Samsung Research

O-RAN xApp Developer Session

Joint Session by

SAMSUNG

NOKIA

03-July-2021

Agenda

Developer Session Contents

- Introduction of O-RAN Alliance and OSC
- Introduction to xApp
- xApp development framework overview
- Demo
- Q/A

Our Speakers Today

Samsung Research

Thoralf Czichy (Nokia)

Thoralf is an expert in telco platforms, with over 20 years of experience in building complex real-time systems. Recently, he has been the key contributor in RIC architecture work and he is PTL of the near-realtime RIC open source project in O-RAN SC.

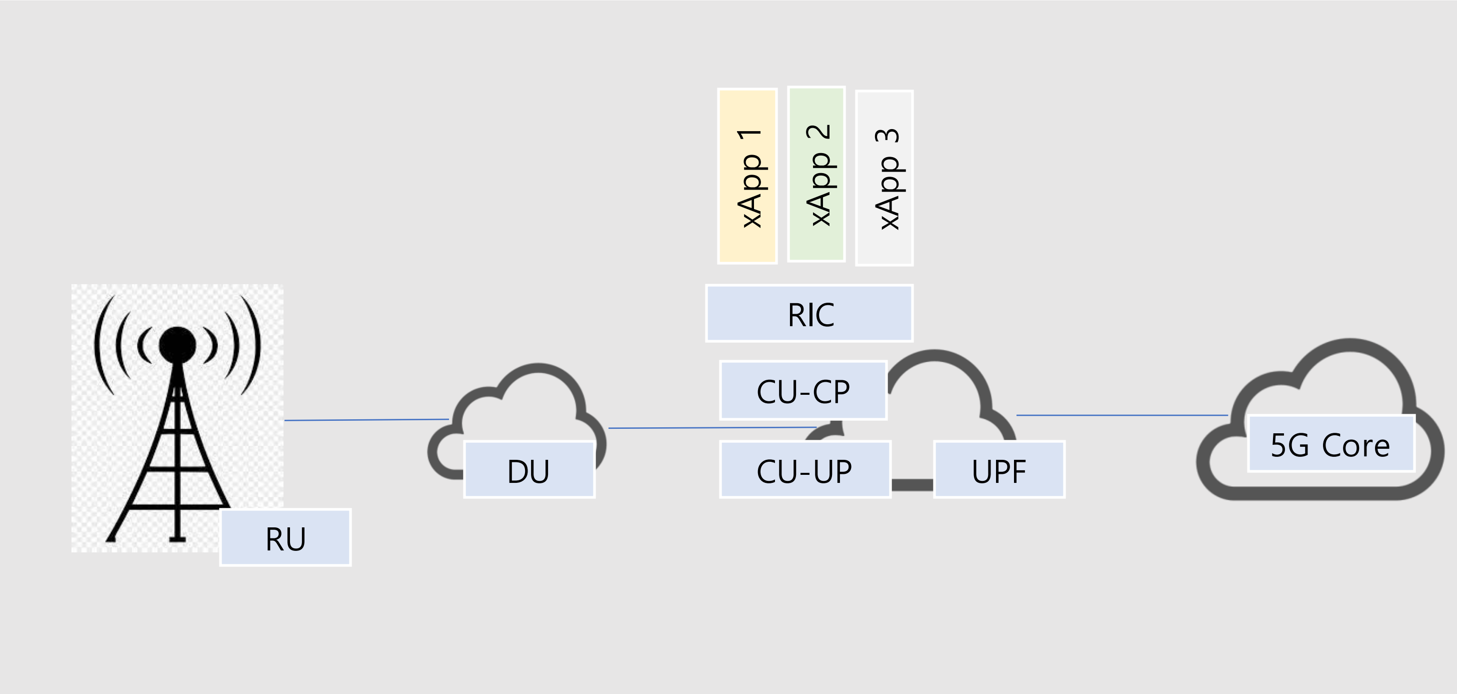
Subhash Kumar Singh (Samsung)

Subhash Kumar Singh has contributed to many OpenSource projects of LFN (Linux Foundation Networking). He holds committer role in RIC-PLT and RIC-APP project of O-RAN SC. Apart from O-RAN SC he has played significant roles in ONAP, ODL, Open-O and ONF projects.

Naman Gupta (Samsung)

