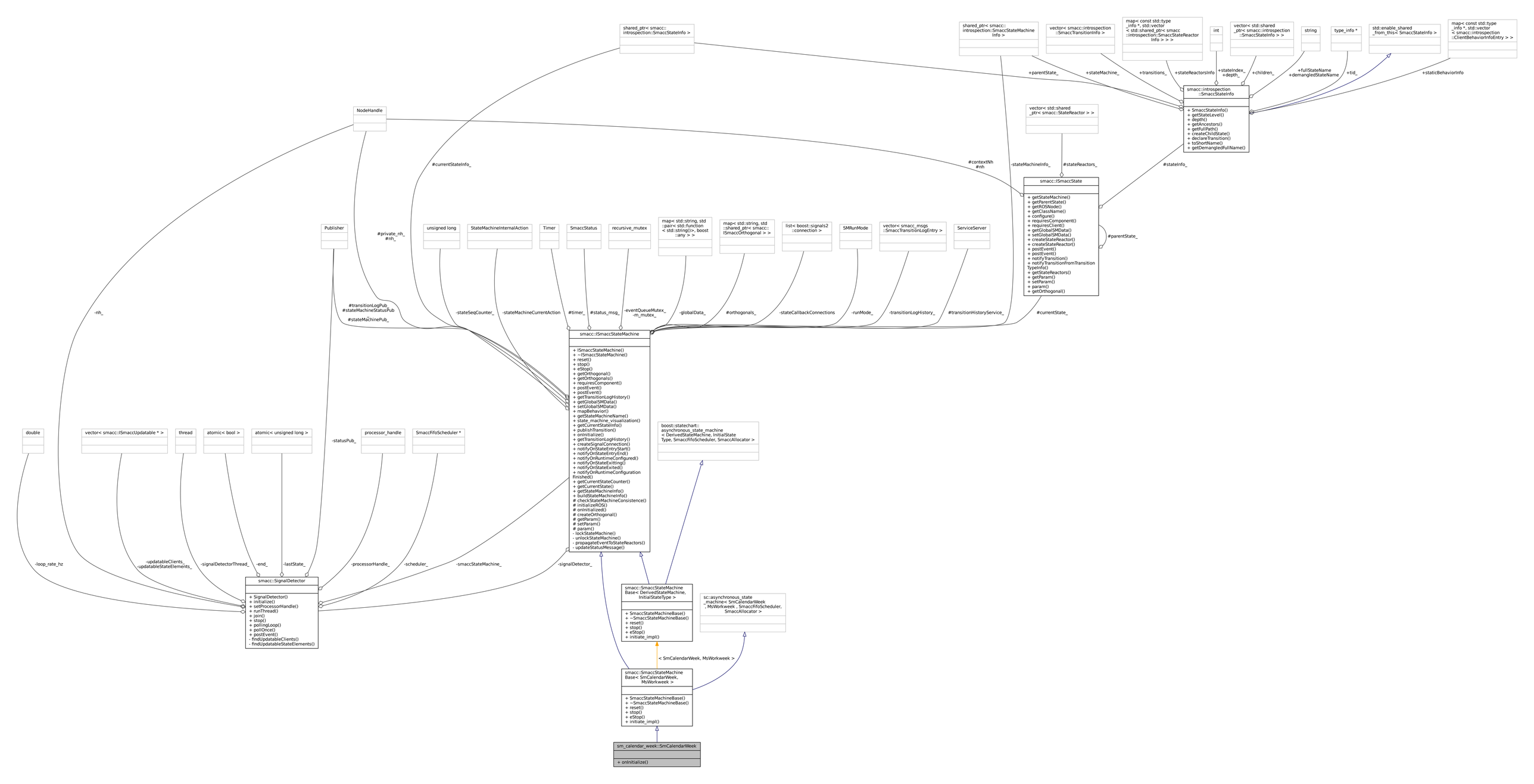
shared_ptr< smacc::introspection::SmaccStateInfo >
shared_ptr< smacc::introspection::SmaccStateMachineInfo >
vector< smacc::introspection::SmaccTransitionInfo >
map< const std::type_info *, std::vector< std::shared_ptr< smacc::introspection::SmaccStateReactorInfo > > >
int
vector< std::shared_ptr< smacc::introspection::SmaccStateInfo > >
string
type_info *
std::enable_shared_from_this< SmaccStateInfo >
map< const std::type_info *, std::vector< smacc::introspection::ClientBehaviorInfoEntry > >
+parentState_
+stateMachine_
+transitions_
+stateReactorsInfo
+stateIndex_
+depth_
+children_
+fullStateName
+demangledStateName
+tid_
+staticBehaviorInfo
smacc::introspection::SmaccStateInfo
+ SmaccStateInfo()
+ getStateLevel()
+ depth()
+ getAncestors()
+ getFullPath()
+ createChildState()
+ declareTransition()
+ toShortName()
+ getDemangledFullName()
NodeHandle
vector< std::shared_ptr< smacc::StateReactor > >
#currentStateInfo_
#contextNh
#nh
-stateMachineInfo_
#stateReactors_
#stateInfo_
smacc::ISmaccState
+ getStateMachine()
+ getParentState()
+ getROSNode()
+ getClassName()
+ configure()
+ requiresComponent()
+ requiresClient()
+ getGlobalSMData()
+ setGlobalSMData()
+ createStateReactor()
+ createStateReactor()
+ postEvent()
+ postEvent()
+ notifyTransition()
+ notifyTransitionFromTransitionTypeInfo()
+ getStateReactors()
+ getParam()
+ setParam()
+ param()
+ getOrthogonal()
#parentState_
#currentState_
-nh_
Publisher
#private_nh_
#nh_
unsigned long
StateMachineInternalAction
Timer
SmaccStatus
recursive_mutex
map< std::string, std::pair< std::function< std::string()>, boost::any > >
map< std::string, std::shared_ptr< smacc::ISmaccOrthogonal > >
list< boost::signals2::connection >
SMRunMode
vector< smacc_msgs::SmaccTransitionLogEntry >
ServiceServer
#transitionLogPub_
#stateMachineStatusPub
#stateMachinePub_
-stateSeqCounter_
-stateMachineCurrentAction
#timer_
#status_msg_
-eventQueueMutex_
-m_mutex_
-globalData_
#orthogonals_
-stateCallbackConnections
-runMode_
-transitionLogHistory_
#transitionHistoryService_
