BitCloud SDK 3.3.0 Migration Guide

1 Introduction

BitCloud SDK v3.3.0 combines Atmel ZigBee PRO (rev20) based applications for ZigBee LightLink, ZigBee Home Automation and vendor-specific OEM profiles. It introduces updated applications and improved core stack compared to previous public release BitCloud v3.2.0.

This document describes key changes in the BitCloud stack and applications code that might require porting efforts for custom applications developed on previous version of BitCloud SDK.

All the other major features/enhancements/bug fixes are captured as part of the Release_Notes.txt document available in the SDK.

2 Relevant to all reference applications

This Section describes new items in BC3.3 that are relevant for all BitCloud reference applications, although some of them might apply only to particular HW platforms.

2.1 Items that require mandatory changes when porting any application from BC3.2

This subsection describes changes that are mandatory to do for any BitCloud v3.2-based application in order to successfully recompile and run it on top of BitCloud v3.3 SDK.

Note that there're also other changes that are mandatory to do but are different for each reference application. They are covered in Section 3.1 for ZLLDemo and Section 4.1 for HADevice applications respectively.

1. Due to a hardware limitation in SAMR21 flash, there is a possibility that the entire flash row will be corrupted when a HW reset occurs during a flash page update. Hence previously stored PDS files might get corrupted and not restored.

   This limitation is addressed now in software by limiting the number of writes to a flash row section along with some optimization of the existing NVM driver. The changes done are backward compatible with previous releases of BitCloud SDK but to port a BC3.2 based application requires changes in the set of files to be compiled as part of the application projects (BCZPRO-2109 & BCZPRO-2052):

   I. `S_Nv.c` file got deleted at `…/BitCloud/Components/ZLLPlatform/ZLL/S_Nv/src/` and hence shall be removed from application projects.

   II. `S_Nv-SamR21.c` and `S_Nv-ATmega.c` files are added for SAMR21 and ATmega families respectively and need to be included as part of `/Components/ZLLPlatform/ZLL/S_Nv/src/` directory in the application project.

2. Reference application projects are ported to new versions of IAR Workbench IDE and Atmel Studio (BCZPRO-2510).

   See …/`Documentation/AVR2052_BitCloud_SDK_Quick_Start_Guide.pdf` file for actual supported versions.

3. Linker file changes which were done as part of the release are,

   I. SAMR21 specific changes

      a. `MAIN_STACK_SIZE` increased from 1000 to 1400. The list of files modified and to be looked into are as follows, (BCZPRO-2569)

         i. `atsamr21e18a.ld`

         ii. `atsamr21e18a_OTAU.ld`

         iii. `atsamr21g18a.ld`

         iv. `atsamr21g18a_OTAU.ld`

      b. Stack size reduced in order to compile applications with OTAU enabled. `__ICFEDIT_size_cstack__` value changed from `800` to `400`. The list of files modified and to be looked into are as follows,

         i. `atSAMR21E18A_FLASH_OTAU.icf`

         ii. `atSAMR21G18A_FLASH_OTAU.icf`

   II. MEGARF specific changes

      a. `-D_..X_CSTACK_SIZE` in `lnkm256rfr2s.xcl` file of Atmega256rfr2 modified from 800 to 700.(BCZPRO-2086).

2.2 Items that might impact application behavior if corresponding functionality is used

This subsection describes items in BC3.3 that change the default behavior of the stack and hence might impact application if it is using corresponding functionality. Hence user might need to make certain changes to adapt to the new behavior or return to the same behavior as was in BC3.2 if that is desired.

There are also ZLLDemo-specific items covered in Section 3.2.

4. Default clock system on SAMR21 is modified in BC3.3 release (BCZPRO-2034).

   Clock system can be configured to own preferences in .../`BitCloud/Components/HAL/Configuration` file (requires recompilation of HAL library to apply the changes).