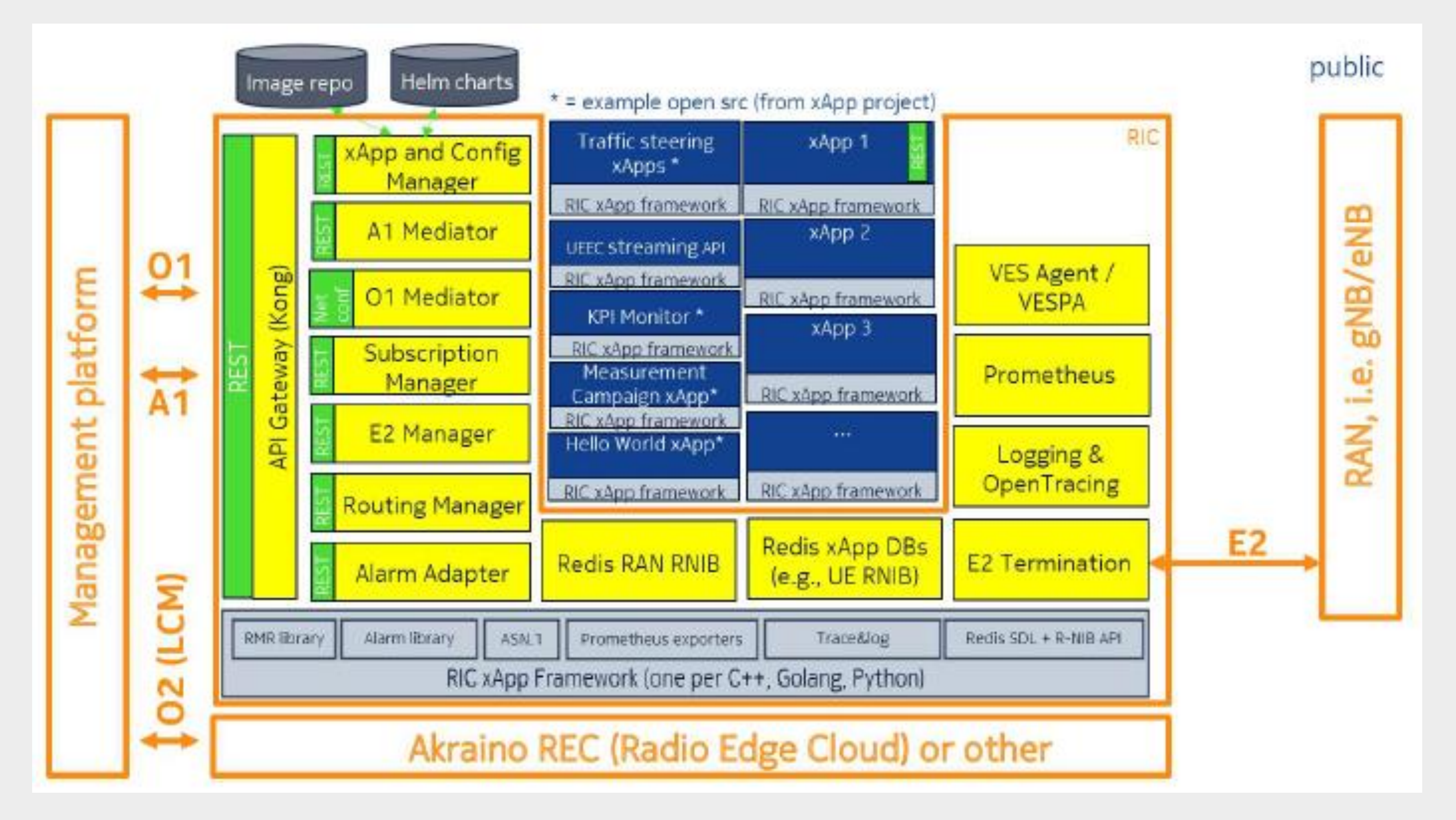
Naman Gupta has been contributing to ric-xapp framework and HW xapp of O-RAN SC. He holds committer role in RIC-PLT and RIC-APP project. He has also worked on ONAP and non-RT RIC projects.

What is xApp ??



"xApp is an entity that implements a well defined function."

