

# Flatbuffer

Tight Rope v0.65

17th February 2016

## 1 ID Files

### 1.1 MissionIds

**section** *MissionIds* **parents** *scj\_prelude*, *MissionId*

<i>FlatBufferMissionMID</i> : <i>MissionID</i>
--

<i>distinct</i> $\langle$ <i>nullMissionId</i> , <i>FlatBufferMissionMID</i> $\rangle$
--

## 1.2 SchedulablesIds

**section** *SchedulableIds* **parents** *scj\_prelude*, *SchedulableId*

*FlatBufferMissionSequencerSID* : *SchedulableID*

*ReaderSID* : *SchedulableID*

*WriterSID* : *SchedulableID*

*distinct*(*nullSequencerId*, *nullSchedulableId*, *FlatBufferMissionSequencerSID*,  
*ReaderSID*, *WriterSID*)

### 1.3 ThreadIds

**section** *ThreadId* **parents** *scj\_prelude, GlobalTypes*

*WriterTID* : *ThreadID*

*ReaderTID* : *ThreadID*

---

*distinct*(*SafeletTID*, *nullTID*,  
*WriterTID*, *ReaderTID*)

## 1.4 ObjectIds

**section** *ObjectIds* **parents** *scj\_prelude, GlobalTypes*

$FlatBufferMissionOID : ObjectID$
-----------------------------------

$distinct \langle FlatBufferMissionOID \rangle$
---

## 2 Network

### 2.1 Network Channel Sets

**section** *NetworkChannels* **parents** *scj\_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableChan, TopLevelMissionSequencerFWChan, FrameworkChan, SafeletChan*

**channelset** *TerminateSync* ==  
    { *schedulables\_terminated, schedulables\_stopped, get\_activeSchedulables* }

**channelset** *ControlTierSync* ==  
    { *start\_toplevel\_sequencer, done\_toplevel\_sequencer, done\_safeletFW* }

**channelset** *TierSync* ==  
    { *start\_mission.FlatBufferMission, done\_mission.FlatBufferMission, done\_safeletFW, done\_toplevel\_sequencer* }

**channelset** *MissionSync* ==  
    { *done\_safeletFW, done\_toplevel\_sequencer, register, signalTerminationCall, signalTerminationRet, activate\_schedulables, done\_schedulable, cleanupSchedulableCall, cleanupSchedulableRet* }

**channelset** *SchedulablesSync* ==  
    { *activate\_schedulables, done\_safeletFW, done\_toplevel\_sequencer* }

**channelset** *ClusterSync* ==  
    { *done\_toplevel\_sequencer, done\_safeletFW* }

**channelset** *AppSync* ==  
    { *SafeltAppSync, MissionSequencerAppSync, MissionAppSync, MTAAppSync, OSEHSync, APEHSync, getSequencer, end\_mission\_app, end\_managedThread\_app, setCeilingPriority, requestTerminationCall, requestTerminationRet, terminationPendingCall, terminationPendingRet, handleAsyncEventCall, handleAsyncEventRet* }

**channelset** *ThreadSync* ==  
    { *raise\_thread\_priority, lower\_thread\_priority, isInterruptedCall, isInterruptedRet, get\_priorityLevel* }

**channelset** *LockingSync* ==  
    { *lockAcquired, startSyncMeth, endSyncMeth, waitCall, waitRet, notify, isInterruptedCall, isInterruptedRet, interruptedCall, interruptedRet, done\_toplevel\_sequencer, get\_priorityLevel* }

## 2.2 MethodCallBinder

**section** *MethodCallBindingChannels* **parents** *scj\_prelude*, *GlobalTypes*, *FrameworkChan*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *ThreadId*

**channel** *binder\_readCall* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  
**channel** *binder\_readRet* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  $\times$   $\mathbb{Z}$

*readLocs* == { *FlatBufferMissionMID* }  
*readCallers* == { *ReaderSID* }

**channel** *binder\_writeCall* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  $\times$   $\mathbb{Z}$   
**channel** *binder\_writeRet* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*

*writeLocs* == { *FlatBufferMissionMID* }  
*writeCallers* == { *WriterSID* }

**channelset** *MethodCallBinderSync* == { *done\_toplevel\_sequencer*,  
*binder\_readCall*, *binder\_readRet*,  
*binder\_writeCall*, *binder\_writeRet* }

