Release Notes

Product: Application Framework for EmberZNet 5.0.0
Release Date: May 1, 2013

1 Overview

This release contains the Beta 1 release of Application Framework for use with EmberZNet 5.0 for the EM250, EM260, or EM35x. This release must be used with InSight AppBuilder 3.0.716 or greater.

2 Getting Started

The source code for Application Framework is bundled with the EmberZNet 5.0 installer. Once the stack has been installed, follow these steps to add the stack to Ember AppBuilder.

1. Start Ember Desktop by selecting **Start** ⇒ **All Programs** ⇒ **Ember** ⇒ **Ember Desktop** ⇒ **Ember Desktop**.
2. Open the preferences panel by selecting **File** ⇒ **Preferences** from the Ember Desktop menu.
3. Select **AppBuilder (ZCL)** from the list on the left of the preferences panel.
4. Click the **Add** button and browse to the directory where EmberZNet 5.0 is installed (e.g., C:\Users\Administrator\Ember\EmberZNet5.0\em35x).
5. Click the **OK** button to save the changes and close the preferences panel.

Once the stack is loaded, Application Framework projects can be created and configured using Ember AppBuilder.

1. Select **File** ⇒ **New** ⇒ **Application Configuration** from the Ember Desktop menu.
2. Select the appropriate stack for your application (e.g., EmberZNet 5.0 EM35X).
3. Click the **OK** button.

2.1 Support

Development Kit customers are eligible for training and technical support. You can use the Silicon Laboratories web site http://www.silabs.com/zigbee to obtain information about all Silicon Laboratories ZigBee products and services and to sign up for product support. You can contact Silicon Labs technical support at http://www.silabs.com/zigbee-support.

3 The 5.0 Release

This release includes a number of new features, improvements, and bug fixes. Customers upgrading from previous releases are strongly encouraged to carefully review this section. It is important to become with familiar with the changes in this release before attempting to migrate applications.

3.1 New Features

This release includes a number of new features and enhancements over previous releases. Customers upgrading from previous releases should review the new features to determine how they affect existing applications.

3.1.1 NCP Support for ZigBee Light Link

This release introduces support for the ZigBee Light Link (ZLL) profile on the EM35x network coprocessor (NCP). All 35x NCP images now include stack support for ZLL and customers are now able to use Ember AppBuilder to configure and generate host applications with ZLL functionality. All customers building ZLL applications, whether on system-on-a-chip (SoC) or host platforms, are strongly encouraged to use the ZLL Commissioning plugin.

3.1.2 Runtime ZLL

With this release, customers are afforded the ability to have a device determine at runtime whether or not to behave as a ZLL device (i.e. a device may behave as a ZLL device while on a ZLL network, and a non-ZLL device while on a non- ZLL network.) Any application for which this functionality is desired must include the ZLL Commissioning plugin.

3.2 Configuration Changes

The manner in which the HAL is configured has changed slightly in this release. The process for selecting and configuring the board header has been improved, as described in section 3.2.1. This release also includes new bootloader choices and a change to the default choice as described in section 3.2.2.

3.2.1 Board Header

New widgets on the "HAL configuration" tab in Ember AppBuilder allow customers to choose from a selection of predefined board header files or to specify a custom board header file to use with the application. For EM250, EM351, and EM357 system-on-a-chip (SoC) platforms, the familiar GPIO editors are also available and allow customers to configure a template version of the standard board headers for these platforms.

An important change in this release is that Ember AppBuilder will no longer overwrite an existing board header file when generating the files for the application. Customers must now explicitly choose to overwrite the existing file. This protection of existing files is similar to how AppBuilder handles the application's callbacks file and is intended to prevent accidental data loss when generating.

Note that there are many options within the board header file for configuring GPIO differently for the needs of your board. Customers are strongly encouraged to review the generated board file and make changes as needed for the target hardware.

3.2.2 Bootloader

This release includes support for a secure application bootloader on the EM35x system-on-a-chip (SoC) platforms. Customers using these platforms can increase security by encrypting their binary images.

