4.3 MRAPI Synchronization Primitives

MRAPI supports three types of synchronization primitives: mutexes, semaphores and reader/writer locks. They provide locking functionality through the use of a flag (mutex) or a counter (semaphores) or combination of flag and counter (reader/writer locks). Although a binary semaphore can be used as a mutex, MRAPI explicitly provides mutexes to allow for hardware acceleration. Although Reader/Writer locks can be implemented on top of mutexes and semaphores, MRAPI provides them as a convenience.

Within MRAPI, there is no concept of ownership for the synchronization primitives. Any node may create or get a mutex, semaphore or reader/writer lock (provided it knows the shared key) and any node may delete the mutex, semaphore or reader/writer lock. To support performance/debuggability tradeoffs, MRAPI provides two types of error checking; basic (default) and extended (enabled via the `MRAPI_ERROR_EXT` attribute). When extended error checking is enabled, if lock is called on a mutex, semaphore or reader/writer lock that no longer exists, an MRAPI_ERR_[MUTEX|SEM|RWL]_DELETED error code will be returned. When extended error checking is disabled, the MRAPI_ERR_[MUTEX | SEM | RWL]_INVALID error will be returned and the lock will fail. The benefit of extended error checking is for early functional verification/validation of the code and the working group feels this is a valuable feature for easing the burden of multicore development and debugging. Because extended error checking can be resource intensive, it is optional and disabled by default.

By default, the synchronization primitives are shared across domains. Set the MRAPI_DOMAIN_SHARED attribute to false when you create the mutex, semaphore or reader/writer lock to disable resource sharing across domains. Note that we can not always expect sharing across domains to be efficient.