**section** *MethodCallBinder* **parents** *scj\_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*

**process** *MethodCallBinder*  $\hat{=}$  **begin**

*read\_MethodBinder*  $\hat{=}$   

$$\left( \begin{array}{l} \text{binder\_readCall} ? \text{loc} : (\text{loc} \in \text{readLocs}) ? \text{caller} : (\text{caller} \in \text{readCallers}) ? \text{callingThread} \longrightarrow \\ \text{readCall} . \text{loc} . \text{caller} . \text{callingThread} \longrightarrow \\ \text{readRet} . \text{loc} . \text{caller} . \text{callingThread} ? \text{ret} \longrightarrow \\ \text{binder\_readRet} . \text{loc} . \text{caller} . \text{callingThread} ! \text{ret} \longrightarrow \\ \text{read\_MethodBinder} \end{array} \right)$$

*write\_MethodBinder*  $\hat{=}$   

$$\left( \begin{array}{l} \text{binder\_writeCall} ? \text{loc} : (\text{loc} \in \text{writeLocs}) ? \text{caller} : (\text{caller} \in \text{writeCallers}) ? p1 ? \text{callingThread} \longrightarrow \\ \text{writeCall} . \text{loc} . \text{caller} . \text{callingThread} ! p1 \longrightarrow \\ \text{writeRet} . \text{loc} . \text{caller} . \text{callingThread} \longrightarrow \\ \text{binder\_writeRet} . \text{loc} . \text{caller} . \text{callingThread} \longrightarrow \\ \text{write\_MethodBinder} \end{array} \right)$$

*BinderActions*  $\hat{=}$   

$$\left( \begin{array}{l} \text{read\_MethodBinder} \\ \parallel \\ \text{write\_MethodBinder} \end{array} \right)$$

• *BinderActions*  $\triangle$  (*done\_toplevel\_sequencer*  $\longrightarrow$  **Skip**)

**end**

**process** *ApplicationB*  $\hat{=}$  *Application* [ *MethodCallBinderSync* ] *MethodCallBinder*

## 2.3 Locking

**process** *Threads*  $\hat{=}$   
 $\left( \begin{array}{c} \textit{ThreadFW}(\textit{WriterTID}, 10) \\ ||| \\ \textit{ThreadFW}(\textit{ReaderTID}, 10) \end{array} \right)$

**process** *Objects*  $\hat{=}$   
 $(\textit{ObjectFW}(\textit{FlatBufferMissionOID}))$

**process** *Locking*  $\hat{=}$  *Threads*  $\llbracket$  *ThreadSync*  $\rrbracket$  *Objects*

## 2.4 Program

**section** *Program* **parents** *scj\_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, MissionFW, SafeletFW, TopLevelMissionSequencerFW, NetworkChannels, ManagedThreadFW, SchedulableMissionSequencerFW, PeriodicEventHandlerFW, OneShotEventHandlerFW, AperiodicEventHandlerFW, ObjectFW, ThreadFW, FlatBufferApp, FlatBufferMissionSequencerApp, FlatBufferMissionApp, ReaderApp, WriterApp*

**process** *ControlTier*  $\triangleq$   

$$\left( \begin{array}{l} \text{SafeletFW} \\ \llbracket \text{ControlTierSync} \rrbracket \\ \text{TopLevelMissionSequencerFW}(\text{FlatBufferMissionSequencer}) \end{array} \right)$$

**process** *Tier0*  $\triangleq$   

$$\left( \begin{array}{l} \text{MissionFW}(\text{FlatBufferMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left( \begin{array}{l} \text{ManagedThreadFW}(\text{ReaderID}) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{ManagedThreadFW}(\text{WriterID}) \end{array} \right) \end{array} \right)$$

**process** *Framework*  $\triangleq$   

$$\left( \begin{array}{l} \text{ControlTier} \\ \llbracket \text{TierSync} \rrbracket \\ (\text{Tier0}) \end{array} \right)$$

**process** *Application*  $\triangleq$   

$$\left( \begin{array}{l} \text{FlatBufferApp} \\ ||| \\ \text{FlatBufferMissionSequencerApp} \\ ||| \\ \text{FlatBufferMissionApp} \\ ||| \\ \text{ReaderApp}(\text{FlatBufferMissionID}) \\ ||| \\ \text{WriterApp}(\text{FlatBufferMissionID}) \end{array} \right)$$

**process** *Program*  $\triangleq (\text{Framework} \llbracket \text{AppSync} \rrbracket \text{ApplicationB}) \llbracket \text{LockingSync} \rrbracket \text{Locking}$



### 3 Safelet

**section** *FlatBufferApp* **parents** *scj\_prelude, SchedulableId, SchedulableIds, SafeletChan, MethodCallBindingChannels*