Additionally, the default bootloader for new EM35x SoC applications has been changed. Previously, no bootloader was selected by default. (This type of setup is also known as a "null bootloader" configuration.) The new default for EM35x SoC applications is the application bootloader, which more accurately reflects what is used in typical customer applications. Customers creating new applications should verify the bootloader selection is appropriate for intended application.

3.3 Removed APIs

The following APIs have been removed from this release. Customer who are upgrading from a previous release and are using any of these APIs must update their applications to use supported APIs.

3.3.1 ADDITIONAL_BITMASK_OPTIONS

`ADDITIONAL_BITMASK_OPTIONS` was a macro that allowed the application to add options to the default initial security bitmask used by the framework when forming or joining networks. The macro did not allow the application to specify separate options for each network or to adapt the options at runtime. Because of

these limitation, the macro has been replaced by `emberAfSecurityInitCallback`, which provides similar functionality with the added flexibility of allowing per-network options and runtime changes.

The companion macro `CLEARED_BITMASK_OPTIONS` has also been removed, as described in section 3.3.3.

3.3.2 ADDITIONAL_EXTENDED_BITMASK_OPTIONS

`ADDITIONAL_EXTENDED_BITMASK_OPTIONS` was a macro that allowed the application to add options to the default extended security bitmask used by the framework when forming or joining networks. The macro did not allow the application to specify separate options for each network or to adapt the options at runtime. Because of these limitation, the macro has been replaced by `emberAfSecurityInitCallback`, which provides similar functionality with the added flexibility of allowing per-network options and runtime changes.

The companion macro `CLEARED_EXTENDED_BITMASK_OPTIONS` has also been removed, as described in section 3.3.4.

3.3.3 CLEARED_BITMASK_OPTIONS

`CLEARED_BITMASK_OPTIONS` was a macro that allowed the application to remove options from the default initial security bitmask used by the framework when forming or joining networks. The macro did not allow the application to specify separate options for each network or to adapt the options at runtime. Because of these limitation, the macro has been replaced by `emberAfSecurityInitCallback`, which provides similar functionality with the added flexibility of allowing per-network options and runtime changes.

The companion macro `ADDITIONAL_BITMASK_OPTIONS` has also been removed, as described in section 3.3.1.

3.3.4 CLEARED_EXTENDED_BITMASK_OPTIONS

`CLEARED_EXTENDED_BITMASK_OPTIONS` was a macro that allowed the application to remove options from the default extended security bitmask used by the framework when forming or joining networks. The macro did not allow the application to specify separate options for each network or to adapt the options at runtime. Because of these limitation, the macro has been replaced by `emberAfSecurityInitCallback`, which provides similar functionality with the added flexibility of allowing per-network options and runtime changes.

The companion macro `ADDITIONAL_EXTENDED_BITMASK_OPTIONS` has also been removed, as described in section 3.3.2.

3.3.5 emberAfHibernateDuration

`emberAfHibernateDuration` was a variable used to maintain the long polling interval. Each network's polling intervals are configured independently, so a single variable is no longer appropriate. In addition, the sleep-specific and unit-less naming of the variable caused confusion. Because of these limitations, the variable has been removed and replaced by `emberAfGetLongPollIntervalQsCallback` and `emberAfSetLongPollIntervalQsCallback`.

A similar change has been made to `emberAfNapDuration`, as described in section 3.3.6.

3.3.6 emberAfNapDuration

`emberAfNapDuration` was a variable used to maintain the short polling interval. Each network's polling intervals are configured independently, so a single variable is no longer appropriate. In addition, the sleep-specific and unit-less naming of the variable caused confusion. Because of these limitations, the variable has been removed and replaced by `emberAfGetShortPollIntervalQsCallback` and `emberAfSetShortPollIntervalQsCallback`.

A similar change has been made to `emberAfHibernateDuration`, as described in section 3.3.5.

3.3.7 emberAfReadOrUpdateAttribute

`emberAfReadOrUpdateAttribute` was a low-level function for reading and writing attributes. It did not verify the destination buffer was large enough to accommodate the source data or perform type checking. Because

of this limitation, the function has been removed. Customers should use the higher-level `emberAfRead` and `emberAfWrite` APIs, which perform size and type checking and also provide a friendlier interface.