   Now SAMR21E18/E19 devices on reset initialize from OSC8M clock and then switch to external clock from RF crystal (`HAL_CLOCK_SOURCE=CLKM`) vs. `XOSC32K` used in BC3.2.

   SAMR21E18/E19 devices also now rely by default on internal 32K crystal as asynchronous sleep clock (`HAL_ASYNC_CLOCK_SOURCE=RC_32K`) vs. external one (`CRYSTAL_32K)` used in BC3.2 as necessary pins for external crystal are not available on them.

5. Default values for `CS_MAC_TRANSACTION_TIME` and `CS_END_DEVICE_SLEEP_PERIOD` parameters are changed (BCZPRO-1564).

   Changes are done in …/`BitCloud/Components/ConfigServer/include/csDefaults.h` file.

   Default value for `CS_MAC_TRANSACTION_TIME` (max buffering time for data destined to child end device) is changed to `7680` ms vs. formula based on sleep and poll times used in BC3.2. This is to avoid that buffers gets blocked for long time when sending data to end devices that poll rare or got disconnected.

   Default value for `CS_END_DEVICE_SLEEP_PERIOD` (sleep period/parent poll interval for end devices) is changed to `7500` ms vs. `10000` ms in BC3.2. This is to adapt the default behavior to MAC transaction time.

   Application can redefine values for these parameters in own `configuration.h` file and BitCloud ZLLDemo and WSNDemo references applications do so.

6. Reduced Tx RF output power shall be used on channel 26 to meet FCC requirements (BCZPRO-1566).

   `CS_RF_MAX_CH26_TX_POWER` ConfigServer parameter is implemented to enable run time configuration of Tx power on channel 26. Default value defined in .../`BitCloud/Components/ConfigServer/include/csDefaults.h` file is -12 dBm. BitCloud MAC/PHY Component automatically applies the value set for the Tx power when Channel 26 is used and then restores to the `CS_RF_TX_POWER` power when other channels are used.

7. Calculation of PWM TOP value was wrong (hence leading to wrong PWM frequency) and is corrected now (BCZPRO-1075).

   Changes are done as part of `.../BitCloud/Components/HAL/include/pwm.h` file.

8. PWM frequency on SAMR21 didn't match with the `APP_PWM_FREQUENCY` configuration and is corrected (BCZPRO-944).

   Changes are done as part of `.../BitCloud/Components/BSP/SAMR21/src/leds.c` file.

2.3 Newly added features that require changes in BC3.2 applications to be used:

9. USB driver is added for SAMR21 devices (BCZPRO-1965).

   BitCloud reference applications for SAMR21 have now a USB as option for serial interface in their `configuration.h` files. It can be used by selecting `#define APP_INTERFACE APP_INTERFACE_USBCDC.`

   USB functionality is primarily implemented in HAL component and becomes automatically available when using HAL files and library from BC3.3. Only file `…/BitCloud/Components/BSP/SAMR21/src/bspUsbVbus.c` shall be added to the application project to handle board-specific USB configuration (primarily for handling VBus).

   Use of USB in reference applications requires changes in `uartManager.h` and `uartManager.c` files. USB specific changes in these files can be found under checks for `APP_INTERFACE_USBCDC` compile switch.

10. Added Pin Change interrupt support (PCINT) for megaRFR2 as part of HAL (BCZPRO-1768).

    The APIs are declared as part of `../Components/HAL/include/irq.h` and implemented in `…/BitCloud/Components/HAL/avr/atmega128rfa1/common/src/halIrq.c` file. Interrupt handling routines are defined in that file and hence will conflict with custom implementation in application if such is present. It is recommended to change to BitCloud API:

    ```
    int HAL_RegisterPCIrq(HAL_PCIrqNumber_t irqNumber, uint8_t pinMask, void (*f)(void))
    – To register
    int HAL_UnregisterPCIrq(HAL_PCIrqNumber_t irqNumber) – To unregister
    int HAL_EnablePCIrq(HAL_PCIrqNumber_t irqNumber) – To enable
    int HAL_DisablePCIrq(HAL_PCIrqNumber_t irqNumber) – To disable
    ```

