



Memory Manager

Component name		Memory Manager	
Category (IP/ Reusable)	IP	Component type (HW/SW/product etc)	SW
HW Platform	NA	SW Platform /OS	Windows
Applications / applicable products	C++ code for platforms with memory limitations		
Market/ industries applicable	General		
Product Description	<p>Memory Manager for reserving memory in heap for working with controlled number of instances of a class.</p> <p>The default new and delete operators which ship with the compiler are not optimized for any special allocation pattern. Overridden new and delete operators which are optimized for rapid construction and destruction of objects in the heap.</p> <p>Reusable pool of objects which are refreshed and reissued again for new memory allocation requests.</p>		
Technical specifications	<ol style="list-style-type: none">1. In the declaration of the class whose allocation is to be managed, USE_MEMPOOL macro as below:<pre>class ClassA { public: CClassA(void); ~CClassA(void); . . private: int m_nVar ; USE_MEMPOOL(ClassA); };</pre>2. In the cpp file of the class, put the INITIALIZE_MEMPOOL macro<pre>INITIALIZE_MEMPOOL(ClassA); ClassA::ClassA(void) { } </pre>3. During process initialization, call the static function of the class for allocating specific number of instances of the class.<pre>CParcel::CreatePool(POOL_SIZE_PARCEL); //POOL_SIZE_PARCEL ->constant for #</pre>3. During process de-initialization, call the static function of the class for de-allocating the allocated block.<pre>CParcel::FreePool();</pre>		



Features /benefits	Ultra fast memory allocation for new memory requests on the heap for any type of objects whose object pool is constructed and maintained in the heap. Minimal code change required in existing code as new and delete operators are overridden. More detailed usage statistics for memory allocation requests. Macros for creating and registering objects in the pool.
Readiness	Available