3.3.8 emberAfRetrieveAttribute

`emberAfRetrieveAttribute` was a function for reading attributes. It did not verify that the destination buffer was large enough to accommodate the attribute data and only provided a simple boolean indication of whether the read succeeded or failed, which was usually insufficient for use on production devices. Because of these limitations, the function has been removed. Customers should use the higher-level `emberAfRead` APIs or `emberAfLocateAttributeMetadata` instead.

The companion function `emberAfUpdateAttribute` has also been removed, as described in section 3.3.9.

3.3.9 emberAfUpdateAttribute

`emberAfUpdateAttribute` was a function for writing attributes. It only provided a simple boolean indication of whether the write succeeded or failed, which was usually insufficient for use on production devices. Because of this limitation, the function has been removed. Customers should use the higher-level `emberAfWrite` APIs instead.

The companion function `emberAfRetrieveAttribute` has also been removed, as described in section 3.3.8.

3.3.10 EMBER_AF_SET_HIBERNATE_DURATION

`EMBER_AF_SET_HIBERNATE_DURATION` was a macro used to set the long polling interval. As with `emberAfHibernateDuration`, as described in section 3.3.5, the sleep-specific and unit-less naming of the macro has caused confusion. Because of this, the macro has been removed and replaced by `emberAfSetLongPollIntervalQsCallback`.

`EMBER_AF_SET_NAP_DURATION`, as described in section 3.3.11, has also been removed.

3.3.11 EMBER_AF_SET_NAP_DURATION

`EMBER_AF_SET_NAP_DURATION` was a macro used to set the short polling interval. As with `emberAfNapDuration`, as described in section 3.3.6, the sleep-specific and unit-less naming of the macro has caused confusion. Because of this, the macro has been removed and replaced by `emberAfSetShortPollIntervalQsCallback`.

`EMBER_AF_SET_HIBERNATE_DURATION`, as described in section 3.3.10, has also been removed

3.3.12 EMBER_AF_SECURITY_PROFILE

`EMBER_AF_SECURITY_PROFILE` was a generated macro that configured the security profile used by the device. The security profile is now configured for each network and a single macro is no longer appropriate. Because of this, the macro has been removed.

3.3.13 haNumPktsRxAll

`haNumPktsRxAll` was a variable used to maintain a count of the number of packets received by the framework. The counter was not maintained and has been removed. The macro `INCR_haNumPktsRxAll`, which incremented the counter, has also been removed, as described in section 3.3.18.

3.3.14 haNumPktsRxGivenToZCL

`haNumPktsRxGivenToZCL` was a variable used to maintain a count of the number of ZigBee cluster library (ZCL) messages received by the framework. The counter was not maintained and has been removed. The macro `INCR_haNumPktsRxGivenToZCL`, which incremented the counter, has also been removed, as described in section 3.3.19.

3.3.15 haNumPktsRxLengthError

`haNumPktsRxLengthError` was a variable used to maintain a count of the number of messages received by the framework with invalid lengths. The counter was not maintained and has been removed. The macro `INCR_haNumPktsRxLengthError`, which incremented the counter, has also been removed, as described in section 3.3.20.

3.3.16 haNumPktsRxWrongEndpoint

`haNumPktsRxWrongEndpoint` was a variable used to maintain a count of the number of messages received by the framework for invalid endpoints. The counter was not maintained and has been removed. The macro `INCR_haNumPktsRxWrongEndpoint`, which incremented the counter, has also been removed, as described in section 3.3.21.

3.3.17 haNumPktsRxWrongProfile

`haNumPktsRxWrongEndpoint` was a variable used to maintain a count of the number of messages received by the framework for invalid profiles. The counter was not maintained and has been removed. The macro `INCR_haNumPktsRxWrongProfile`, which incremented the counter, has also been removed, as described in section 3.3.22.

3.3.18 INCR_haNumPktsRxAll

`INCR_haNumPktsRxAll` was a macro used to increment `haNumPktsRxAll`, a counter tracking the number of packets received by the framework. The counter was not maintained and has been removed, as described in section 3.3.13. The macro, therefore, has been also been removed.