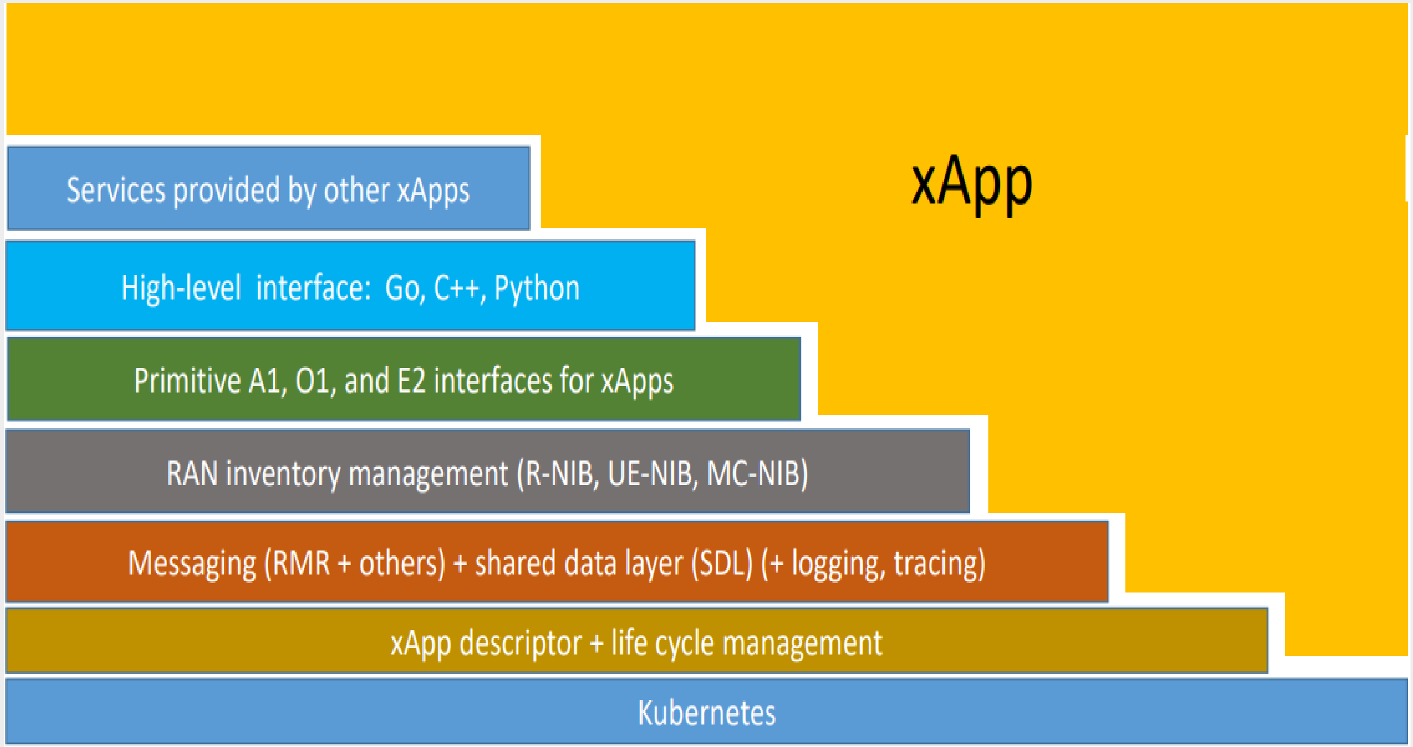
O-RAN nearRT RIC Platform

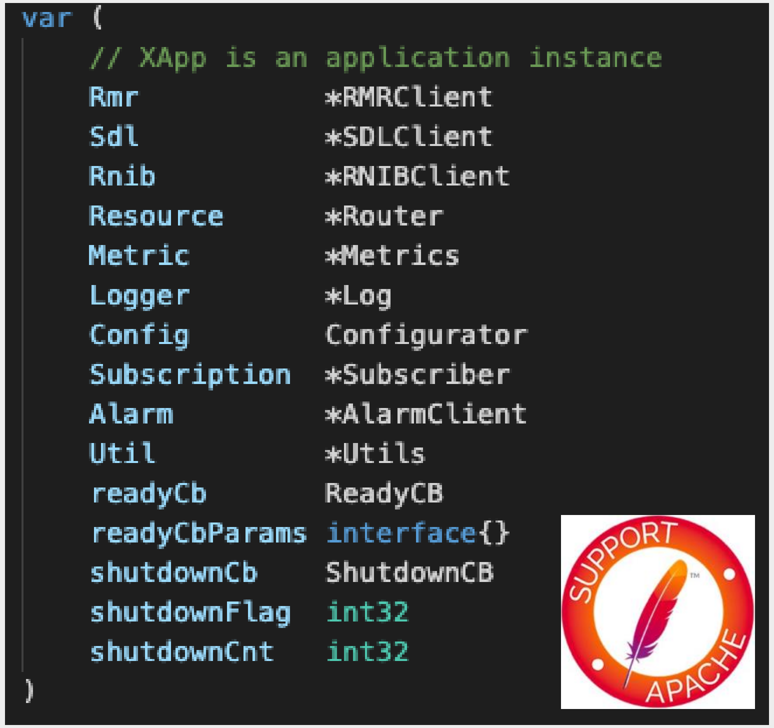


Reference: https://ones2020.sched.com/event/bWPL/the-o-ran-sc-ric-ran-intelligent-controller-xapp-api-and-the-northbound-a1-o1-and-southbound-protocols-towards-enbgnb-e2ap-e2sm-thoralf-czichy-nokia-matti-hiltunen-att

xApp Development Framework Overview

xApp Framework Overview

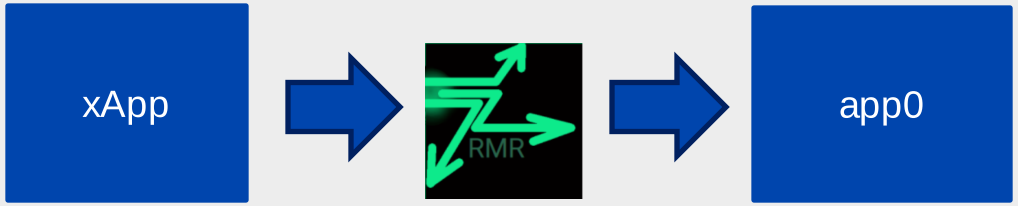

xApp
Services provided by other xApps
High-level interface: Go, C++, Python
Primitive A1, O1, and E2 interfaces for xApps
RAN inventory management (R-NIB, UE-NIB, MC-NIB)
Messaging (RMR + others) + shared data layer (SDL) (+ logging, tracing)
xApp descriptor + life cycle management
Kubernetes



```
var (
    // XApp is an application instance
    Rmr             *RMRClient
    Sdl             *SDLClient
    Rnib            *RNIBClient
    Resource        *Router
    Metric          *Metrics
    Logger          *Log
    Config          Configurator
    Subscription    *Subscriber
    Alarm           *AlarmClient
    Util            *Utils
    readyCb         ReadyCB
    readyCbParams   interface{}
    shutdownCb      ShutdownCB
    shutdownFlag    int32
    shutdownCnt     int32
)
```

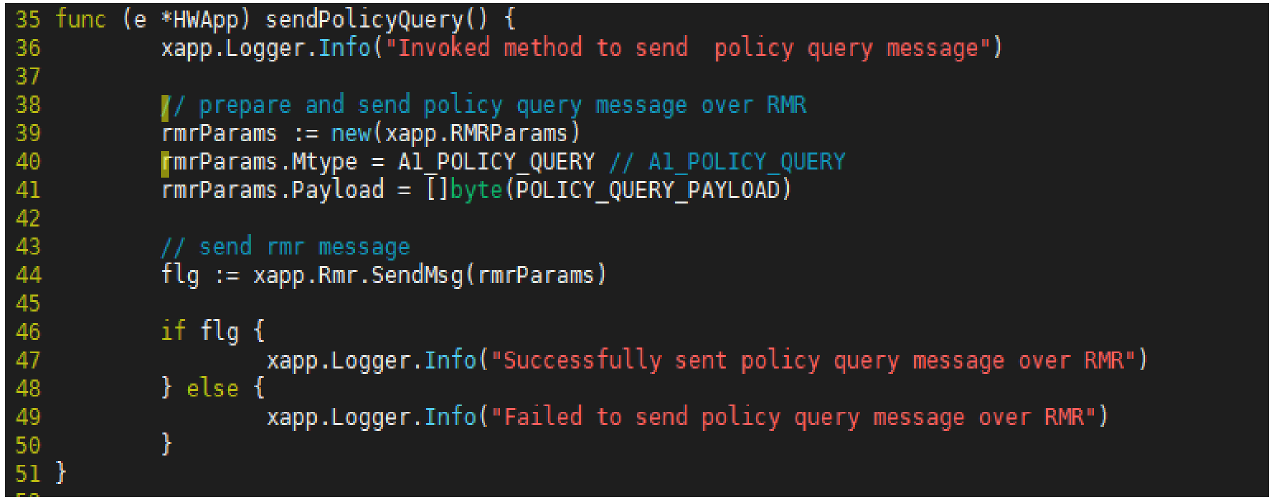
Reference: https://ones2020.sched.com/event/bWPL/the-o-ran-sc-ric-ran-intelligent-controller-xapp-api-and-the-northbound-a1-o1-and-southbound-protocols-towards-enbgnb-e2ap-e2sm-thoralf-czichy-nokia-matti-hiltunen-att

RMR: xApp Communication



```
newrt | start | rt-0928
rte   | 2000  | logger:30311
mse   | 1000  | 10 | forwarder:43086
mse   | 1000  | 21 | app0:43086,app1:43086
newrt | end   | 3
```

