Architecture

The architecture of an xApp consists of the code implementing the xApp's logic and the RIC libraries that allow the xApp to

1. Send and receive messages.
2. Read from, write to, and get notifications from the SDL layer.
3. Write log messages.

Additional libraries will be available in future versions including libraries for setting and resetting alarms and sending statistics.

Furthermore, xApps can use access libraries to access specific name-spaces in the SDL layer. For example, the R-NIB that provides information about which E2 nodes (e.g., CU/DU) the RIC is connected to and which SMs are supported by each E2 node, can be read by using the R-NIB access library.

The O-RAN standard interfaces (O1, A1, and E2) are exposed to the xApps as follows:

- xApp will receive its configuration via K8s ConfigMap - the configuration can be updated while the xApp is running and the xApp can be notified of this modification by using `inotify()`.
- xApp can send statistics (PM) either by (a) sending it directly to VES collector in VES format, (b) by exposing statistics via a REST interface for Prometheus to collect
- xApp will receive A1 policy guidance via an RMR message of a specific kind (policy instance creation and deletion operations)
- xApp can subscribe to E2 events by constructing the E2 subscription ASN.1 payload and sending it as a message (RMR). xApp will receive E2 messages (e.g., E2 INDICATION) as RMR messages with the ASN.1 payload. Similarly xApp can issue E2 control messages.

In addition to A1 and E2 related messages, xApps can send messages that are processes by other xApps and can receive messages produced by other xApps. Communication inside the RIC is policy driven, that is, an xApp cannot specify the target of a message. It simply sends a message of a specific type and the routing policies specified for the RIC instance will determine to which destinations this message will be delivered (logical pub/sub).