11. SAMR21G18 Module added as part of BitCloud SDK and ported applications onto it. The BSP of the same is supported under `BOARD_SAMR21G18_MR210UA_MODULE` switch. (BCZPRO-2260). The list of files modified are

    ```
    ../Components/BSP/SAMR21/include/bspLeds.h
    ../Components/BSP/SAMR21/src/bspSpi.c
    ../Components/BSP/SAMR21/src/bspUid.c
    ../Components/BSP/SAMR21/src/bspUsart.c
    ../Components/BSP/SAMR21/src/leds.c
    ```

12. SAMR21B18 Module added as part of BitCloud SDK and ported applications onto it. The BSP of the same is supported under BOARD_SAMR21B18_MR210UA_MODULE switch. (BCZPRO-2259). The list of files modified are

    ```
    ../Components/BSP/SAMR21/src/bspSpi.c
    ../Components/HAL/drivers/OFD/include/ofdMemoryDriver.h
    ../Components/HAL/drivers/OFD/src/ofdMx25l2006eDriver.c
    ../Components/HAL/drivers/OFD/src/ofdSpiSerializer.c
    ```

13. If Antenna Diversity feature is not enabled then board specific antenna selection shall be possible in SAMR21 BSP (BCZPRO-2812).

    The option of selecting specific antenna is implemented in BSP as board-specific functionality. This requires function `bool BSP_BoardSpecificAntennaDivesityPinInit(uint8_t* antennaSelected)` to be defined and implemented as it now always gets called from HAL library (in `HAL_InitAntennaDiversity` function in `halRfCtrl.c` file) independent on antenna diversity configuration.

    For Atmel development boards and modules this function is implemented now in a new file - `BitCloud/Components/BSP/SAMR21/src/bspAntDiversity.c` that shall be added to the application project. For custom boards `(BOARD_SUPPORT` set to `BOARD_FAKE) fakeBSP.c` is updated to include this function.

    Antenna selection is done based on `BSP_ANTENNA_SELECTED` define that is set to `1` by default and can be redefined in `configuration.h` file.

3 Relevant only to ZLLDemo application

BitCloud stack and ZLLDemo application code are updated to support new features and address certain bug-fixes. It is generally strongly recommended to reuse updated ZLLDemo application code for own ZLL applications. This section lists all the ZLLDemo-specific modifications applied in addition to generic ones given in Chapter 2.

3.1 Items that require mandatory changes when porting ZLLDemo from BC3.2

This subsection describes changes that are mandatory to do only when porting BitCloud v3.2 ZLLDemo application to BC v3.3 SDK. Other mandatory changes that a required for all BC3.2 reference applications (incl. ZLLDemo) are described in Section 2.1 and shall be applied in addition to those listed below.

14. `N_Hac` component is deleted and its functionality is moved into `N_PacketDistributor` (BCZPRO-2006).

To simplify endpoint registration and remove unnecessary redundancy `N_Hac` functionality is moved to `N_PacketDistributor` so corresponding API shall be used in applications as well. Now `N_PacketDistributor` would check for endpoint registrations for a ZLL profile ID and makes sure that ZHA profile ID is also accepted on the same endpoint.

To support this update following changes are needed in the application:

I. Remove `/Components/ZLLPlatform/N_Hac/` folder (includes `N_Hac.h, N_Hac_Bindings.h` and `N_Hac.c` files) from application project.

II. Remove line

```
#define N_Hac_RegisterEndpoint N_Hac_RegisterEndpoint_Impl
```

in `colorSceneRemoteBindings.h`, `lightBindings.h` and `bridgeBindings.h` files.

III. Remove line

```
#include <N_Hac.h>
```

in `colorSceneRemote.c, light.c` and `bridge.c` files.