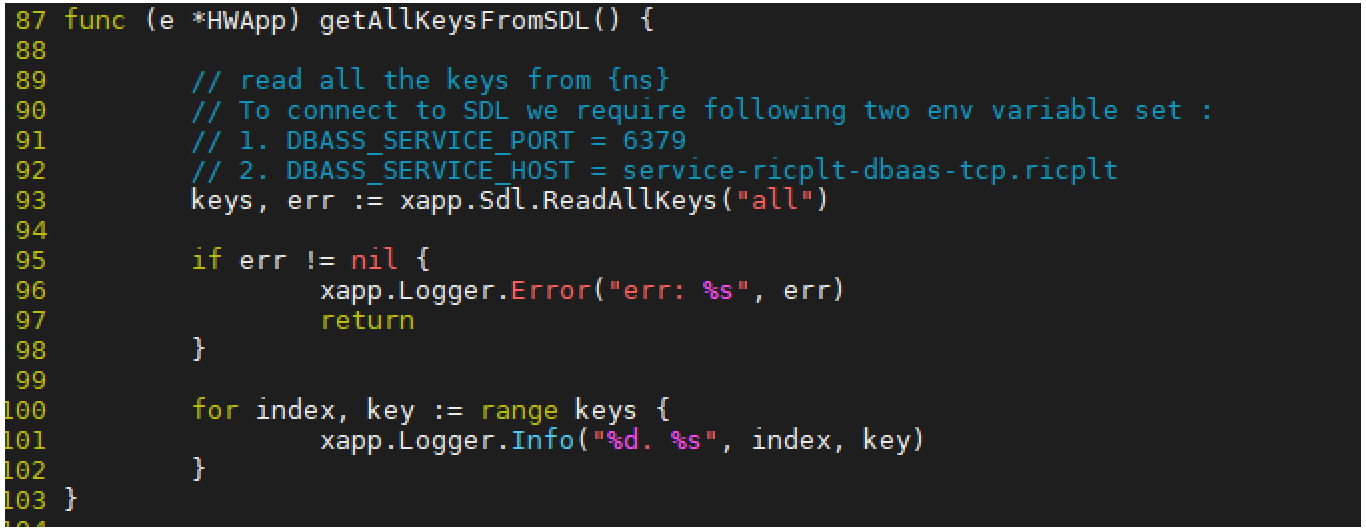
uta_rtg.rt



```
func (e *HWApp) sendPolicyQuery() {
	xapp.Logger.Info("Invoked method to send  policy query message")

	// prepare and send policy query message over RMR
	rmrParams := new(xapp.RMRParams)
	rmrParams.Mtype = A1_POLICY_QUERY // A1_POLICY_QUERY
	rmrParams.Payload = []byte(POLICY_QUERY_PAYLOAD)

	// send rmr message
	flg := xapp.Rmr.SendMsg(rmrParams)

	if flg {
		xapp.Logger.Info("Successfully sent policy query message over RMR")
	} else {
		xapp.Logger.Info("Failed to send policy query message over RMR")
	}
}
```

SDL: Shared Data Layer

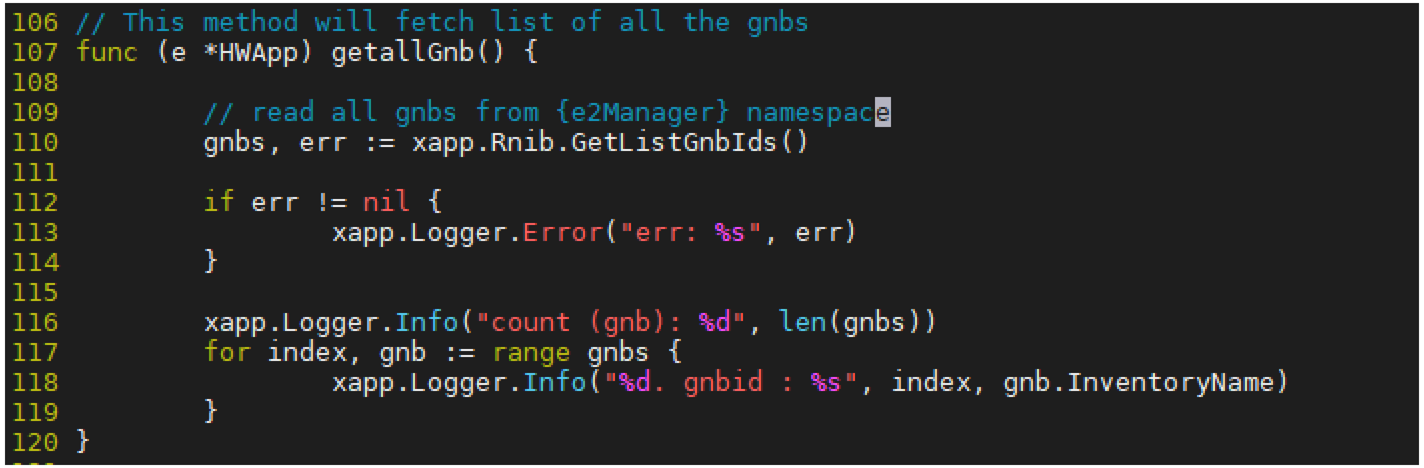
Shared Data Layer provides a lightweight, highspeed interface for accessing shared data storage. The purpose is to enable utilizing clients to become stateless, conforming with, e.g., the requirements of the fifth generation mobile networks.



```go
func (e *HWApp) getAllKeysFromSDL() {

        // read all the keys from {ns}
        // To connect to SDL we require following two env variable set :
        // 1. DBASS_SERVICE_PORT = 6379
        // 2. DBASS_SERVICE_HOST = service-ricplt-dbaas-tcp.ricplt
        keys, err := xapp.Sdl.ReadAllKeys("all")

        if err != nil {
                xapp.Logger.Error("err: %s", err)
                return
        }

        for index, key := range keys {
                xapp.Logger.Info("%d. %s", index, key)
        }
}
```

