RDK-B MoCA component architecture

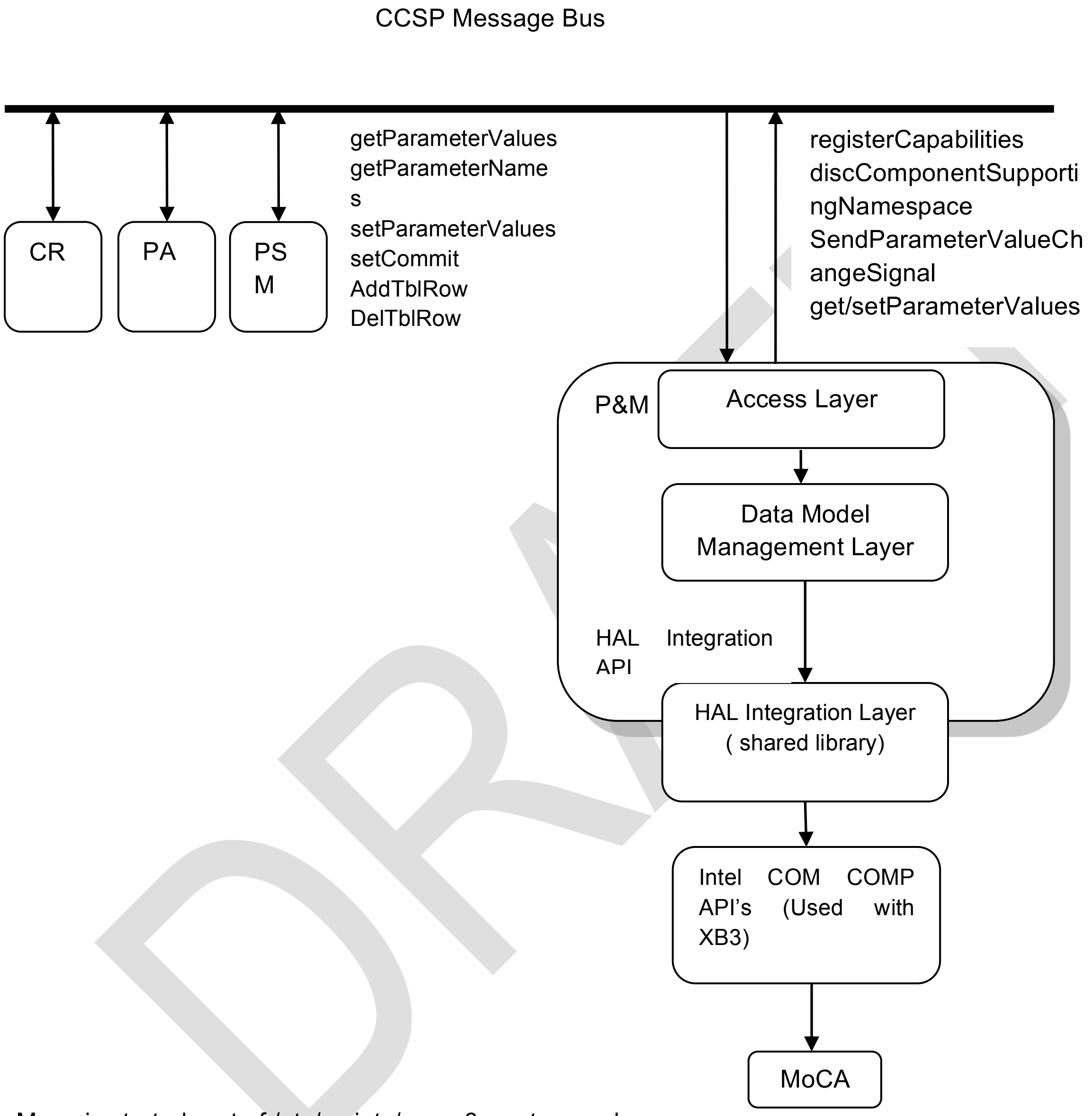
☒ **Draft**
☐ Baselined for Estimates

Author	
Authorized by	
Version	0.3
Date	05-09-2014

Table of Contents

1. MoCA Component Architecture

RDK-B usages SoC level MoCA abstraction layer to integrate MoCA with RDK-B stack.