IV. Replace `N_Hac_RegisterEndpoint` function used in `colorSceneRemote.c, light.c` and `bridge.c` files with `N_PacketDistributor_RegisterEndpoint` function using same arguments.

15. `N-Radio_Disable` functionality added as part of `N_Radio` component and extends the list of task handlers (BCZPRO-2518).

    An event handler for disabling the radio is introduced in `N_Radio.c` and the same must be added as part of `N_Task_HandleEvent_t s_taskArray[]` definition in `colorSceneRemote.c`, `light.c` and `bridge.c` files. For example for the `colorSceneRemote.c`:

```
static const N_Task_HandleEvent_t s_taskArray[] =
{
  ColorSceneRemote_EventHandler,
  N_ConnectionEndDevice_EventHandler,
  N_LinkInitiator_EventHandler,
  N_DeviceInfo_EventHandler,
  N_LinkTarget_EventHandler,
  N_ReconnectHandler_EventHandler,
  N_Radio_EventHandler, // newly added in BC3.3
};
```

16. Definition of `TOUCHLINK_ZERO_DBM_TX_POWER` parameter is moved to application level (BCZPRO-2031).

    Touchlink operation requires 0dBm Tx power but actual setting for the transceiver is generally board-specific depending on the chip as well as RF design (presence of power amplifier, etc). To simplify this configuration for users definition of corresponding parameter `TOUCHLINK_ZERO_DBM_TX_POWER` is moved from stack level (`N_DeviceInfo.c`) to application files `light.c, colorSceneRemote.c` and `bridge.c`. Hence ZLLDemo applications that are being ported from BC3.2 require following code to be added in these files as well (although actual values can be different if needed) :

```
#if  defined(ATSAMR21G18A) || defined(ATSAMR21E18A) || defined(ATSAMR21E19A)
#define TOUCHLINK_ZERO_DBM_TX_POWER          0x07u
#elif defined(ATMEGA256RFR2) || defined(ATMEGA2564RFR2)
#define TOUCHLINK_ZERO_DBM_TX_POWER          0x06u
#endif
```

3.2 Items that might require changes if corresponding functionality is used

17. `N_Zdp_GetSourceAddress` function is updated to return short address in some cases (BCZPRO-2046).

    The behavior now would be that if a device receives a message via local-loopback (sent by itself) and if the device is factory new it shall return IEEE address otherwise it would return short address. Changes are done in `fillAddressStructure` function of `N_Zdp.c` file. ZLLDemo reference application doesn't use this API but updated behavior should be considered if custom application relies on this API.

3.3 Newly added features that require changes in BC3.2 applications to be used:

Generic features that apply to all BitCloud reference applications incl. ZLLDemo are listed in Section 2.2.

18. Changes from ZLL spec (CCBs) are addressed as part of BC3.3 release (BCZPRO-2163).
    I. CCB#1608: Rate field may be zero in case of stop move mode (BCZPRO-2183)
        a. Changes are done in `moveHueInd` function in `lightColorControlCluster.c` file.
    II. CCB#1987: Inconsistency in Color Temperature Light Capabilities (BCZPRO-2182).
        a. Changes are done in `moveToColorTemperatureInd` function in `lightColorControlCluster.c` file.
    III. CCB#2012: Frequency Agility for ZLL network (secondary channel) (BCZPRO-2185)
        a. Changes done in `buttonHandler` function in `buttonHandlers.c`
    IV. CCB#2032: Color Temperature Functionality now certifiable (BCZPRO-2186).
        a. Certification script `3.21.py` is changed `at ../Evaluation Tools/ZLL_Scripts/`
    V. CCB#2011: Changes ZLL Test Specifications Comments from Color Temp SVE (BCZPRO-2184)
        a. Test Case `3.12.py` modified `at` ../Evaluation Tools/ZLL_Scripts/

19. `QTouch functionality is added f`or ZLL Color Scene Remote Controller (BCZPRO-670, BCZPRO-1967 & BCZPRO-1407).

