FairMQ

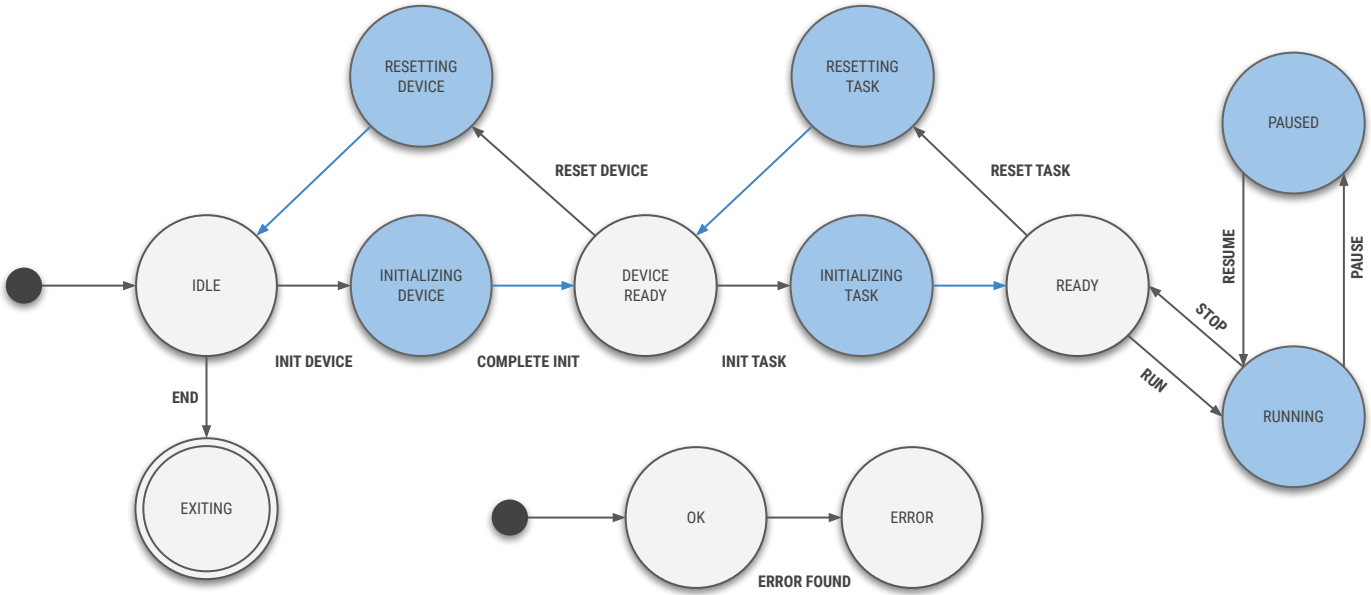
State Machine Update & Other Developments

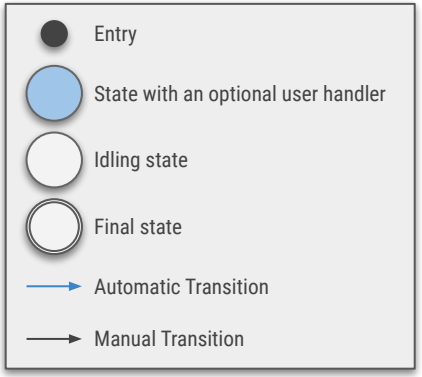
Alexey Rybalchenko
GSI Darmstadt, Software Development for Experiments

ALICE Software and Computing Week
CERN, April 5, 2019

FairMQ State Machine

Based on Boost Meta State Machine (MSM).





States:

FairMQStateMachine::State::**IDLE**
FairMQStateMachine::State::**INITIALIZING_DEVICE**
FairMQStateMachine::State::**DEVICE_READY**
FairMQStateMachine::State::**INITIALIZING_TASK**
FairMQStateMachine::State::**READY**
FairMQStateMachine::State::**RUNNING**
FairMQStateMachine::State::**PAUSED**
FairMQStateMachine::State::**RESETTING_TASK**
FairMQStateMachine::State::**RESETTING_DEVICE**
FairMQStateMachine::State::**PAUSED**
FairMQStateMachine::State::**EXITING**
FairMQStateMachine::State::**OK**
FairMQStateMachine::State::**ERROR**

Transitions:

FairMQStateMachine::Event::**INIT_DEVICE**
FairMQStateMachine::Event::**INIT_TASK**
FairMQStateMachine::Event::**RUN**
FairMQStateMachine::Event::**STOP**
FairMQStateMachine::Event::**RESET_TASK**
FairMQStateMachine::Event::**RESET_DEVICE**
FairMQStateMachine::Event::**END**
FairMQStateMachine::Event::**ERROR_FOUND**

Issues:

- **Channels can only be created before state machine runs (before InitializingDevice state).**
- **A user state handler could hang while the state machine has already transitioned to the next state.**
- **FairMQDevice and FairMQStateMachine are too tightly coupled.**