3.3.19 INCR_haNumPktsRxGivenToZCL

`INCR_haNumPktsRxGivenToZCL` was a macro used to increment `haNumPktsRxGivenToZCL`, a counter tracking the number of ZigBee cluster library (ZCL) messages received by the framework. The counter was not maintained and has been removed, as described in section 3.3.14. The macro, therefore, has been also been removed.

3.3.20 INCR_haNumPktsRxLengthError

`INCR_haNumPktsRxLengthError` was a macro used to increment `haNumPktsRxLengthError`, a counter tracking the number of messages received by the framework with invalid lengths. The counter was not maintained and has been removed, as described in section 3.3.15. The macro, therefore, has been also been removed.

3.3.21 INCR_haNumPktsRxWrongEndpoint

`INCR_haNumPktsRxWrongEndpoint` was a macro used to increment `haNumPktsRxWrongEndpoint`, a counter tracking the number of messages received by the framework for invalid endpoints. The counter was not maintained and has been removed, as described in section 3.3.16. The macro, therefore, has been also been removed.

3.3.22 INCR_haNumPktsRxWrongProfile

`INCR_haNumPktsRxWrongProfile` was a macro used to increment `haNumPktsRxWrongProfile`, a counter tracking the number of messages received by the framework for invalid profiles. The counter was not maintained and has been removed, as described in section 3.3.17. The macro, therefore, has been also been removed.

3.3.23 ZA_DEVICE_TYPE

`ZA_DEVICE_TYPE` was a generated macro that configured the ZigBee device type used by the application. The device type is now configured for each network and a single macro is no longer appropriate. Because of this, the macro has been removed.

3.4 Removed Callbacks

No callbacks were removed in this release.

3.5 Removed CLI Commands

No CLI commands were removed in this release.

3.6 Removed Plugins

The following plugins have been removed from this release. Customers who require the functionality these plugins provided should enable the appropriate callbacks in Ember AppBuilder and implement equivalent functionality in their application directly.

3.6.1 Time Client Plugin

The Time Client plugin provided basic timekeeping to the application and the framework and was useful during development when a more accurate clock was not available. The plugin previously implemented the following callback:

- `emberAfTimeClusterClientInitCallback`

However, the plugin was not actually an implementation of the Time cluster client. Despite this, the plugin could not be used on devices that did not include the cluster. The incorrect naming and unnecessary restriction caused confusion. Because of this, the Time Client plugin has been removed and replaced by the Simple Clock plugin, which has a friendlier name, provides the same basic functionality, and also supports multiple networks. Customers who require basic timekeeping functionality must implement it in their application directly or enable the Simple Clock plugin.

3.7 Changed APIs

No APIs were changed in this release.

3.8 Changed Callbacks

The following callbacks have been changed in this release. Customers who use these callbacks must update their applications appropriately.

3.8.1 emberAfExternalAttributeReadCallback

`emberAfExternalAttributeReadCallback` is called when the framework needs to read an attribute that is handled externally by the application. In the past, return values were limited to either `TRUE` or `FALSE`, which necessarily limited the range of statuses that the framework would send in response to over-the-air attribute requests. To allow more flexibility, the return type of the callback has been changed to `EmberAfStatus`. When this callback is called due to an over-the-air attribute read, the framework will now respond to the remote node with the status returned by the callback.

Customers who implemented this callback should update their application to reflect the new signature. A similar change was made to `emberAfExternalAttributeWriteCallback`, as described in section 3.8.2.

3.8.2 emberAfExternalAttributeWriteCallback

`emberAfExternalAttributeWriteCallback` is called when the framework needs to write an attribute that is handled externally by the application. In the past, return values were limited to either `TRUE` or `FALSE`, which necessarily limited the range of statuses that the framework would send in response to over-the-air attribute requests. To allow more flexibility, the return type of the callback has been changed to `EmberAfStatus`. When this callback is called due to an over-the-air attribute write, the framework will now respond to the remote node with the status returned by the callback.

Customers who implemented this callback should update their application to reflect the new signature. A similar change was made to `emberAfExternalAttributeReadCallback`, as described in section 3.8.1.