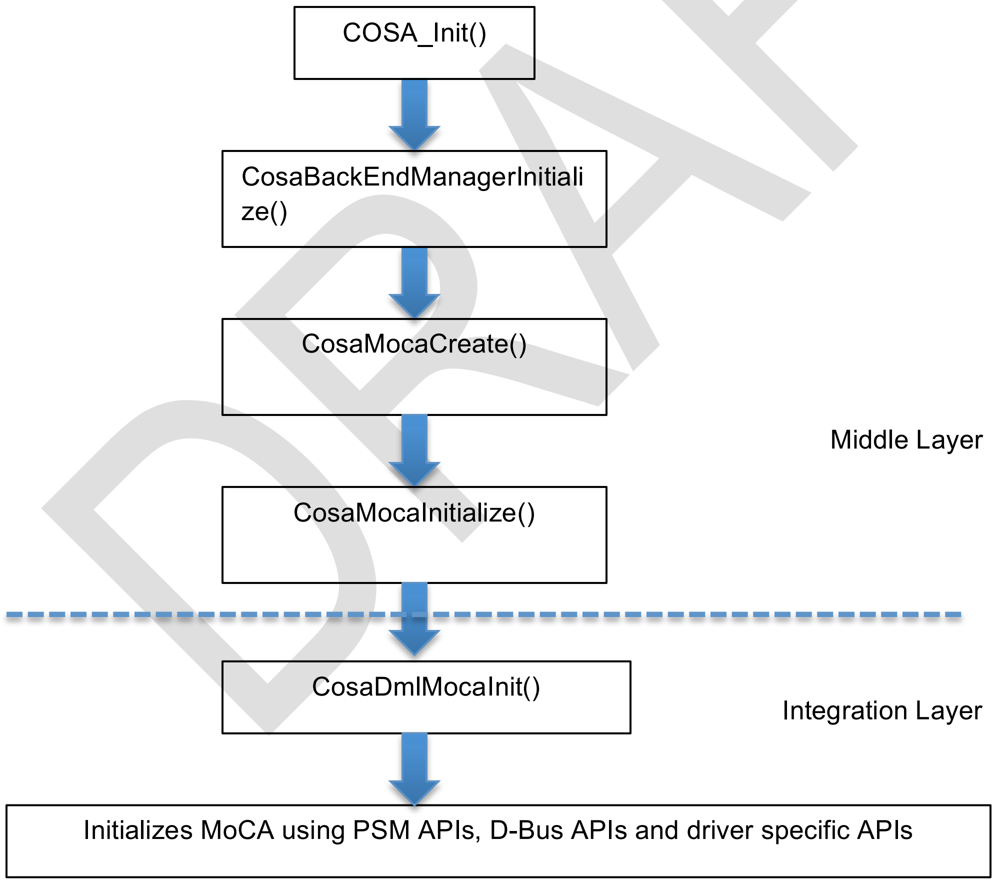
Moca is started part of /etc/scripts/puma6_system.pcd
This will start the script /etc/scripts/moca_enable.sh. This script will start /usr/sbin/mocactld process