    The feature is added only for ZLL-EK platform as it has Touch Buttons and Slider on board.

    Changes done as part of `colorSceneRemote.c` which would demonstrate Touchlink, ON/OFF and brightness increase/decrease from remote to light. BSP of ZLL-EK is also extended to support Touch feature appropriately by modifying its BSP and added few Touch specific files to the release..

    List of files modified/added as part of this feature are
    I. `qTouch.c, touch.c, qTouch.h, touch.h` added as part of `../Components/BSP/SAMR21`
    II. `Modified appTouchInd()as part of colorSceneRemote.c` file.

20. Console command for `powerOff` is added as part of `colorSceneRemoteConsole.c` even when `APP_ENABLE_CERTIFICATION_EXTENSION` is disabled (BCZPRO-2026).

21. Few references which were made to `pSimpleDescription` as part of previous release got removed. The changes are done in the following files, (BCZPRO-2309).
    I. `N_PacketDistributor.h , N_PacketDistributor.c` files @ `../ZLLDemo/common/include/` and `../ZLLDemo/common/src/` respectively got modified.
    II. `N_PacketDistributor.h , N_PacketDistributor.c` files @ `../Components/ZLLPlatform/ZLL/include/` and `../Components/ZLLPlatform/ZLL/src/` respectively got modified.

4 Relevant only to HADevice application

BitCloud stack and HADevice application are updated to support new features and address certain bug-fixes. It is generally strongly recommended to reuse updated HADevice application code for own ZHA applications. This section lists all the HADevice-specific modifications applied in addition to generic ones given in Chapter 2.

4.1 Items that require mandatory changes

Mandatory changes that a required for all BC3.2 reference applications (incl. HADevice) are described in Section 2.1. There're no other HADevice specific mandatory changes.

4.2 Items that might require changes if corresponding functionality is used

Only those covered in Section 2.2.

4.3 Newly added features that require changes in BC3.2 applications to be used:

22. EZMode Commissioning Procedure- Optimization: (BCZPRO-1966, BCZPRO-1956).
    I. The behavior of sending Match descriptor requests followed by IEEE address request for each cluster is optimized.
        a. This is done in such way that only one match descriptor requests with all the input and Output clusters would be sent from EZ-mode initiator.
        b. Target would actually respond with match descriptor contain endpoint(s) that have matching clusters.
        c. Initiator after receiving will send SimpleDescriptor requests and gets the actual list of clusters to be bound.
        d. Initiator then sends IEEE address requests followed by the bind requests based on the Simple Descriptor response.
    II. Files `ezModeManager.c, msConsole.c, ciSOncole.c, dlConsole.c, dsCOnsole.c, thCOnsole.c, dimmableLight.c, multisensor.c, thermostat.c, zcl.c and zdoCommon.h` got changed to address/enhance the feature.
    III. List of changes done as part of ezModeManager.c file,
        a. List of new functions added,
            i. `static void doSimpleDescReq(void)`
            ii. `static void zdpSimpleDescResp(ZDO_ZdpResp_t *resp)`
            iii. `static void continueWithApsZdoBinding(void)`
            iv. `static void continueWithOnlyApsBinding(void)`
            v. `static void clearMatchedClsterList(void)`
        b. New structure addition,

```
typedef struct
{
  uint8_t      AppInClustersCount;
  ClusterId_t  AppInClustersList[MAX_CLUSTERS_AMOUNT_TO_BIND];
  uint8_t      AppOutClustersCount;
  ClusterId_t  AppOutClustersList[MAX_CLUSTERS_AMOUNT_TO_BIND];
} MatchedClusterList_t;
```