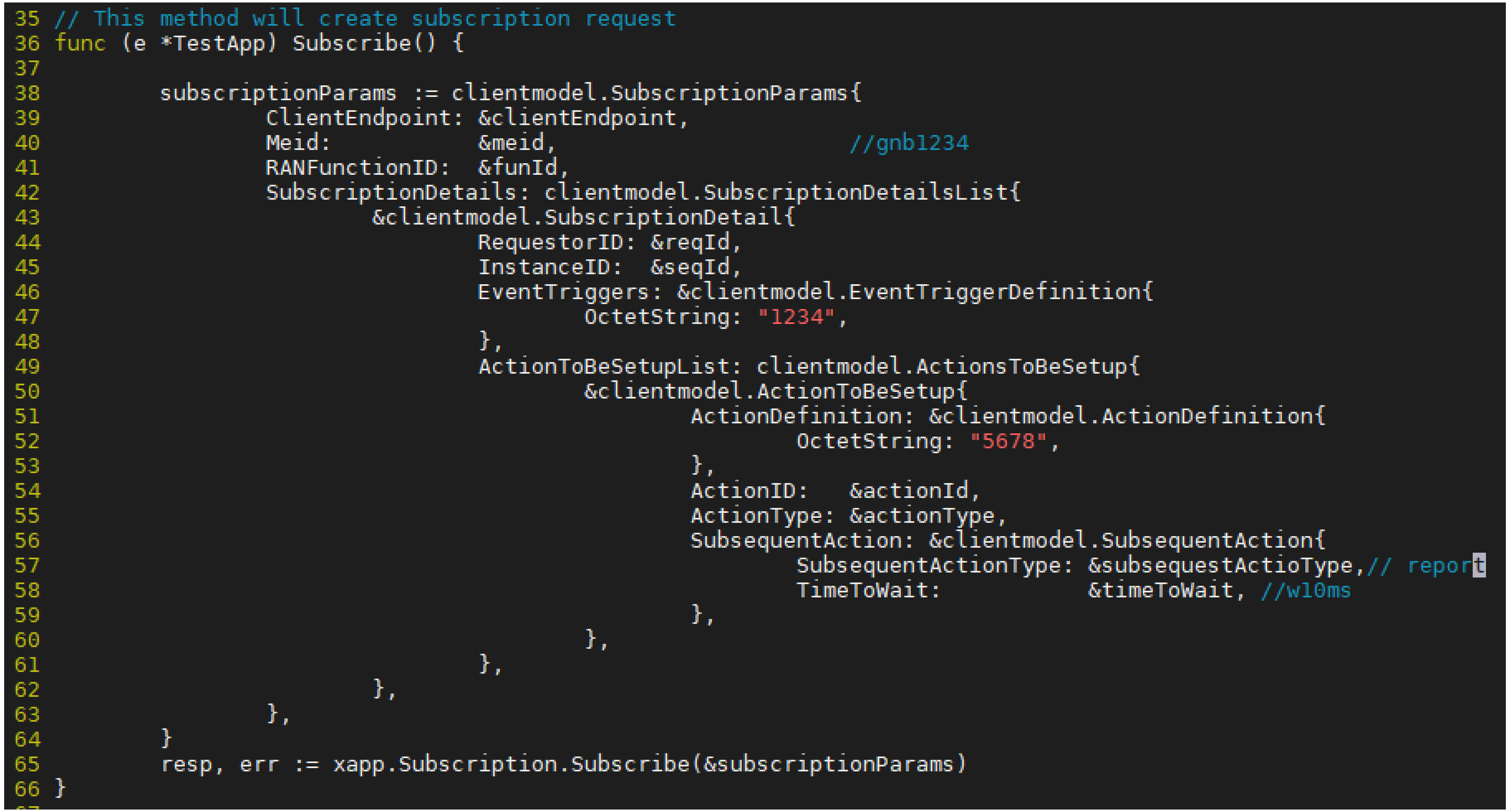
R-NIB

RIC applications can obtain the list of GNBs connected/discovered, and as well as their connection status stored in RNIB.



```go
// This method will fetch list of all the gnbs
func (e *HWApp) getallGnb() {

        // read all gnbs from {e2Manager} namespace
        gnbs, err := xapp.Rnib.GetListGnbIds()

        if err != nil {
                xapp.Logger.Error("err: %s", err)
        }

        xapp.Logger.Info("count (gnb): %d", len(gnbs))
        for index, gnb := range gnbs {
                xapp.Logger.Info("%d. gnbid : %s", index, gnb.InventoryName)
        }
}
```

Subscription



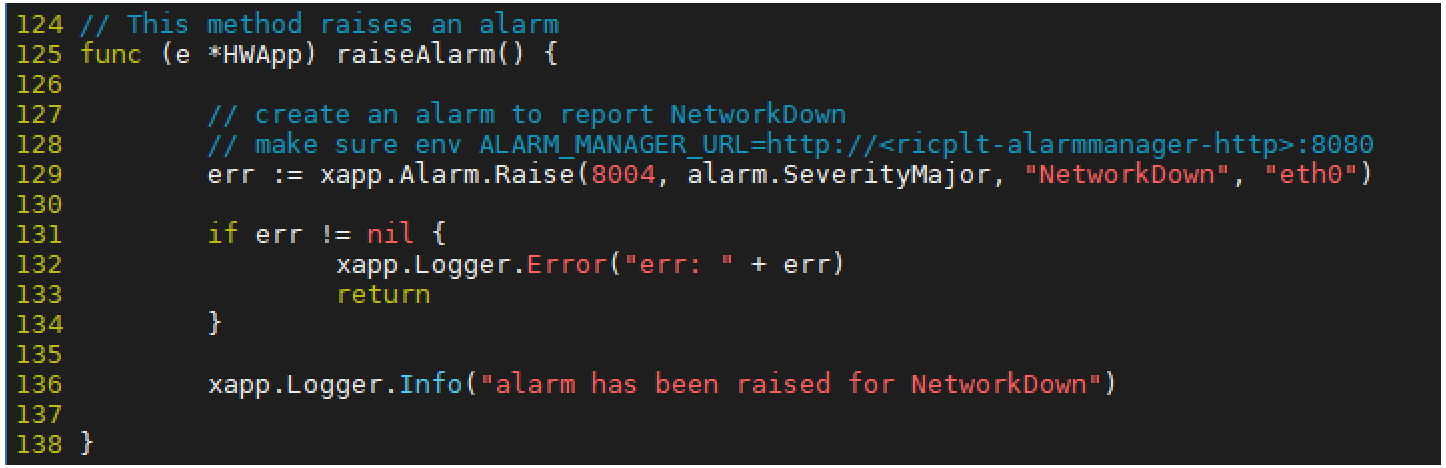
```go
// This method will create subscription request
func (e *TestApp) Subscribe() {

        subscriptionParams := clientmodel.SubscriptionParams{
                ClientEndpoint: &clientEndpoint,
                Meid:           &meid,                      //gnb1234
                RANFunctionID:  &funId,
                SubscriptionDetails: clientmodel.SubscriptionDetailsList{
                        &clientmodel.SubscriptionDetail{
                                RequestorID: &reqId,
                                InstanceID:  &seqId,
                                EventTriggers: &clientmodel.EventTriggerDefinition{
                                        OctetString: "1234",
                                },
                                ActionToBeSetupList: clientmodel.ActionsToBeSetup{
                                        &clientmodel.ActionToBeSetup{
                                                ActionDefinition: &clientmodel.ActionDefinition{
                                                        OctetString: "5678",
                                                },
                                                ActionID:   &actionId,
                                                ActionType: &actionType,
                                                SubsequentAction: &clientmodel.SubsequentAction{
                                                        SubsequentActionType: &subsequestActioType,// report
                                                        TimeToWait:           &timeToWait, //w10ms
                                                },
                                        },
                                },
                        },
                },
        }
        resp, err := xapp.Subscription.Subscribe(&subscriptionParams)
}
```