smacc::ISmaccStateMachine
+ ISmaccStateMachine()
+ ~ISmaccStateMachine()
+ reset()
+ stop()
+ eStop()
+ getOrthogonal()
+ getOrthogonals()
+ requiresComponent()
+ postEvent()
+ postEvent()
+ getTransitionLogHistory()
+ getGlobalSMData()
+ setGlobalSMData()
+ mapBehavior()
+ getStateMachineName()
+ state_machine_visualization()
+ getCurrentStateInfo()
+ publishTransition()
+ onInitialize()
+ getTransitionLonLogHistory()
+ createSignalConnection()
+ notifyOnStateEntryStart()
+ notifyOnStateEntryEnd()
+ notifyOnRuntimeConfigured()
+ notifyOnStateExitting()
+ notifyOnStateExited()
+ notifyOnRuntimeConfigurationFinished()
+ getCurrentStateCounter()
+ getCurrentState()
+ getStateMachineInfo()
+ buildStateMachineInfo()
# checkStateMachineConsistence()
# initializeROS()
# onInitialized()
# createOrthogonal()
# getParam()
# setParam()
# param()
- lockStateMachine()
- unlockStateMachine()
- propagateEventToStateReactors()
- updateStatusMessage()
double
vector< smacc::ISmaccUpdatable * >
thread
atomic< bool >
atomic< unsigned long >
-statusPub_
processor_handle
SmaccFifoScheduler *
boost::statechart::asynchronous_state_machine< DerivedStateMachine, InitialState Type, SmaccFifoScheduler, SmaccAllocator >
-loop_rate_hz
-updatableClients_
-updatableStateElements_
-signalDetectorThread_
-end_
-lastState_
-processorHandle_
-scheduler_
-smaccStateMachine_
-signalDetector_
smacc::SignalDetector
+ SignalDetector()
+ initialize()
+ setProcessorHandle()
+ runThread()
+ join()
+ stop()
+ pollingLoop()
+ pollOnce()
+ postEvent()
- findUpdatableClients()
- findUpdatableStateElements()
smacc::SmaccStateMachine Base< DerivedStateMachine, InitialStateType >
+ SmaccStateMachineBase()
+ ~SmaccStateMachineBase()
+ reset()
+ stop()
+ eStop()
+ initiate_impl()
sc::asynchronous_state_machine< SmCalendarWeek, MsWorkweek, SmaccFifoScheduler, SmaccAllocator >
< SmCalendarWeek, MsWorkweek >
smacc::SmaccStateMachine Base< SmCalendarWeek, MsWorkweek >
+ SmaccStateMachineBase()
+ ~SmaccStateMachineBase()
+ reset()
+ stop()
+ eStop()
+ initiate_impl()
sm_calendar_week::SmCalendarWeek
+ onInitialize()