

smacc::ISmaccStateMachine

# nh\_  
# private\_nh\_  
# timer  
# stateMachinePub\_  
# stateMachineStatusPub\_  
# transitionLogPub\_  
# transitionHistoryService\_  
# currentState\_  
# currentStateInfo\_  
# status\_msg\_  
# orthogonals\_  
- m\_mutex\_  
- eventQueueMutex\_  
- stateMachineCurrentAction  
- stateCallbackConnections  
- globalData\_  
- transitionLogHistory\_  
- runMode\_  
- signalDetector\_  
- stateSeqCounter\_  
- stateMachineInfo\_  
  
+ ISmaccStateMachine()  
+ ~ISmaccStateMachine()  
+ reset()  
+ stop()  
+ eStop()  
+ getOrthogonal()  
+ getOrthogonals()  
+ requiresComponent()  
+ postEvent()  
+ postEvent()  
+ getTransitionLogHistory()  
+ getGlobalSMDData()  
+ setGlobalSMDData()  
+ mapBehavior()  
+ getStateMachineName()  
+ state\_machine\_visualization()  
+ getCurrentStateInfo()  
+ publishTransition()  
+ onInitialize()  
+ getTransitionLogHistory()  
+ createSignalConnection()  
+ notifyOnStateEntryStart()  
+ notifyOnStateEntryEnd()  
+ notifyOnRuntimeConfigured()  
+ notifyOnStateExiting()  
+ notifyOnStateExited()  
+ notifyOnRuntimeConfigurationFinished()  
+ getCurrentStateCounter()  
+ getCurrentState()  
+ getStateMachineInfo()  
+ buildStateMachineInfo()  
+ getNode()  
# checkStateMachineConsistence()  
# initializeROS()  
# onInitialized()  
# createOrthogonal()  
# getParam()  
# setParam()  
# param()  
- lockStateMachine()  
- unlockStateMachine()  
- propagateEventToStateReactors()  
- updateStatusMessage()

boost::statechart::  
asynchronous\_state\_machine  
< DerivedStateMachine, InitialState  
Type, SmaccFifoScheduler, SmaccAllocator >

smacc::SmaccStateMachine  
Base< DerivedStateMachine,  
InitialStateType >

+ SmaccStateMachineBase()  
+ ~SmaccStateMachineBase()  
+ reset()  
+ stop()  
+ eStop()  
+ initiate\_impl()

boost::statechart::  
asynchronous\_state\_machine  
< SmUpdateLoop, State1,  
SmaccFifoScheduler, SmaccAllocator >

smacc::SmaccStateMachine  
Base< SmUpdateLoop, State1 >

+ SmaccStateMachineBase()  
+ ~SmaccStateMachineBase()  
+ reset()  
+ stop()  
+ eStop()  
+ initiate\_impl()

sm\_update\_loop::SmUpdateLoop

+ onInitialize()

< SmUpdateLoop, State1 >