- it is normally recommended that System V or POSIX®.1b semaphores should be used for process-to-process synchronization rather than pthreads mutexes (but this requires an SMP operating system at this time for Multicore)
- mutexes are useful for managing access to a single resource and simpler to use than System V and POSIX® semaphores
- priorities and associated protocols (PTHREAD_PRIO_NONE, PTHREAD_PRIO_INHERIT, PTHREAD_PRIO_PROTECT) are probably not something that could be guaranteed by MRAPI unless/until MTAPI is created; the MRAPI group chose to defer considering this feature of the POSIX®; we recognize that this puts the burden of dealing with priority inversion squarely on the applications programmer at this time
- the attribute 'types' for error checking are powerful and useful and are included in MRAPI

It was decided for MRAPI to cover a subset of POSIX® mutex functionality along with the following new requirements as follows:

- functionality equivalent to: `pthread_mutex_init, pthread_mutex_destroy, pthread_mutex_lock, pthread_mutex_trylock, pthread_mutex_unlock`
- mutex attributes for reporting basic deadlock detection
- the ability to manage mutex attributes in a way that is consistent with MCAPI
- non-blocking operations in a way that is consistent with MCAPI
- default is for the mutex to be visible across processes/tasks/etc.
- does not require shared memory or SMP OS
- priority inversion cannot be dealt with by MRAPI until MTAPI comes along

2.2.2.3 POSIX® Semaphores

POSIX® semaphores are declared as part of either the 'Realtime' services or the 'XSI Interprocess Communications' services. XSI is the 'X/Open System Interface Extension', which is an extension to IEEE 1003.1b. The XSI interfaces are essentially the same as the System V IPC interfaces, which have been widely supported across most Unix systems. Functionality marked XSI is also an extension to the ISO C standard. Semaphores themselves are a POSIX® option and are not required on all implementations.

2.2.2.4 POSIX® Semaphores Analysis

After reviewing the semaphores API and semantics, the working group came to the following conclusions:

- According to the standard REALTIME semaphores may be process-private or process-shared; there is a lot of evidence that not all operating systems support process-shared REALTIME semaphores - notably Linux - and the standard does not state that process-shared is required.
- REALTIME supports named and unnamed semaphores. Named and unnamed semaphores have distinct operations, for example you must call `sem_close` to close a named semaphore and `sem_unlink` to destroy a named semaphore, while `sem_destroy` is used to close and destroy an un-named semaphore
- It is unspecified whether XSI functions can interoperate with the realtime interprocess communication facilities defined in Realtime.
- The REALTIME API is much simpler to use, the XSI interface is more tied to the operating system although it is clearly defined to be flexible and fast. REALTIME works on a single `sem_t` identifier, while XSI uses arrays of semaphores and arrays of operations per API call
- The `sem_wait, sem_trywait, and sem_timedwait` functions do provide simple deadlock detection errors

For MRAPI, the working group decided the following: