3.12.12 mrapi_rmem_atype_t

The `mrapi_rmem_atype_t` type is used to specify the access type to be used for remote memory (see Section 3.5.2 and Section 4.4.2). Access semantics are per remote memory buffer instance, and are either *strict* (meaning all clients must use the same access type), or *any* (meaning that clients may use any type supported by the MRAPI implementation). Implementations may define multiple access types (depending on underlying silicon capabilities), but must provide at minimum: `MRAPI_RMEM_ATYPE_ANY` (which indicates any semantics), and `MRAPI_RMEM_ATYPE_DEFAULT`, which has strict semantics. Note that `MRAPI_RMEM_ATYPE_ANY` is only valid for remote memory buffer creation, clients must use `MRAPI_RMEM_ATYPE_DEFAULT` or another specific type of access mechanism provided by the MRAPI implementation (for example DMA, etc.)

3.12.13 Identifiers: mrapi_mutex_id_t, mrapi_sem_id_t, mrapi_shmem_id_t, mrapi_rmem_id_t

The `mrapi_mutex_id_t, mrapi_sem_id_t, mrapi_shmem_id_t,` and `mrapi_rmem_id_t` types are used to get shared resources. The ID types are only used to get handles to the associated types of MRAPI entities.

- These ids may either be known *a priori* or passed as messages to the other nodes.
- The implementation defines what is "invalid". For any identifier, `mrapi_`*X*`_id` (for example `mrapi_mutex_id_t,` where *X*=`mutex`) there is a pair of corresponding identifiers in the MRAPI header file: `MRAPI_MAX_`*X*`_ID` and `MRAPI_MAX_USER_`*X*`_ID`, which can be examined by the application writer to determine valid ID ranges. MRAPI also supports `MRAPI_`*X*`_ID_ANY` (as in MCAPI endpoint creation). Thus, user specified ids can range from `0..MRAPI_MAX_USER_`*X*`_ID` and 'ANY' ids range from `MRAPI_MAX_USER_`*X*`_ID+1 .. MRAPI_MAX_`*X*`_ID`
- The user-specified space is disjoint from the ANY space to avoid race conditions for the user-specified ids.

3.12.14 Scalars: mrapi_uint64_t, mrapi_uint32_t, mrapi_uint16_t mrapi_uint8_t, mrapi_int64_t, mrapi_int32_t, mrapi_int16_t & mrapi_int8_t

The `mrapi_uint64_t`, `mrapi_uint32_t`, `mrapi_uint16_t`, `mrapi_uint8_t` , `mrapi_int64_t`, `mrapi_int32_t`, `mrapi_int16_t`, and `mrapi_int8_t` types are used for signed and unsigned 64-, 32-, 16-, and 8-bit scalars.

3.12.15 mrapi_request_t

The `mrapi_request_t` type is used to record the state of a pending non-blocking MRAPI transaction (see Section 4.5). Non-blocking MRAPI routines exist for only for reading and writing remote memory. An `mrapi_request_t` can only be used by the node it was created on. The `mrapi_request_t` has an `mca_request_t` equivalent.

NOTE: The MRAPI API user should not attempt to examine the contents of this datatype as this can result in non-portable application code.

3.12.16 mrapi_status_t

The `mrapi_status_t` type is an enumerated type used to record the result of an MRAPI API call. If a status can be returned by an API call, the associated MRAPI API call will allow a `mrapi_status_t` to be passed by reference. The API call will fill in the status code and the API user may examine the `mrapi_status_t` variable to determine the result of the call. The `mrapi_status_t` has an `mca_status_t` equivalent.