3.8.3 emberAfAllowNetworkWriteAttributeCallback

`emberAfAllowNetworkWriteAttributeCallback` is called by the framework before it process an over-the-air attribute write request. The return value of the callback determines how the framework will handle the request. In previous releases, customers could not specify a reason for rejecting the write, which necessarily limited the range of statuses that the framework would send in response to the request. To provide greater flexibility, new values were added to the `EmberAfAttributeWritePermission` enumeration. Customers who implemeted this callback should review their application to determine if any of the new return values are more appropriate.

3.9 Changed CLI Commands

The following CLI commands were changed in this release:

3.9.1 OTA CLI

OTA CLI commands have been migrated to their respective plugins for this release. Listed below are the individual command changes.

Old Command	New Command
ota server notify	plugin ota-server notify
ota server upgrade	plugin ota-server upgrade
ota server load-file	plugin ota-server load-file
ota server policy query	plugin ota-server policy query
ota server policy blockRequest	plugin ota-server policy blockRequest
ota server policy upgrade	plugin ota-server policy upgrade
ota server policy page-req-miss	plugin ota-server policy page-req-miss
ota server policy page-req-sup	plugin ota-server policy page-req-sup
ota server policy image-req-min-period	plugin ota-server policy image-req-min-period
ota server printImages	plugin ota-storage-common printImages
ota server delete	plugin ota-storage-common delete
ota server reload	plugin ota-storage-common reload
ota server storage-info	plugin ota-storage-common storage-info
ota client bootload	plugin ota-client bootload
ota client verify	plugin ota-client verify
ota client info	plugin ota-client info
ota client start	plugin ota-client start
ota client stop	plugin ota-client stop
ota client status	plugin ota-client status
ota client block-test	plugin ota-client block-test
ota client page-request	plugin ota-client page-request
ota client pause-at	plugin ota-client pause-at
ota client printImages	plugin ota-storage-common printImages
ota client delete	plugin ota-storage-common delete
ota client reload	plugin ota-storage-common reload
ota client storage-info	plugin ota-storage-common storage-info

3.9.2 Option CLI

Plugin-specific option CLI commands have also been moved to their respective plugins. Specifically, commands for the button joining and concentrator plugins were migrated. Listed below are the individual command changes.

Old Command	New Command
option button0	plugin button-joining button0
option button1	plugin button-joining button1
option agg	plugin concentrator agg

3.9.3 Security CLI

Commands related to the partner-link-key-exchange plugin have also been moved to that plugin. Listed below are the individual command changes.

Old Command	New Command
cbke partner	plugin partner-link-key-exchange partner
cbke allow-partner	plugin partner-link-key-exchange allow-partner

3.10 Changed Plugins

The following plugins have been changed in this release. Customers who use these plugins should review the changes to determine if their applications are affected.

3.10.1 button-joining

The button-joining plugin can now optionally use Home Automation's EZ-Mode Commissioning in place of end device binding. In order to make this work, the button-joining plugin needed to mandate the inclusion of the EZ-Mode Commissioning plugin. However, if EZ-Mode Commissioning is turned off in the plugin preferences then it is expected that the EZ-Mode code will be dead stripped from the application when the application is compiled.

3.10.2 Concentrator Support

This release includes concentrator support on network coprocessor (NCP). A new option in the Concentrator Support plugin configures the NCP to enable this feature. Note that this option only affects host platforms and is not applicable to system-on-a-chip (SOC) platforms. To enable the feature, check the box labeled "Enable concentrator support at the NCP" in the plugin options.

3.11 Deprecated APIs

No APIs were deprecated in this release.

3.12 Deprecated Callbacks

No callbacks were deprecated in this release.

3.13 Deprecated CLI Commands

Tiny CLI has been deprecated in this release.

3.14 Deprecated Plugins

No plugins were deprecated in this release.

3.15 New APIs

The following APIs were introduced in this release. Customers are encouraged to review the functionality provided by these APIs and consider using them in their applications.

3.15.1 emberAfFindClustersByDeviceAndEndpoint

We have added a new wrapper for the simple descriptor request used in service discovery. The `emberAfFindClustersByDeviceAndEndpoint` function sends out a ZigBee Device Object (ZDO) simple descriptor request to the given endpoint and parses the results into a struct of type EmberAfClusterList which is described in app/framework/include/af-types.h. The EmberAfClusterList is passed back into the given callback inside the EmberAfServiceDiscoveryResult's responseData value.

