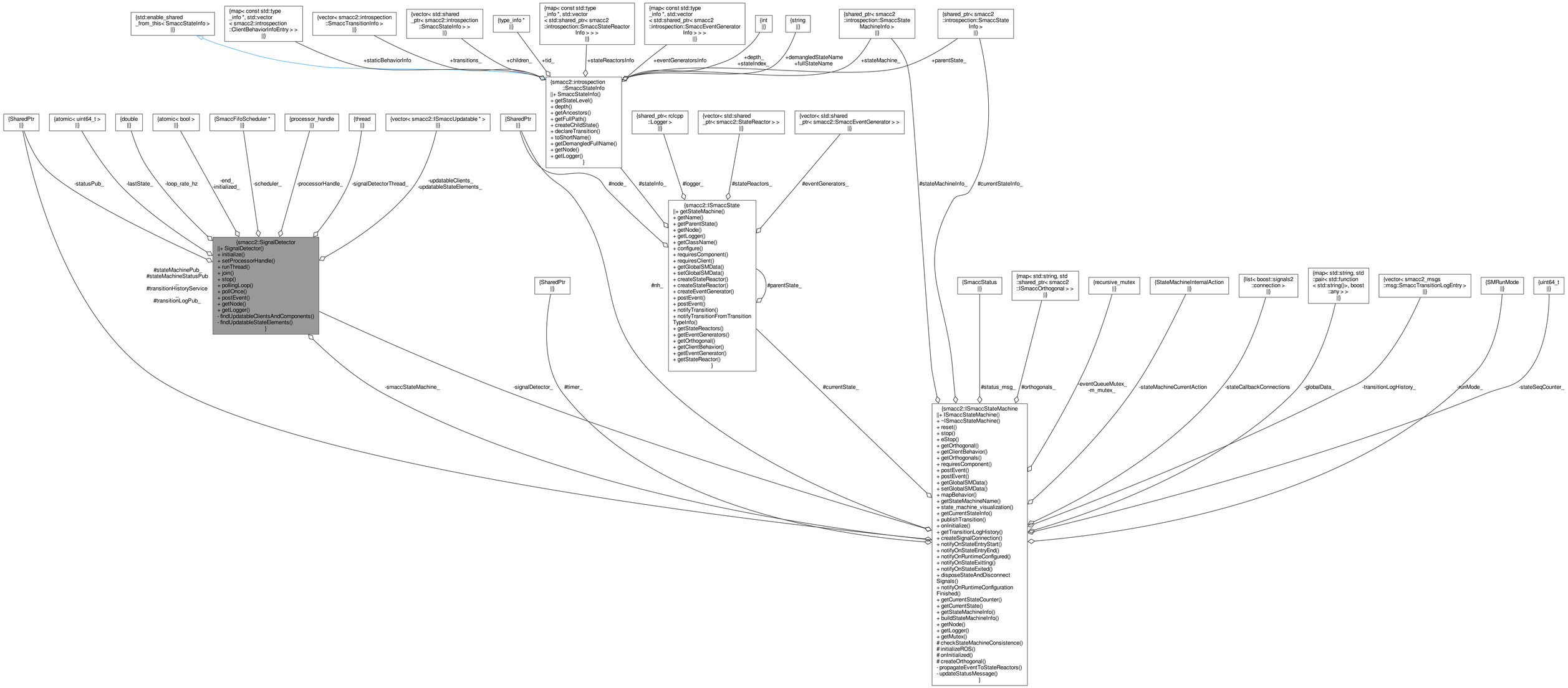
{std::enable_shared
_from_this< SmaccStateInfo >
|||
{map< const std::type
_info *, std::vector
< smacc2::introspection
::ClientBehaviorInfoEntry > >
|||
{vector< smacc2::introspection
::SmaccTransitionInfo >
|||
{vector< std::shared
_ptr< smacc2::introspection
::SmaccStateInfo > >
|||
{type_info *
|||
{map< const std::type
_info *, std::vector
< std::shared_ptr< smacc2
::introspection::SmaccStateReactor
Info > > >
|||
{map< const std::type
_info *, std::vector
< std::shared_ptr< smacc2
::introspection::SmaccEventGenerator
Info > > >
|||
{int
|||
{string
|||
{shared_ptr< smacc2
::introspection::SmaccState
MachineInfo >
|||
{shared_ptr< smacc2
::introspection::SmaccState
Info >
|||
+staticBehaviorInfo
+transitions_
+children_
+tid_
+stateReactorsInfo
+eventGeneratorsInfo
+depth_
+stateIndex_
+demangledStateName
+fullStateName
+stateMachine_
+parentState_
{smacc2::introspection
::SmaccStateInfo
||+ SmaccStateInfo()
+ getStateLevel()
+ depth()
+ getAncestors()
+ getFullPath()
+ createChildState()
+ declareTransition()
+ toShortName()
+ getDemangledFullName()
+ getNode()
+ getLogger()
}
{SharedPtr
|||
{atomic< uint64_t >
|||
{double
|||
{atomic< bool >
|||
{SmaccFifoScheduler *
|||
{processor_handle
|||
{thread
|||
{vector< smacc2::ISmaccUpdatable * >
|||
{SharedPtr
|||
{shared_ptr< rclcpp
::Logger >
|||
{vector< std::shared
_ptr< smacc2::StateReactor > >
|||
{vector< std::shared
_ptr< smacc2::SmaccEventGenerator > >
|||
-statusPub_
-lastState_
-loop_rate_hz
-end_
-initialized_
-scheduler_
-processorHandle_
-signalDetectorThread_
-updatableClients_
-updatableStateElements_
#node_
#stateInfo_
#logger_
#stateReactors_
#eventGenerators_
#stateMachineInfo_
#currentStateInfo_
{smacc2::SignalDetector
||+ SignalDetector()
+ initialize()
+ setProcessorHandle()
+ runThread()
+ join()
+ stop()
+ pollingLoop()
+ pollOnce()
+ postEvent()
+ getNode()
+ getLogger()
- findUpdatableClientsAndComponents()
- findUpdatableStateElements()
}
#stateMachinePub_
#stateMachineStatusPub
_
#transitionHistoryService
_
#transitionLogPub_