REST interface for xApps towards E2 subscription manager. No need to encode E2AP subscription messages in the xApps anymore.

Alert

The Alarm Manager is responsible for managing alarm situations in RIC cluster and interfacing with Northboubd applications such as Prometheus AlertManager to post the alarms as alerts.

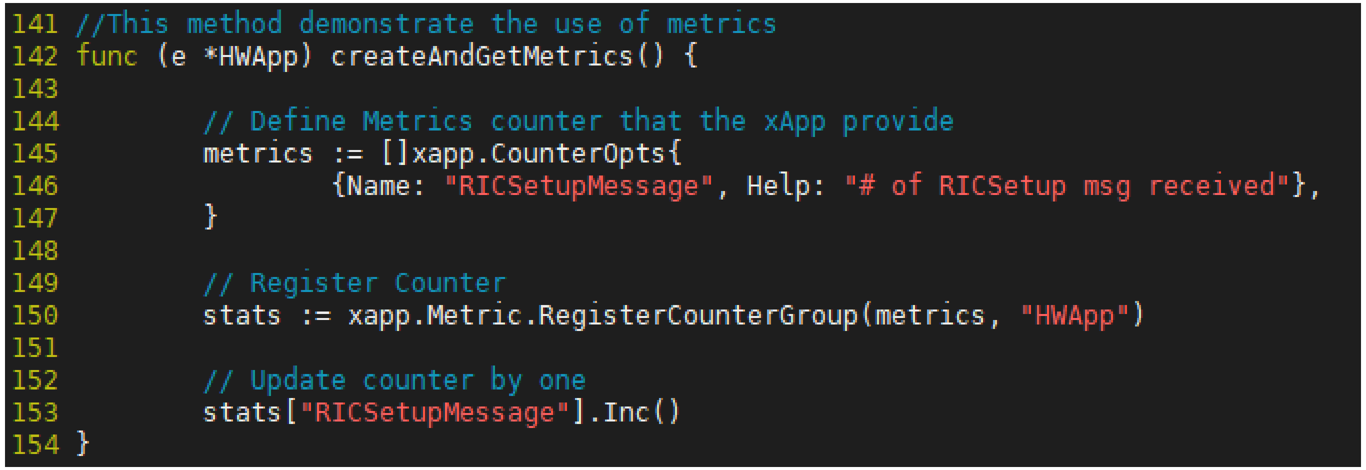
Raise: Raises the alarm instance given as a parameter
Clear: Clears the alarm instance given as a parameter, if it the alarm active
Reraise: Attempts to re-raise the alarm instance given as a parameter
ClearAll: Clears all alarms matching moId and appId given as parameters



```
// This method raises an alarm
func (e *HWApp) raiseAlarm() {

        // create an alarm to report NetworkDown
        // make sure env ALARM_MANAGER_URL=http://<ricplt-alarmmanager-http>:8080
        err := xapp.Alarm.Raise(8004, alarm.SeverityMajor, "NetworkDown", "eth0")

        if err != nil {
                xapp.Logger.Error("err: " + err)
                return
        }

        xapp.Logger.Info("alarm has been raised for NetworkDown")

}
```

Metrics

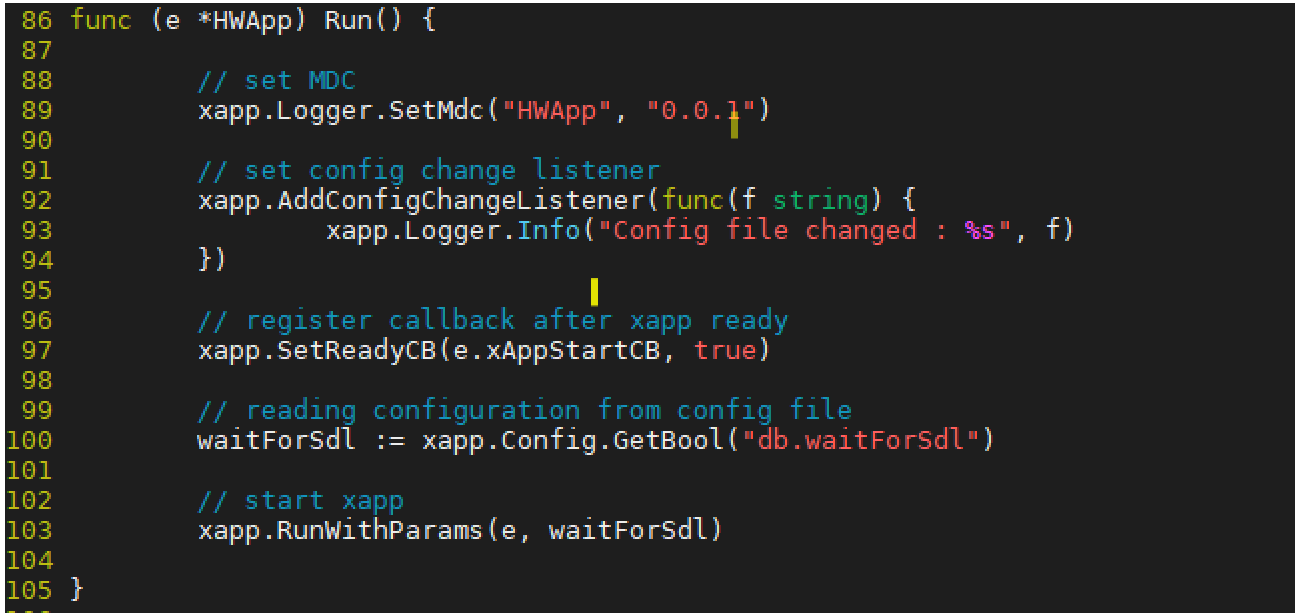
RIC applications can act as metrics providers and expose the metrics data to external centralized time-series DB servers. Prometheus interface is used to periodically collect metrics data and forward to ONAP via VES agent.



```go
//This method demonstrate the use of metrics
func (e *HWApp) createAndGetMetrics() {

        // Define Metrics counter that the xApp provide
        metrics := []xapp.CounterOpts{
                {Name: "RICSetupMessage", Help: "# of RICSetup msg received"},
        }

        // Register Counter
        stats := xapp.Metric.RegisterCounterGroup(metrics, "HWApp")

        // Update counter by one
        stats["RICSetupMessage"].Inc()
}
```

