIN, IERROR}

 \mpicppemptybind{MPI::Win::Start(const MPI::Group\& group, int
assert) const}{void}
@@ -2263,7 +2289,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_complete(MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_complete(win, ierror) BIND(C) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_complete(win, ierror) \fargs TYPE(MPI\_Win),
INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_COMPLETE(WIN, IERROR)\fargs INTEGER WIN,  IERROR}

 \mpicppemptybind{MPI::Win::Complete() const}{void}
@@ -2329,7 +2355,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_post(MPI\_Group group, int assert, MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_post(group, assert, win, ierror) BIND(C)
\fargs TYPE(MPI\_Group), INTENT(IN) :: group \\ INTEGER, INTENT(IN) ::
assert \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_post(group, assert, win, ierror) \fargs
TYPE(MPI\_Group), INTENT(IN) :: group \\ INTEGER, INTENT(IN) :: assert
\\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL,
INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_POST(GROUP, ASSERT, WIN, IERROR)\fargs INTEGER
GROUP, ASSERT, WIN, IERROR}

 \mpicppemptybind{MPI::Win::Post(const MPI::Group\& group, int assert)
const}{void}
@@ -2350,7 +2376,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_wait(MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_wait(win, ierror) BIND(C) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_wait(win, ierror) \fargs TYPE(MPI\_Win),
INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_WAIT(WIN, IERROR)\fargs INTEGER WIN,  IERROR}

 \mpicppemptybind{MPI::Win::Wait() const}{void}
@@ -2396,7 +2422,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_test(MPI\_Win win, int *flag)}

-\mpifnewbind{MPI\_Win\_test(win, flag, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ LOGICAL, INTENT(OUT) :: flag \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_test(win, flag, ierror) \fargs TYPE(MPI\_Win),
INTENT(IN) :: win \\ LOGICAL, INTENT(OUT) :: flag \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_TEST(WIN, FLAG, IERROR)\fargs INTEGER WIN, IERROR
\\LOGICAL FLAG}

 \mpicppemptybind{MPI::Win::Test() const}{bool}
@@ -2501,7 +2527,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_lock(int lock\_type, int rank, int assert, MPI
\_Win win)}

-\mpifnewbind{MPI\_Win\_lock(lock\_type, rank, assert, win, ierror)
BIND(C) \fargs INTEGER, INTENT(IN) :: lock\_type, rank, assert \\
TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) ::
ierror}
+\mpifnewbind{MPI\_Win\_lock(lock\_type, rank, assert, win, ierror)
\fargs INTEGER, INTENT(IN) :: lock\_type, rank, assert \\ TYPE(MPI
\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_LOCK(LOCK\_TYPE, RANK, ASSERT, WIN, IERROR)\fargs
INTEGER LOCK\_TYPE, RANK, ASSERT, WIN, IERROR}

 \mpicppemptybind{MPI::Win::Lock(int lock\_type, int rank, int assert)
const}{void}
@@ -2519,7 +2545,7 @@
```

```
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_lock\_all(int assert, MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_lock\_all(assert, win, ierror) BIND(C) \fargs
INTEGER, INTENT(IN) :: assert \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_lock\_all(assert, win, ierror) \fargs INTEGER,
INTENT(IN) :: assert \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_LOCK\_ALL(ASSERT, WIN, IERROR)\fargs INTEGER
ASSERT, WIN, IERROR}

 %\mpicppemptybind{MPI::Win::Lock\_all(int assert) const}{void}
@@ -2548,7 +2574,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_unlock(int rank, MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_unlock(rank, win, ierror) BIND(C) \fargs
INTEGER, INTENT(IN) :: rank \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_unlock(rank, win, ierror) \fargs INTEGER,
INTENT(IN) :: rank \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_UNLOCK(RANK, WIN, IERROR)\fargs INTEGER RANK,
WIN, IERROR}

 \mpicppemptybind{MPI::Win::Unlock(int rank) const}{void}
@@ -2564,7 +2590,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_unlock\_all(MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_unlock\_all(win, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) ::
ierror}
+\mpifnewbind{MPI\_Win\_unlock\_all(win, ierror) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_UNLOCK\_ALL(WIN, IERROR)\fargs INTEGER WIN,
IERROR}

 %\mpicppemptybind{MPI::Win::Unlock\_all() const}{void}
@@ -2688,7 +2714,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_flush(int rank, MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_flush(rank, win, ierror) BIND(C) \fargs
INTEGER, INTENT(IN) :: rank \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_flush(rank, win, ierror) \fargs INTEGER,
INTENT(IN) :: rank \\ TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER,
OPTIONAL, INTENT(OUT) :: ierror}
```

```
 \mpifbind{MPI\_WIN\_FLUSH(RANK, WIN, IERROR)\fargs INTEGER RANK, WIN,
IERROR}

 %\mpicppemptybind{MPI::Win::Flush(int rank) const}{void}
@@ -2704,7 +2730,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_flush\_all(MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_flush\_all(win, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) ::
ierror}
+\mpifnewbind{MPI\_Win\_flush\_all(win, ierror) \fargs TYPE(MPI\_Win),
INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_FLUSH\_ALL(WIN, IERROR)\fargs INTEGER WIN,
IERROR}

 %\mpicppemptybind{MPI::Win::Flush\_all() const}{void}
@@ -2722,7 +2748,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_flush\_local(int rank, MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_flush\_local(rank, win, ierror) BIND(C) \fargs
INTEGER, INTENT(IN) :: rank \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
+\mpifnewbind{MPI\_Win\_flush\_local(rank, win, ierror) \fargs
INTEGER, INTENT(IN) :: rank \\ TYPE(MPI\_Win), INTENT(IN) :: win \\
INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_FLUSH\_LOCAL(RANK, WIN, IERROR)\fargs INTEGER
RANK, WIN, IERROR}

 %\mpicppemptybind{MPI::Win::Flush\_local(int rank) const}{void}
@@ -2739,7 +2765,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_flush\_local\_all(MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_flush\_local\_all(win, ierror) BIND(C) \fargs
TYPE(MPI\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) ::
ierror}
+\mpifnewbind{MPI\_Win\_flush\_local\_all(win, ierror) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_FLUSH\_LOCAL\_ALL(WIN, IERROR)\fargs INTEGER WIN,
IERROR}

 %\mpicppemptybind{MPI::Win::Flush\_local\_all() const}{void}
@@ -2755,7 +2781,7 @@
 \cdeclindex{MPI\_Win}%
 \mpibind{MPI\_Win\_sync(MPI\_Win win)}

-\mpifnewbind{MPI\_Win\_sync(win, ierror) BIND(C) \fargs TYPE(MPI
\_Win), INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
```

```
+\mpifnewbind{MPI\_Win\_sync(win, ierror) \fargs TYPE(MPI\_Win),
INTENT(IN) :: win \\ INTEGER, OPTIONAL, INTENT(OUT) :: ierror}
 \mpifbind{MPI\_WIN\_SYNC(WIN, IERROR)\fargs INTEGER WIN, IERROR}

 %\mpicppemptybind{MPI::Win::sync() const}{void}
```