Elara Application Types (Overview)

		⇶	⇉|→	—□→	—□	⋘
		Input Sourcing	Command Sequencing	Command Processing	Event Application	Output Distribution
InputApp	→	y				
SequencerApp	⇉|→	(o)	y			
ProcessorApp	—□→			y	y	
OutputApp	→					y
TimerApp	□→	y				
SequencerProcessorApp	⇉|—□→	(o)	y	y	y	
AllInOneApp		(o)	y	y	y	y
FeedbackApp	—□→	y			y	
PlaybackApp	—|					y
ReplicationApp	|→			(p)		
FollowerApp	|—□				y	
Legend						
Non-deterministic App						
Deterministic App						
Deterministic Replay App						
→ *Command flow*						
→ *Event flow*						
y *Involved*						
(o) *Optionally involved*						
(p) *Partially involved*						

Elara Functional Modules

Input Sourcing			
	-	Connecting to any source of input messages for instance by subscribing to an upstream system	
	-	Includes adaptation of transport and codec if necessary and may perform filtering of undesired messages	
	-	Source can also be an abstract input source such as a wall clock time source or a source of random numbers	
	-	Typically non-deterministic since source of messages is usually not deterministic	
	-	Uses *Command Sender* to send input message to the sequencer as a command	
Command Sending			
	-	Assigns source ID and sequence to a message and sends the message — now called command — to the sequencer	
	-	Commands can originate from different situations, but mostly from inputs or in reaction to events	
Command Sequencing			
	-	Sequences commands from multiple streams into a defined sequence of commands	
	-	Non-deterministic since multiple input sources are connected and racing can occur between the sources	
Command Processing			
	-	Processes commands by routing events	
	-	Has read-only access to application state	
Event Routing			
	-	Process of creating an event in the command processor	
	-	When the event is complete it is routed, which means it is stored in the event store and typically also immediately applied to the processor's application state, and later possibly also picked for output distribution	
Event Applying			
	-	Modification of application state by applying events	
	-	Has read/write access to application state	
	-	All information required to modify the state must be contained in the event	
Output Distribution			
	-	Publication of messages to downstream applications based on events	
	-	Performs protocol and codec transformation if necessary	
	-	Publication can also mean persistence to file or database or any other means of processing of event data targeting external formats or systems	
	-	Replayed events are flagged as such to prevent double-publication of events; events can be committed or marked as retry e.g. if the downstream system is currently unavailable	

Elara Application Types (Descriptions)

InputApp		
	-	Subscribes to an input source such as an upstream application
	-	Adapts transport and codec from source system to sequencer app
	-	Assigns source ID and source sequence and sends commands to sequencer
	-	Multiple active subscriber apps can be present in a running Elara multi-application installation
SequencerApp		
	-	Sequences multiple command streams into a single sequenced stream
	-	Sequenced commands are either (i) stored in a command store and later polled and processed via *PollerProcessorApp*, or (ii) directly processed (see *SequencerProcessorApp*)
	-	Only one sequencer app is active at any point in time in a running Elara multi-application installation
ProcessorApp		
	-	Processes commands by routing events, and then applies those events to the application state
	-	Has read-only access to application state while processing a command
	-	Has read/write access to application state while applying a routed event to the application state
	-	App type exists in two variants: *PollerProcessorApp* and *SequencerProcessorApp*
	-	Only one processor app is active at any point in time in a running Elara multi-application installation
PollerProcessorApp		
	-	*ProcessorApp* that polls commands to process from a command store
SequencerProcessorApp		
	-	*SequencerApp* that directly invokes command processor with sequenced commands; no command store is present
SequencerPassthroughApp		
	-	*SequencerProcessorApp* that directly invokes a standard passthrough processor with every sequenced command; the passthrough processor routes a single event for every command with the same payload as the command
OutputApp		
	-	Publishes messages to downstream applications based on events
	-	Performs protocol and codec transformation if necessary
	-	Replayed events are flagged as such to prevent double-publication of events; events can be committed or marked as retry e.g. if the downstream system is currently unavailable
	-	Multiple active publisher apps can be present in a running Elara multi-application installation; each publisher app has its own last-played event marker in case of replays
TimerApp		
	-	Time as an example of a non-deterministic input source
	-	The simplest timer app version sends regular wall clock time commands into the sequencer to make time deterministic
	-	More sophisticated versions are *FeedbackApps* and hence also consume events and maintain state; with that they can selectively install timers based on events and send expiration commands when a timer expires
AllInOneApp		
	-	Combination of *SequencerApp, ProcessorApp* and *PublisherApp* n a single-threaded Application
	-	Performs all functions from input adaptation and sequencing to command processing and output publishing and hence can act as a fully featured application

	-	Since output publishing is processed in the same thread as sequencing, this application type supports a direct command loopback feature which allows insertion of commands directly into the front of the command queue when processing output events. This can be seen as a special case of an embedded *FeedbackApp* where commands are sent back to the sequencer (but not added to the front of the queue in this case)
FeedbackApp		
	-	A fully-deterministic application that consumes events, updates the application state and outputs commands
	-	This type of application allows parallelisation of functionality and remove computing load from the ProcessorApp
	-	Multiple active feedback apps can be present in a running Elara multi-application installation
PlaybackApp		
	-	A special application for publishing events on request by subscribing apps
	-	An example client is the *ReplicationApp* but playback app can also be used to bring restarted nodes back in sync
	-	Multiple active playback apps can be present in a running Elara multi-application installation
ReplicationApp		
	-	Subscribes for event playback using the *PlaybackApp* and reproduces an exact copy of the event store on another host
	-	Event store copies can be created for cold or warm standby applications, or purely for backup purposes for test regression replays
	-	Multiple active replication apps can be present in a running Elara multi-application installation
FollowerApp		
	-	Inactive *ProcessorApp* that consumes and applies events but does not process commands
	-	While command processing (and event routing) is disabled, commands are still polled from the queue and skipped when corresponding events are applied so that a failover can be performed with lowest possible delay if needed in case of main node failure
	-	Note that an inactive *FeedbackApp* can operate in normal mode if connected to a sequencer with inactive *ProcessorApp.* The *FeedbackApp* must produce exactly identical commands to this from the active sibling connected to the active *ProcessorApp.* If configured correctly with the same source IDs as the active node, the commands will be skipped when the corresponding events are applied.
	-	A *FeedbackApp* connected to a sequencer with active *ProcessorApp* is always active but can still serve as a fallback node for another node. It then shares the source ID with at least one other *FeedbackApp* and they will act as a hot/hot redundant nodes where the first one issuing a command will most likely win (the one whose command is first sequenced into the command queue will win). The *SequencerApp* will recognise duplicate sequence numbers from the same source ID issued by the other nodes and will simply drop them as duplicates.
	-	Multiple inactive follower apps can be present in a running Elara multi-application installation, but only one should ever be activated at any point in time.