Reading and watching config changes

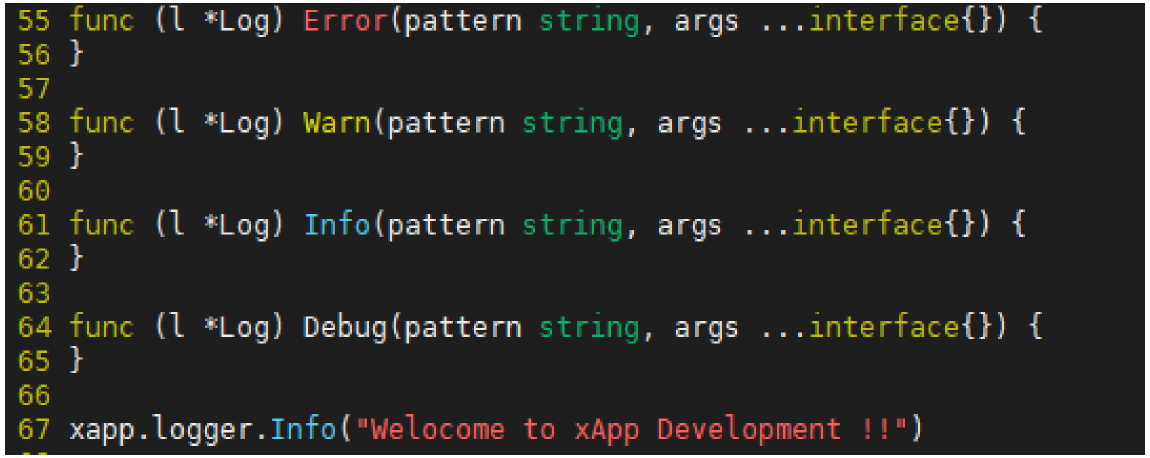
xApps can watch and read live config file while running. xApps don't need to be restarted to have their config file changes to take effect. To watch for config file changes, xApps provide a callback function for xApp-framework to run whenever a file change occurs.



```
func (e *HWApp) Run() {

        // set MDC
        xapp.Logger.SetMdc("HWApp", "0.0.1")

        // set config change listener
        xapp.AddConfigChangeListener(func(f string) {
                xapp.Logger.Info("Config file changed : %s", f)
        })

        // register callback after xapp ready
        xapp.SetReadyCB(e.xAppStartCB, true)

        // reading configuration from config file
        waitForSdl := xapp.Config.GetBool("db.waitForSdl")

        // start xapp
        xapp.RunWithParams(e, waitForSdl)

}
```