2. MOCA Abstraction layer (CISCO needs to confirm this.)

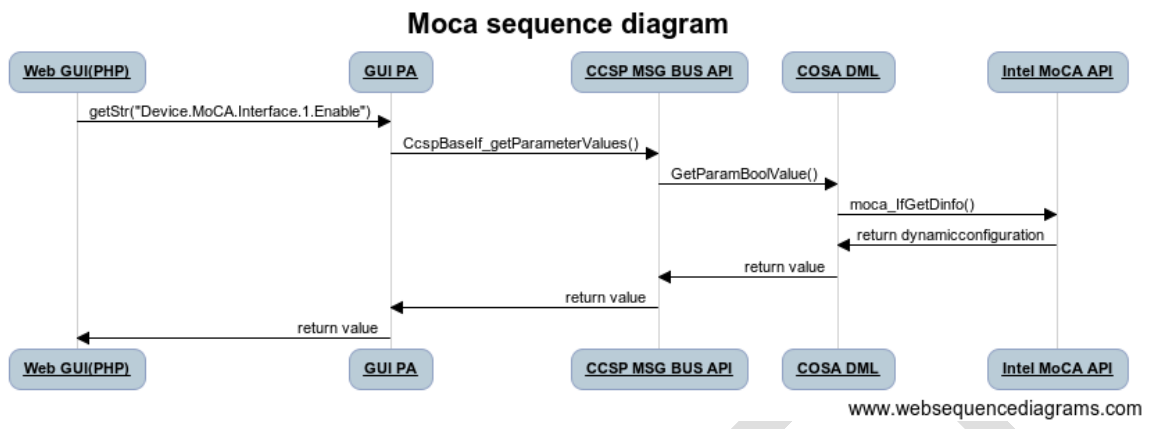
The file is from intel_usg/arm/ti/common_components/src/moca/src/moca_api_lib/include/moca_api.h

```
**** Functions prototypes ****/
eMoca_result_t moca_IsAttached(bool *status);
eMoca_result_t moca_GetAdapterInfo (unsigned char *name, char *addr);

/*  Get MoCA Attributes */

eMoca_result_t  moca_GetVersionInfo (moca_version_info_t * info);
eMoca_result_t  moca_GetMacCtrl(get_mac_control_response_t *mac_info);
eMoca_result_t  moca_GetMyNode (uint8_t *node_id, ETHER_ADDR* mac_addr);
eMoca_result_t  moca_GetNcNode (uint8_t *node_id, ETHER_ADDR* mac_addr);
eMoca_result_t  moca_GetBackUpNcNode (uint8_t *node_id, ETHER_ADDR* mac_addr);
eMoca_result_t  moca_GetLMONode (uint8_t *node_id, ETHER_ADDR *mac_addr);
eMoca_result_t  moca_GetLinkStatus (uint32_t  *link_status);
eMoca_result_t  moca_GetPrivacy (bool *enable);
eMoca_result_t  moca_GetMyNodeInfo(moca_node_private_info_t *node_info);

eMoca_result_t  moca_GetPhyData(moca_phy_data_t *phy_data);
eMoca_result_t  moca_GetTabooData(uint32_t *taboo_start_freq,  uint32_t* taboo_channel_mask);
eMoca_result_t  moca_GetLof (uint32_t *last_operational_freq);
eMoca_result_t  moca_GetNetFreq (uint32_t *rf_channel);
eMoca_result_t  moca_GetLinkProfiles(moca_LinkProfile_t pLinkProfiles[4]);
eMoca_result_t  moca_GetAllStats(moca_all_stats_t *stats, uint16_t* size);
eMoca_result_t  moca_GetMocaStats(moca_get_moca_stats_t *stats, uint16_t* size);
eMoca_result_t  moca_GetVendorId(get_vendor_id *vendor_id);
eMoca_result_t  moca_GetFullMeshRate(get_fmr_t *fmr, uint32_t *size);
eMoca_result_t  moca_GetGmacRxStats(moca_gmac_rx_stats_t *gmac_rx, uint16_t* len);
eMoca_result_t  moca_GetGmacTxStats(moca_gmac_tx_stats_t *gmac_tx, uint16_t* len);
eMoca_result_t  moca_GetNodeStats(Moca_AllNodeStats  *node_stat, uint16_t* size);
eMoca_result_t  moca_GetMacErrorStats(Moca_Mmac_Errors* err_stats, uint16_t* len);
eMoca_result_t  moca_GetPhyErrorStats(Moca_Mphy_Errors* err_stats, uint16_t* len);

eMoca_result_t  moca_GetEnabledData(enable_operation_request_API* enabled_data);

/*  Set MoCA Attributes */
eMoca_result_t moca_SetTargetPhyRate(uint32_t target, uint32_t alarm);
eMoca_result_t moca_SetClearStats();
eMoca_result_t moca_ReadPhyReg(uint32_t addr, uint32_t* value);
eMoca_result_t moca_WritePhyReg(uint32_t addr, uint32_t value);
eMoca_result_t moca_MocaDiagnose(uint32_t,uint32_t);
eMoca_result_t moca_delorean_temperature(uint32_t* T_out);
eMoca_result_t moca_ducati_temperature(uint32_t* T_out);
eMoca_result_t was_device_enabled(uint32_t *mocaIfEnable);
eMoca_result_t moca_VlanRx(uint32_t,uint32_t);
eMoca_result_t moca_VlanTx(uint32_t,uint32_t);

#ifdef MOCA_E_BAND
eMoca_result_t moca_SetSapm(bool enable, int threshold, const unsigned char margin[256]);
eMoca_result_t moca_SetRlapm(bool enable, int numEntries,
        int thresholds[kMidrf_MaxGarpIPairs], unsigned int margins[kMidrf_MaxGarpIPairs]);
eMoca_result_t moca_SetReset(uint32_t node_bitmask, uint8_t reset_timer, uint16_t seq_num);
#endif
//eMoca_result_t moca_SetFactoryCalibration(int8_t, uint8_t, uint8_t, uint8_t);

eMoca_result_t moca_Disable();
eMoca_result_t moca_SetEnable();
eMoca_result_t moca_SetEnableDefault();
eMoca_result_t moca_EnableSetPassword(const char *password);
eMoca_result_t moca_EnableSetMac(const char *mac_addr);
eMoca_result_t moca_EnableSetBand(int32_t lof, int32_t low, int32_t high, uint32_t spacing);
eMoca_result_t moca_EnableSetLof(int32_t lof);
eMoca_result_t moca_EnableSetConstantPower(bool value);