        c. List of function definitions which got modified,
            i. `static void doApsAndZdoBinding(ExtAddr_t *remoteDevExtAddr)`
               to
               `static void doApsAndZdoBinding(void)`
            ii. `static bool doApsBinding(ExtAddr_t *ownExtAddr, ExtAddr_t *remoteDevExtAddr, Endpoint_t dstEndpoint)`
               to
               `static bool doApsBinding(void)`
        d. List of functions which got removed,
            i. `static void bindNextCluster(void)`
        e. List of functions which got modified,
            i. In `commissionNextEndpoint()` function, `bindNextCluster()` got replaced with `matchBindDevices()`
            ii. In `matchBindDevices()` function, the process of sending match descriptor request for each cluster is modified to send only one match descriptor with all

input and output clusters. The changes which are done as part of `matchBindDevices()` function are,

```
  if (!ezModeMem.client)
  {
    matchDescReq->numInClusters    = 1;
    matchDescReq->numOutClusters   = 0;
    matchDescReq->inClusterList[0] = appBindReq->
   remoteServers[ezModeMem.clusterNumber];
   ezModeMem.clusterId       = matchDescReq-
   >inClusterList[0];
  }
  else
  {
    matchDescReq->numInClusters     = 0;
    matchDescReq->numOutClusters    = 1;
    matchDescReq->outClusterList[0] = appBindReq->
   remoteClients[ezModeMem.clusterNumber];
    ezModeMem.clusterId             = matchDescReq-
>outClusterList[0];
  }
```

to

```
  matchDescReq->numInClusters = appBindReq-
>remoteServersCnt;
  for (uint8_t i = 0; i < appBindReq->remoteServersCnt;
i++)
    matchDescReq->inClusterList[i] = appBindReq->
   remoteServers[i];
  matchDescReq->numOutClusters = appBindReq-
>remoteClientsCnt;
  for (uint8_t i = 0; i < appBindReq->remoteClientsCnt;
i++)
    matchDescReq->outClusterList[i] = appBindReq->
   remoteClients[i];
```

iii. In `zdpMatchDescResp()` function , once the match descriptor response received and if the status is `(ZDO_SUCCESS_STATUS != status)` `bindNextCluster()` function is replaced with `commissionNextEndpoint`. The changes are as follows,

```
  else if (ZDO_SUCCESS_STATUS != status)
    bindNextCluster();
```

to

```
  else if (ZDO_SUCCESS_STATUS != status)
  {
    ezModeMem.endpointNumber++;
    commissionNextEndpoint();
```

```
}
```

iv. Function `matchHandlingFinished()` got revamped totally as we do not call `doIeeeAddrReq ()` and `bindNextCluster()` function anymore.

v. Function `zdpIeeeAddrResp()` got revamped totally with respect to the binding procedure for the matched clusters.

vi. Handling of `zdpBindResp()` changed for the binding procedure.

IV. List of changes done as part of `zdoCommon.h` file.

a. `MAX_REQUESTED_CLUSTER_NUMBER` modified from `9` to `17` as part of the release.

V. List of changes done as part of `msConsole.c`, `ciConsole.c` , `dlConsole.c` , `dsConsole.c`, `thConsole.c` files.

a. `invokeEzMode(NULL)` changed to `invokeEzMode(appEzModeDone)` as part of `processStartEzModeCmd(..)` function

VI. List of changes done as part of `dimmableLight.c` file.

a. `ZDO_BindIndication()` function changed in order to set the `dlClustersBoundMask` for all the clusters which were bound successfully.

b. `updateCommissioningStateCb ()` ad `dlConfigureReportingResp()` function changed internally by introducing a new function (`handleDlConfigReporting()`) to handle configure reporting after commissioning.