{smacc2::ISmaccState
||+ getStateMachine()
+ getName()
+ getParentState()
+ getNode()
+ getLogger()
+ getClassName()
+ configure()
+ requiresComponent()
+ requiresClient()
+ getGlobalSMData()
+ setGlobalSMData()
+ createStateReactor()
+ createStateReactor()
+ createEventGenerator()
+ postEvent()
+ postEvent()
+ notifyTransition()
+ notifyTransitionFromTransition
TypeInfo()
+ getStateReactors()
+ getEventGenerators()
+ getOrthogonal()
+ getClientBehavior()
+ getEventGenerator()
+ getStateReactor()
}
#nh_
#parentState_
{SharedPtr
|||
{SmaccStatus
|||
{map< std::string, std
::shared_ptr< smacc2
::ISmaccOrthogonal > >
|||
{recursive_mutex
|||
{StateMachineInternalAction
|||
{list< boost::signals2
::connection >
|||
{map< std::string, std
::pair< std::function
< std::string()>, boost
::any > >
|||
{vector< smacc2_msgs
::msg::SmaccTransitionLogEntry >
|||
{SMRunMode
|||
{uint64_t
|||
-smaccStateMachine_
-signalDetector_
#timer_
#currentState_
#status_msg_
#orthogonals_
-eventQueueMutex_
-m_mutex_
-stateMachineCurrentAction
-stateCallbackConnections
-globalData_
-transitionLogHistory_
-runMode_
-stateSeqCounter_
{smacc2::ISmaccStateMachine
||+ ISmaccStateMachine()
+ ~ISmaccStateMachine()
+ reset()
+ stop()
+ eStop()
+ getOrthogonal()
+ getClientBehavior()
+ getOrthogonals()
+ requiresComponent()
+ postEvent()
+ postEvent()
+ getGlobalSMData()
+ setGlobalSMData()
+ mapBehavior()
+ getStateMachineName()
+ state_machine_visualization()
+ getCurrentStateInfo()
+ publishTransition()
+ onInitialize()
+ getTransitionLogHistory()
+ createSignalConnection()
+ notifyOnStateEntryStart()
+ notifyOnStateEntryEnd()
+ notifyOnRuntimeConfigured()
+ notifyOnStateExitting()
+ notifyOnStateExited()
+ disposeStateAndDisconnect
Signals()
+ notifyOnRuntimeConfiguration
Finished()
+ getCurrentStateCounter()
+ getCurrentState()
+ getStateMachineInfo()
+ buildStateMachineInfo()
+ getNode()
+ getLogger()
+ getMutex()
# checkStateMachineConsistence()
# initializeROS()
# onInitialized()
# createOrthogonal()
- propagateEventToStateReactors()
- updateStatusMessage()
}