eMoca_result_t moca_EnableSetMaxPower(uint16_t value);
eMoca_result_t moca_EnableSetDigOffset(uint16_t value);
eMoca_result_t moca_EnableSetBeaconackoff(uint8_t value);
eMoca_result_t moca_EnableSetTxQam256(bool txQam256);
eMoca_result_t moca_EnableSetMode(eMoca_PreferredNC value);
eMoca_result_t moca_EnableSetScan (uint32_t mask);
eMoca_result_t moca_EnableSetPlan(uint32_t mask);
eMoca_result_t moca_start_fw(uint8_t * device_name);


eMoca_result_t moca_RegisterNotficationEvent(eMoca_EventType event_type, moca_FunPtr callback,
const void *Cookie, moca_EventHandle *handle);
eMoca_result_t moca_UnRegisterNotficationEvent(moca_EventHandle handle);
/* PQOS requests */

eMoca_result_t moca_QosCreateMcastFlow(moca_pqos_flow_create_update_t *pqos_flow ,
                        moca_pqos_create_flow_response_t *pqos_result);
eMoca_result_t moca_QosUpdateFlow(moca_pqos_flow_create_update_t  *pqos_flow,
                        moca_pqos_create_flow_response_t *pqos_result);
eMoca_result_t moca_QosCreateUcastFlow(moca_pqos_flow_create_update_t *flow,
                         pqos_unicast_attrs_t *ucast_attrs,moca_pqos_create_flow_response_t *info);
eMoca_result_t moca_QosQueryFlow(ETHER_ADDR flow_id,  moca_pqos_query_result_t *info);
eMoca_result_t moca_QosDeleteFlow(ETHER_ADDR flow_id);

eMoca_result_t moca_GetMaxNodes(uint32_t *max_nodes);
eMoca_result_t moca_GetMaxIngress(uint32_t *max_ingress_bw);
eMoca_result_t moca_GetMaxEgress(uint32_t *max_egress_bw);
eMoca_result_t moca_GetFreqInfo(moca_freq_info_t *moca_freq_info);
eMoca_result_t moca_GetLocalNodeStats(moca_node_stats_t *local_node_stats);
eMoca_result_t                                moca_GetFlowStats(moca_egress_flow_info_t
moca_egress_flow[MAX_INGRESS_FLOWS], uint32_t *num_flows);
eMoca_result_t                              moca_GetAssocDevices(moca_assoc_dev_info_t
moca_dev_info[MAX_ASSOC_DEVICES], uint32_t *num_devices);
eMoca_result_t moca_GetExtLinkStatus (uint32_t  *ext_link_status);
eMoca_result_t moca_GetBestNcNode (uint8_t *node_id, ETHER_ADDR *mac_addr);
eMoca_result_t moca_GetNetworkVersion(uint32_t *version);
eMoca_result_t moca_SetDiagnosticMode(uint32_t operation);
eMoca_result_t moca_SetDiagnosticPwrCtrl (uint32_t tpc_backoff);
eMoca_result_t moca_getPacketCount(uint32_t *rx_packet_array, uint32_t *tx_packet_array );
eMoca_result_t moca_getAggregatePacketInfo(uint32_t *rx_array, uint32_t *tx_array, uint32_t *mocaIfPacketsAggrCapability );
eMoca_result_t moca_setPowerSaveMode(eMoca_PwrSaveMode  state);
eMoca_result_t moca_GetPowerSaveMode (uint32_t *power_save_mode);
eMoca_result_t moca_SetMocaVerion(uint32_t ver);
eMoca_result_t moca_SetPrivacy(bool enable);
eMoca_result_t moca_DisableApc(void);
eMoca_result_t moca_GetRegPeek(uint32_t address,uint32_t *buffer);
eMoca_result_t moca_GetMemPeek(uint32_t address, uint32_t size, uint32_t *buffer);
int save_current_lof();
```

This is the init sequence for the MoCA Component specific abstraction layer (COSA).


COSA_Init()
CosaBackEndManagerInitialize()
CosaMocaCreate()
Middle Layer
CosaMocaInitialize()
CosaDmlMocaInit()
Integration Layer
Initializes MoCA using PSM APIs, D-Bus APIs and driver specific APIs

3. MoCA Sequence Flow


Moca sequence diagram
Web GUI(PHP)
GUI PA
CCSP MSG BUS API
COSA DML
Intel MoCA API
getStr("Device.MoCA.Interface.1.Enable")
CcspBaseIf_getParameterValues()
GetParamBoolValue()
moca_IfGetDinfo()
return dynamicconfiguration
return value
return value
return value
Web GUI(PHP)
GUI PA
CCSP MSG BUS API
COSA DML
Intel MoCA API
www.websequencediagrams.com