VII. List of changes done as part of multiSensor.c file.

a. Callback function changed from NULL to `msBindingFinished` for the list of structures below,

```
   i. Static AppBindReq_t osBindReq
  ii. static AppBindReq_t tsBindReq
 iii. static AppBindReq_t hsBindReq
  iv. static AppBindReq_t lsBindReq
```

b. `enum _ReportingState_t` modified by introducing a new item, `CONFIG_REPORT_INIT_STATE = 0` as a first element .

c. Introduced a new `enum state msBoundClustersInEzMode_t` to check the number of bound clusters

d. Introduced a static global variable `msClustersBoundMask` to mask all the bounded clusters in order to handle configure reporting after commissioning

e. `appEzModeDone()` and `msConfigureReportingResp()` functions modified internally by introducing a new function `handleMsConfigReporting` to handle configure reporting after commissioning

f. `msBindingFinished()` function newly added in order to set the `msClustersBoundMask` variable for all the clusters which were bound successfully.

VIII. List of changes done as part of `thermostat.c` file.

a. enum _ReportingState_t modified by introducing new element and renaming an existing element.

i. In BC 3.2, `_ReportingState_t` had the following elements,

```
typedef enum _ReportingState_t
{
  THERMOSTAT_REPORTING_CONFIGURED = 0,
  OCCUPANCY_REPORTING_CONFIGURED,
  HUMIDITY_MEASURED_VALUE_REPORTING_CONFIGURED,
  HUMIDITY_TOLERANCE_VALUE_REPORTING_CONFIGURED,
  TEMPERATURE_MEASURED_VALUE_REPORTING_CONFIGURED,
  TEMPERATURE_TOLERANCE_VALUE_REPORTING_CONFIGURED,
  CONFIGURATION_COMPLETED
}ReportingState_t;
```

ii. In BC 3.3 , `ReportingState_t` will have the following elements,

```
typedef enum _ReportingState_t
{
  CONFIG_REPORT_INIT_STATE = 0,
  THERMOSTAT_REPORTING_CONFIGURED,
  OCCUPANCY_REPORTING_CONFIGURED,
  HUMIDITY_MEASURED_VALUE_REPORTING_CONFIGURED,
  HUMIDITY_TOLERANCE_VALUE_REPORTING_CONFIGURED,
  TEMPERATURE_MEASURED_VALUE_REPORTING_CONFIGURED,
  TEMPERATURE_TOLERANCE_VALUE_REPORTING_CONFIGURED,
  CONFIGURE_REPORTING_COMPLETED
}ReportingState_t;
```

b. Introduced a new enum state `_thBoundClustersInEzMode_t` to check the number of bound clusters

c. Introduced a static global variable `thClustersBoundMask` to mask all the bounded clusters in order to handle configure reporting after commissioning

d. `updateCommissioningStateCb ()` ad `thConfigureReportingResp ()` function changed internally by introducing a new function (`handleThConfigReporting ()`) to handle configure reporting after commissioning.

e. `ZDO_BindIndication()` function changed in order to set the `thClustersBoundMask` for all the clusters which were bound successfully

IX. List of changes done as part of zcl.c file.

a. In ZCL_StartReporting() function an extra check added for report configuration. Idea is to configure reporting only if it was not configured before.

`if (attr->properties & ZCL_REPORTABLE_ATTRIBUTE)` to `if ((attr->properties & ZCL_REPORTABLE_ATTRIBUTE) && !(attr->properties & ZCL_REPORTING_CONFIGURED))`

23. OTAU cluster on IAS ACE device corrected to be registered as client (BCZPRO-2162).

    Files `iasACECluster.h` and `iasACECluster.c` got changed to address the issue.

24. Console command support for `getAppDeviceType` added as part of `ciConsole.c, dlConsole.c, dsConsole.c, iasACEConsole.c, msConsole.c, thConsole.c` (BCZPRO-1995

25. Console command support for `SimpleDescReg and MatchDescReq` added as part of `ciConsole.c, haClusters.c, dsConsole.c, msConsole.c` (BCZPRO-1958).

26. Console Command added for `getNetworkAddress` as part of `dlConsole.c` (BCZPRO-1957).