**process** *FlatBufferApp*  $\hat{=}$  **begin**

*InitializeApplication*  $\hat{=}$   
 $\left( \begin{array}{l} \textit{initializeApplicationCall} \longrightarrow \\ \textit{initializeApplicationRet} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

*GetSequencer*  $\hat{=}$   
 $\left( \begin{array}{l} \textit{getSequencerCall} \longrightarrow \\ \textit{getSequencerRet} ! \textit{FlatBufferMissionSequencerSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

*Methods*  $\hat{=}$   
 $\left( \begin{array}{l} \textit{GetSequencer} \\ \square \\ \textit{InitializeApplication} \end{array} \right); \textit{Methods}$

•  $(\textit{Methods}) \triangle (\textit{end\_safelet\_app} \longrightarrow \mathbf{Skip})$

**end**

## 4 Top Level Mission Sequencer

**section** *FlatBufferMissionSequencerApp* **parents** *TopLevelMissionSequencerChan*,  
*MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *FlatBufferMissionSequencerClass*, *MethodCallBindingChannels*

**process** *FlatBufferMissionSequencerApp*  $\hat{=}$  **begin**

<i>State</i> <i>this</i> : <b>ref</b> <i>FlatBufferMissionSequencerClass</i>
---

**state** *State*

<i>Init</i> <i>State'</i>
<i>this'</i> = <b>new</b> <i>FlatBufferMissionSequencerClass</i> ()

*GetNextMission*  $\hat{=}$  **var** *ret* : *MissionID* •  
 $\left( \begin{array}{l} \textit{getNextMissionCall} . \textit{FlatBufferMissionSequencerSID} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getNextMission}(); \\ \textit{getNextMissionRet} . \textit{FlatBufferMissionSequencerSID} ! \textit{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

*Methods*  $\hat{=}$   
 $(\textit{GetNextMission}) ; \textit{Methods}$

•  $(\textit{Init} ; \textit{Methods}) \triangle (\textit{end\_sequencer\_app} . \textit{FlatBufferMissionSequencerSID} \longrightarrow \mathbf{Skip})$

**end**

**section** *FlatBufferMissionSequencerClass* **parents** *scj\_prelude, SchedulableId, SchedulableIds, SafeletChannels, MethodCallBindingChannels, MissionId, MissionIds*

**class** *FlatBufferMissionSequencerClass*  $\hat{=}$  **begin**

<b>state</b> <i>State</i> <i>returnedMission</i> : $\mathbb{B}$
--

**state** *State*

<b>initial</b> <i>Init</i> <i>State</i> '
<i>returnedMission</i> ' = <b>False</b>

**protected** *getNextMission*  $\hat{=}$  **var** *ret* : *MissionID* •

$$\left( \begin{array}{l} \text{if } (\neg \text{returnedMission} = \mathbf{True}) \longrightarrow \\ \quad (\text{ret} := \text{FlatBufferMissionMID}) \\ \parallel \neg (\neg \text{returnedMission} = \mathbf{True}) \longrightarrow \\ \quad (\text{ret} := \text{nullMissionId}) \\ \text{fi} \end{array} \right)$$

• **Skip**

**end**

## 5 Missions

### 5.1 FlatBufferMission

**section** *FlatBufferMissionApp* **parents** *scj\_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, FlatBufferMissionMethChan, FlatBufferMissionClass, MethodCallBindingChannels*

**process** *FlatBufferMissionApp*  $\hat{=}$  **begin**

<i>State</i> <i>this</i> : <b>ref</b> <i>FlatBufferMissionClass</i>
--

**state** *State*

<i>Init</i> <i>State'</i> <i>this'</i> = <b>new</b> <i>FlatBufferMissionClass</i> ()
--

*InitializePhase*  $\hat{=}$

$$\left( \begin{array}{l} \text{initializeCall} . \text{FlatBufferMissionMID} \longrightarrow \\ \text{register} ! \text{ReaderSID} ! \text{FlatBufferMissionMID} \longrightarrow \\ \text{register} ! \text{WriterSID} ! \text{FlatBufferMissionMID} \longrightarrow \\ \text{initializeRet} . \text{FlatBufferMissionMID} \longrightarrow \\ \text{Skip} \end{array} \right)$$

*CleanupPhase*  $\hat{=}$

$$\left( \begin{array}{l} \text{cleanupMissionCall} . \text{FlatBufferMissionMID} \longrightarrow \\ \text{cleanupMissionRet} . \text{FlatBufferMissionMID} ! \text{True} \longrightarrow \\ \text{Skip} \end{array} \right)$$