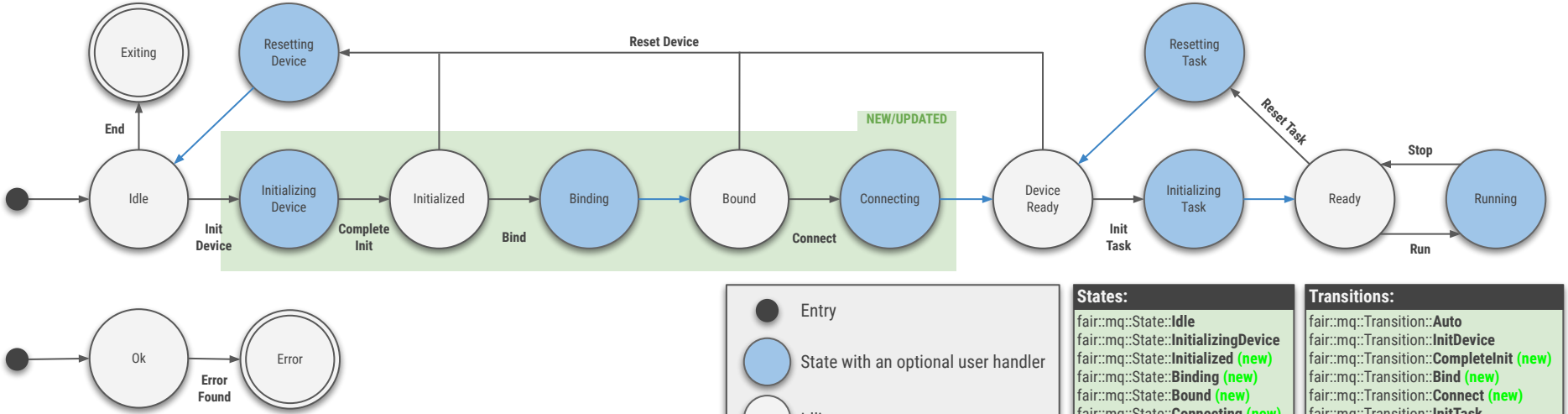
FairMQ State Machine Updates (1)

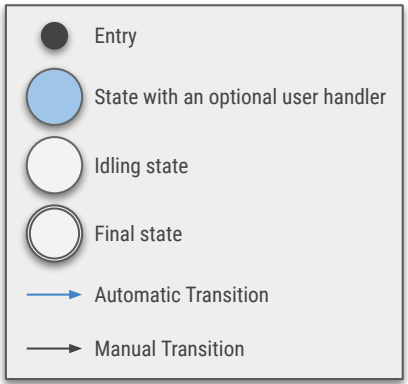
Split InitializingDevice state into InitializingDevice + Binding + Connecting:

- **InitializingDevice state for config plugin to apply any kind of config and/or create channels.**
- **Binding & Connecting states to perform dynamic port configuration.**

Removed Paused state:

- **Did the same as going from Running to Ready and back.**





States:

fair::mq::State::**Idle**
fair::mq::State::**InitializingDevice**
fair::mq::State::**Initialized** (new)
fair::mq::State::**Binding** (new)
fair::mq::State::**Bound** (new)
fair::mq::State::**Connecting** (new)
fair::mq::State::**DeviceReady**
fair::mq::State::**InitializingTask**
fair::mq::State::**Ready**
fair::mq::State::**Running**
fair::mq::State::**ResettingTask**
fair::mq::State::**ResettingDevice**
fair::mq::State::**Exiting**
fair::mq::State::**Ok**
fair::mq::State::**Error**

Transitions:

fair::mq::Transition::**Auto**
fair::mq::Transition::**InitDevice**
fair::mq::Transition::**CompleteInit** (new)
fair::mq::Transition::**Bind** (new)
fair::mq::Transition::**Connect** (new)
fair::mq::Transition::**InitTask**
fair::mq::Transition::**Run**
fair::mq::Transition::**Stop**
fair::mq::Transition::**ResetTask**
fair::mq::Transition::**ResetDevice**
fair::mq::Transition::**End**
fair::mq::Transition::**ErrorFound**

FairMQ State Machine Updates (2)

- **Previously a user state handler could hang while the state machine has already transitioned to the next state.**

  E.g.: Controller initiates STOP transition to change the state from RUNNING to READY. Handler for the RUNNING state (Run() method) could ignore this and/or hang for any reason.

  → Previously the controller would still see the state as READY and only fail on following transitions, which is misleading.

  → Now the transition to the new state happens only once the handler for the previous completes.

  The check that the handler has to make is no longer `CheckCurrentState(RUNNING)`, but `NewStatePending()`. Old one still works, but is deprecated.

- **Deprecate `WaitForEndOfState(transition)` interface in favor of `WaitForState(state)/WaitForNextState()`. Only relevant for custom main() without any control plugin.**

- **Minor modernization: converted state & transition names from enums to enum classes and changed their names from UPPERCASE to CamelCase to avoid confusion with MACROs.**
  - **Breaking change for custom `main()` that doesn't use our plugins.**
  - **Backwards-compatibility provided for `FairMQDevice::ChangeState()` and `FairMQDevice::CheckCurrentState()`.**

Upgrade Instructions

DPL (or its users) and/or any device users that use our supplied config & control plugins

- No breaking changes (relevant ones).