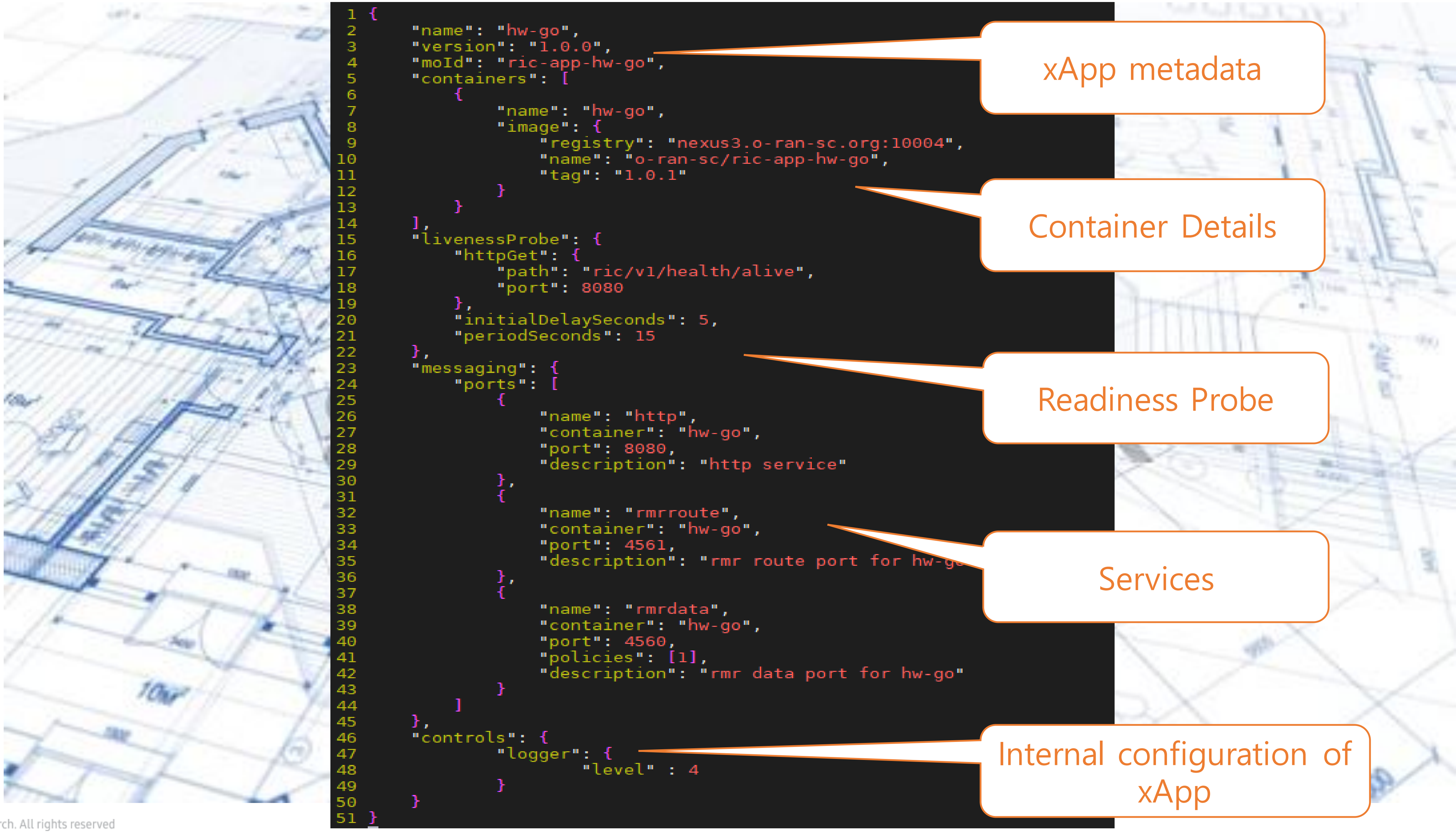
Logging

The RIC platform provides a logging library that ensures that the log entries generated by xApps will have a standard format and will be handled uniformly.



```
func (l *Log) Error(pattern string, args ...interface{}) {
}

func (l *Log) Warn(pattern string, args ...interface{}) {
}

func (l *Log) Info(pattern string, args ...interface{}) {
}

func (l *Log) Debug(pattern string, args ...interface{}) {
}

xapp.logger.Info("Welocome to xApp Development !!")
```

xApp Descriptor



```
{
    "name": "hw-go",
    "version": "1.0.0",
    "moId": "ric-app-hw-go",
    "containers": [
        {
            "name": "hw-go",
            "image": {
                "registry": "nexus3.o-ran-sc.org:10004",
                "name": "o-ran-sc/ric-app-hw-go",
                "tag": "1.0.1"
            }
        }
    ],
    "livenessProbe": {
        "httpGet": {
            "path": "ric/v1/health/alive",
            "port": 8080
        },
        "initialDelaySeconds": 5,
        "periodSeconds": 15
    },
    "messaging": {
        "ports": [
            {
                "name": "http",
                "container": "hw-go",
                "port": 8080,
                "description": "http service"
            },
            {
                "name": "rmrroute",
                "container": "hw-go",
                "port": 4561,
                "description": "rmr route port for hw-go
            },
            {
                "name": "rmrdata",
                "container": "hw-go",
                "port": 4560,
                "policies": [1],
                "description": "rmr data port for hw-go"
            }
        ]
    },
    "controls": {
            "logger": {
                    "level" : 4
            }
    }
}
```

xApp metadata

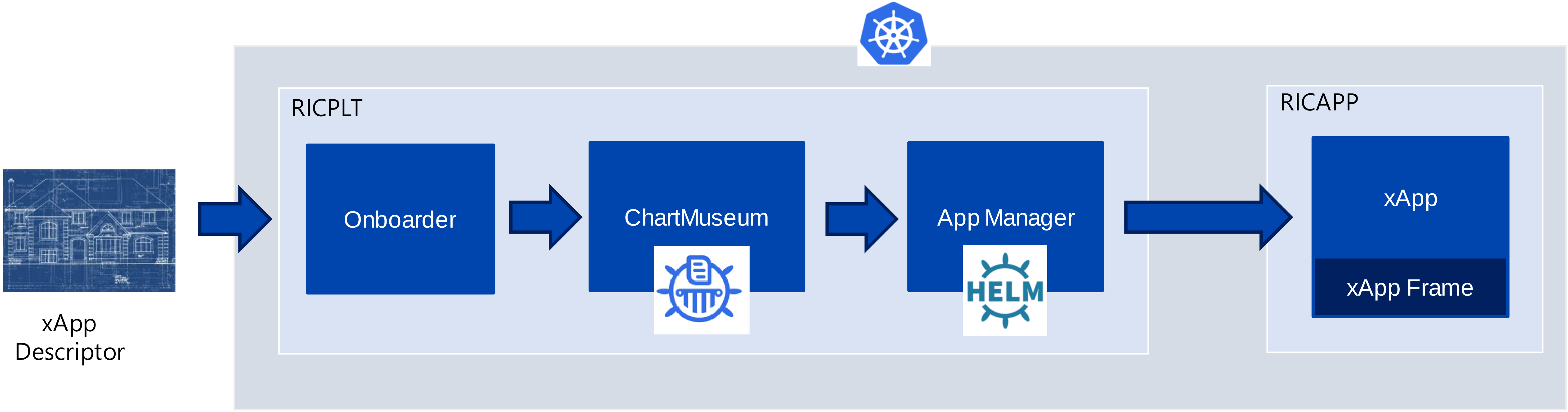
Container Details

Readiness Probe

Services

Internal configuration of xApp

xApp Onboarding and Deployment



Demo

Wiki : https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/hw-go.git;a=blob;f=README.md;
Code : https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/hw-go.git
https://gerrit.o-ran-sc.org/r/gitweb?p=ric-app/hw-python.git

O-RAN SC – Samsung Contribution

- A1-EI interface support at a1-mediator
- Enhancement in xApp-framework.
- Enhancement in nearRT RIC platform.
- **hw-go** and **hw-python** as reference xApp.
- Design to support E2AP v1.1 in nearRT RIC.

Q & A

discuss@lists.o-ran-sc.org

Samsung Research

Thank You