*bufferEmptyMeth*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •

$$\left( \begin{array}{l} \text{bufferEmptyCall} . \text{FlatBufferMissionMID} \longrightarrow \\ \text{ret} := \text{this} . \text{bufferEmpty}(); \\ \text{bufferEmptyRet} . \text{FlatBufferMissionMID} ! \text{ret} \longrightarrow \\ \text{Skip} \end{array} \right)$$

*cleanUpMeth*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •

$$\left( \begin{array}{l} \text{cleanUpCall} . \text{FlatBufferMissionMID} \longrightarrow \\ \text{ret} := \text{this} . \text{cleanUp}(); \\ \text{cleanUpRet} . \text{FlatBufferMissionMID} ! \text{ret} \longrightarrow \\ \text{Skip} \end{array} \right)$$

*writeSyncMeth*  $\hat{=}$

$$\left( \begin{array}{l} \text{writeCall} . \text{FlatBufferMissionMID} ? \text{caller} ? \text{thread} ? \text{update} \longrightarrow \\ \left( \begin{array}{l} \text{startSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{FlatBufferMissionOID} . \text{thread} \longrightarrow \\ \left( \begin{array}{l} \mu X \bullet \\ \left( \begin{array}{l} \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{bufferEmpty}()); \\ \text{if } (\text{loopVar} = \text{True}) \longrightarrow \\ \text{Skip} ; X \\ \square (\text{loopVar} = \text{False}) \longrightarrow \text{Skip} \end{array} \right) \\ \text{fi} \end{array} \right) \\ \text{endSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \longrightarrow \\ \text{writeRet} . \text{FlatBufferMissionMID} . \text{caller} . \text{thread} \longrightarrow \\ \text{Skip} \end{array} \right) \end{array} \right)$$

$$\begin{aligned}
& \text{readSyncMeth} \hat{=} \mathbf{var} \text{ ret} : \mathbb{Z} \bullet \\
& \left( \begin{array}{l}
\text{readCall} . \text{FlatBufferMissionMID} ? \text{caller} ? \text{thread} \longrightarrow \\
\left( \begin{array}{l}
\text{startSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \longrightarrow \\
\text{lockAcquired} . \text{FlatBufferMissionOID} . \text{thread} \longrightarrow \\
\left( \begin{array}{l}
\mu X \bullet \\
\left( \begin{array}{l}
\mathbf{var} \text{ loopVar} : \mathbb{B} \bullet \text{loopVar} := \text{bufferEmpty}(); \\
\mathbf{if} (\text{loopVar} = \mathbf{True}) \longrightarrow \\
\mathbf{Skip} ; X \\
\parallel (\text{loopVar} = \mathbf{False}) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array} \right) \\
; \\
\mathbf{var} \text{ out} : \mathbb{Z} \bullet \text{out} := \text{this} . \text{buffer}; \\
\text{ret} := \text{out}
\end{array} \right) \\
\text{endSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \longrightarrow \\
\text{readRet} . \text{FlatBufferMissionMID} . \text{caller} . \text{thread} ! \text{ret} \longrightarrow \\
\mathbf{Skip}
\end{array} \right) ;
\end{array} \right)
\end{aligned}$$

$$\begin{aligned}
& \text{Methods} \hat{=} \left( \begin{array}{l}
\text{InitializePhase} \\
\Box \\
\text{CleanupPhase} \\
\Box \\
\text{bufferEmptyMeth} \\
\Box \\
\text{cleanUpMeth} \\
\Box \\
\text{writeSyncMeth} \\
\Box \\
\text{readSyncMeth}
\end{array} \right) ; \text{Methods}
\end{aligned}$$

$$\bullet (\text{Init} ; \text{Methods}) \triangle (\text{end\_mission\_app} . \text{FlatBufferMissionMID} \longrightarrow \mathbf{Skip})$$

**end**

**section** *FlatBufferMissionClass* **parents** *scj\_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingChannels*

**class** *FlatBufferMissionClass*  $\hat{=}$  **begin**

<b>state</b> <i>State</i> <i>buffer</i> : $\mathbb{Z}$ <i>t</i> : <i>testClass</i>
--

**state** *State*

<b>initial</b> <i>Init</i> <i>State</i> '
<i>buffer</i> ' = 0 <i>t</i> ' = <i>testClass</i>

**public** *bufferEmpty*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •  

$$\left( \begin{array}{l} \text{if } (buffer = 0) \longrightarrow \\ \quad ret := \mathbf{True} \\ \square \neg (buffer = 0) \longrightarrow \\ \quad ret := \mathbf{False} \\ \text{fi} \end{array} \right)$$