- Replace the deprecated methods (will remain in FairMQ for at least two stable releases):

```
-    CheckCurrentState(RUNNING)
+    NewStatePending()
```

- Remove use of Pause() (unused as far as we can see).

Control plugin writers (PluginServices API)

Add transitions for new states, e.g.:

```
  ChangeDeviceState(DeviceStateTransition::InitDevice);
+ while (WaitForNextState() != DeviceState::InitializingDevice) {}

+ // apply potential config changes here ...

+ ChangeDeviceState(DeviceStateTransition::CompleteInit);
+ while (WaitForNextState() != DeviceState::Initialized) {}
+ ChangeDeviceState(DeviceStateTransition::Bind);
+ while (WaitForNextState() != DeviceState::Bound) {}

+ // read bound channel properties here ...

+ ChangeDeviceState(DeviceStateTransition::Connect);
  while (WaitForNextState() != DeviceState::DeviceReady) {}
  ChangeDeviceState(DeviceStateTransition::InitTask);
  while (WaitForNextState() != DeviceState::Ready) {}
  ChangeDeviceState(DeviceStateTransition::Run);
  while (WaitForNextState() != DeviceState::Running) {}
```

Old code compiles, but will not transition beyond InitializingDevice().

Users with custom main() doing own state control

- Update to new enum class types.

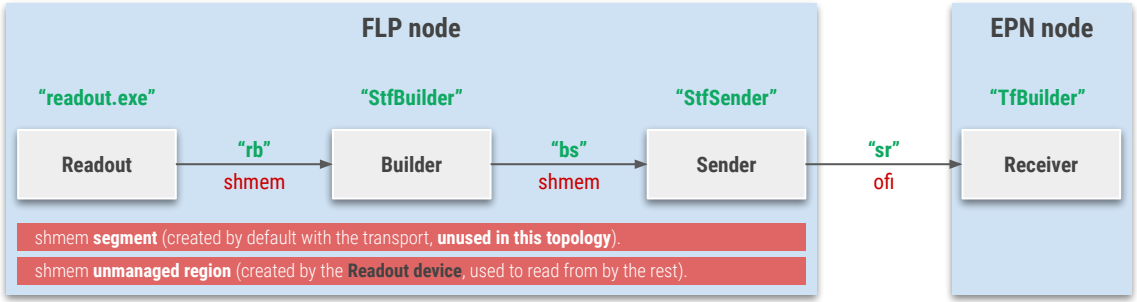
- Replace use of WaitForEndOfState(transition) with WaitForState(state).

- Add transitions for new states (see middle column).

New Example: “Readout”

Includes 2 topologies:

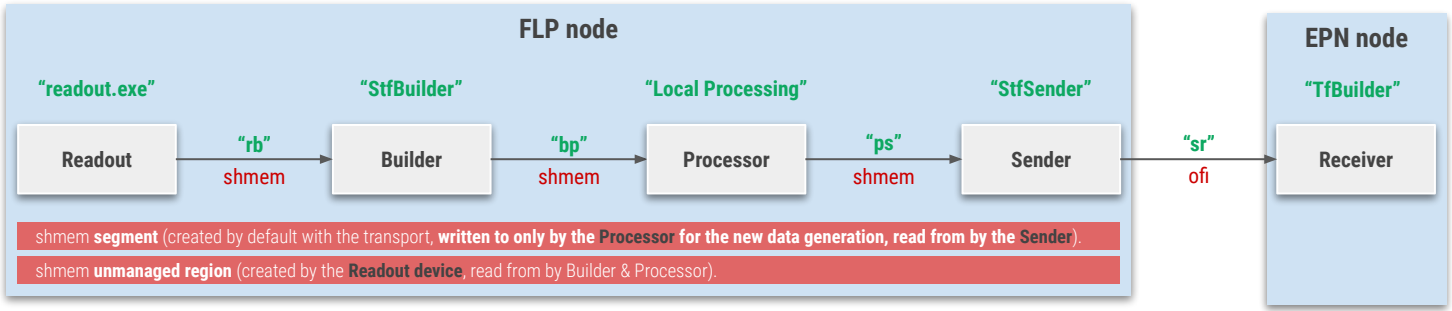
(1) A simpler one, where the data created by the Readout is the same that is sent out to EPN.



A playground/mockup for us to quickly test our developments in a scenario similar to FLP->EPN data flow.

Involves only transfers of dummy data and potential copies/copy avoidance.

(2) Same as above, but the Processor creates a new generation of data that is put in the general shared memory segment (not the unmanaged region used by the readout).



https://github.com/FairRootGroup/FairMQ/tree/dev/examples/readout

FairMQ: Further Minor Updates

Added cmd device option `--max-run-time <num seconds>`, which, if greater than 0 (default), will change state to Ready after Running has been going for the specified number of seconds.

Many improvements to our test suite and tools:

- **execute child processes via boost::process+boost::asio,**
- **improve process tools (support input and/or signaling to the process),**
- **add many more tests to cover exceptional cases (error states, exceptions, signal handling),**
- **improve the output of the tests.**

Improved help output for interactive controller plugin.