**public** *cleanUp*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •  
(*ret* := **False**)

• **Skip**

**end**

**section** *FlatBufferMissionMethChan* **parents** *scj\_prelude, GlobalTypes, MissionId, SchedulableId*

**channel** *bufferEmptyCall* : *MissionID*  
**channel** *bufferEmptyRet* : *MissionID*  $\times$   $\mathbb{B}$

**channel** *cleanUpCall* : *MissionID*  
**channel** *cleanUpRet* : *MissionID*  $\times$   $\mathbb{B}$

**channel** *writeCall* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  $\times$   $\mathbb{Z}$   
**channel** *writeRet* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*

**channel** *readCall* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  
**channel** *readRet* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  $\times$   $\mathbb{Z}$

## 5.2 Schedulables of FlatBufferMission

**section** *ReaderApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*, *MissionMethChan*, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

**process** *ReaderApp*  $\hat{=}$   
*fbMission* : *MissionID* • **begin**

*Run*  $\hat{=}$   

$$\left( \begin{array}{l} \text{runCall} . \text{ReaderSID} \longrightarrow \\ \left( \begin{array}{l} \mu X \bullet \\ \left( \begin{array}{l} \text{terminationPendingCall} . \text{fbMission} \longrightarrow \\ \text{terminationPendingRet} . \text{fbMission} ? \text{terminationPending} \longrightarrow \\ \text{var } \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar} = \text{True}) \longrightarrow \\ \left( \begin{array}{l} \text{var } \text{result} : \mathbb{Z} \bullet \text{result} := 0; \\ \left( \begin{array}{l} \text{binder\_readCall} . \text{fbMission} . \text{ReaderSID} . \text{ReaderTID} \longrightarrow \\ \text{binder\_readRet} . \text{fbMission} . \text{ReaderSID} . \text{ReaderTID} ? \text{read} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \end{array} \right) \\ \parallel (\text{loopVar} = \text{False}) \longrightarrow \text{Skip} \end{array} \right) \\ \text{fi} \end{array} \right) \\ \text{runRet} . \text{ReaderSID} \longrightarrow \\ \text{Skip} \end{array} \right) \end{array} \right)$$

*Methods*  $\hat{=}$   
 $(\text{Run}) ; \text{Methods}$

•  $(\text{Methods}) \triangle (\text{end\_managedThread\_app} . \text{ReaderSID} \longrightarrow \text{Skip})$

**end**



**section** *WriterApp* **parents** *ManagedThreadChan, SchedulableId, SchedulableIds, MethodCallBindingChannels, MissionMethChan, FlatBufferMissionMethChan, ObjectIds, ThreadIds*

**process** *WriterApp*  $\hat{=}$   
*fbMission : MissionID* • **begin**

*Run*  $\hat{=}$

$$\left( \begin{array}{l} \text{runCall} . \text{WriterSID} \longrightarrow \\ \left( \begin{array}{l} \text{var } i : \mathbb{Z} \bullet i := 1; \\ \mu X \bullet \\ \left( \begin{array}{l} \text{terminationPendingCall} . \text{fbMission} \longrightarrow \\ \text{terminationPendingRet} . \text{fbMission} ? \text{terminationPending} \longrightarrow \\ \text{var } \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar} = \text{True}) \longrightarrow \\ \left( \begin{array}{l} \text{binder\_writeCall} . \text{fbMission} . \text{WriterSID} . \text{WriterTID} ! i \longrightarrow \\ \text{binder\_writeRet} . \text{fbMission} . \text{WriterSID} . \text{WriterTID} \longrightarrow \\ \text{Skip}; \\ i := i + 1; \\ \text{if } (i \geq 5) \longrightarrow \\ \left( \begin{array}{l} \text{requestTerminationCall} . \text{fbMission} . \text{WriterSID} \longrightarrow \\ \text{requestTerminationRet} . \text{fbMission} . \text{WriterSID} ? \text{requestTermination} \longrightarrow \end{array} \right) \\ \text{Skip} \\ \parallel \neg (i \geq 5) \longrightarrow \text{Skip} \end{array} \right) \\ \text{fi} \\ \parallel (\text{loopVar} = \text{False}) \longrightarrow \text{Skip} \end{array} \right) \\ \text{fi} \end{array} \right) \end{array} \right) ; X \end{array} \right)$$

*Methods*  $\hat{=}$   
(*Run*) ; *Methods*

• (*Methods*)  $\triangle$  (*end\_managedThread\_app* . *WriterSID*  $\longrightarrow$  **Skip**)

**end**