3.16 New Callbacks

The following callbacks were introduced in this release. Customers who use the Groups Server plugin but do not use the Scenes plugin should implement `emberAfScenesClusterRemoveScenesInGroupCallback`.

3.16.1 emberAfScenesClusterRemoveScenesInGroup

`emberAfScenesClusterRemoveScenesInGroupCallback` is utilized by the Scenes plugin to ensure scenes associated with a particular group are removed when the group is removed. The Groups Server plugin uses this callback any time a group is removed (up to and including when all groups are removed.)

3.17 New CLI Commands

No CLI commands were introduced in this release.

3.18 New Plugins

The following plugins were introduced in this release. All customers are encouraged to review the functionality that these plugins provide and consider including them in their applications. Customers upgrading from a previous release should pay particular attention to the new Address Table plugin, as described in section 3.18.1. This plugin replaces functionality that previously existed in the core framework. Customers who previously relied on functionality provided by this new plugin will require manual reconfiguration to continue functioning properly.

3.18.1 Address Table

This plugin provides support for managing the stack address table. In particular, it allows the framework, plugins, and the application to add, remove, and look up entries by specifying the extended unique identifier (EUI) of the remote node. Three APIs are provided:

- `emberAfAddressTableAddEntry`
- `emberAfAddressTableRemoveEntry`
- `emberAfAddressTableLookup`

On host platforms, this plugin also maintains a copy of the address table that is keep synchronized with the table on the network coprocessor (NCP). This allows the host to perform lookup operations locally and thus more efficiently.

The Address Table and the Trust Center Cache use the same underlying table on both the system-on-a-chip (SoC) and the NCP. The size of this table is a single byte, so the combined value of the "Address Table Size" and the "Trust Center Cache Size" cannot exceed 254.

3.18.2 EEPROM

A new EEPROM plugin was added. This code was previously found in app/framework/util/af-eeprom.c but is now formalized into a proper plugin. This allows it to be easily enabled by customers under the right circumstances. Existing ISC files that use the OTA Simple Storage EEPROM plugin will need to enable this plugin manually. New ISC files created will automatically enable the plugin when the OTA Simple Storage EEPROM plugin is selected.

3.18.3 EZ-Mode Commissioning

The EZ-Mode Commissioning plugin provides an implementation of Home Automation (HA) 1.2 EZ-mode commissioning for both the client and server sides. This plugin exposes two functions used to initiate EZ-Mode: `emberAfEzmodeClientCommission` and `emberAfEzmodeServerCommission`. When `emberAfEzmodeServerCommission` is called, the device is put into identify mode for the time specified in the plugin configuration. When `emberAfEzmodeClientCommission` is called, the plugin begins a state machine that runs the application through the EZ-mode process of first discovering devices in identify mode and then using ZigBee Device Object (ZDO) service discovery to match clusters. If a server cluster for the configured cluster identifier is found, a binding is created to the endpoint from client to server.

4 Known/Fixed Issues

4.1 Fixed Issues

- Case 12251: Messaging, Price, and DRLC plugins were not allowing for clients to track the duration of events that have a "now" start time.

- Case 14772: AppBuilder should automatically enable Gateway Support plugin for UART host platform targets.
- Case 14774: Reporting Plugin: Reports aren't always sent properly at maximum interval.
- Case 14775: AppBuilder attribute defaults don't correctly support decimal input, only hex
- Case 14771: The Gateway Plugin has been document in the AFV2 release notes
- Case 14836: core-cli.c won't compile when CLI is disabled
- Case 14861: End device sometimes unable to rejoin successfully after multiple NWK key changes. Resolution: End Device Support Plugin changed to only poll during Joined state; once device enters Joined No Parent state, the Move/Rejoin state machine is initiated (emberAfStartMove()).
- Case 14767: CLI commands associated with some plugins are not being included in the application by default.
- Case 14869: emberAfFillCommandOtaBootloadClusterImageBlockResponse() macro changed to prevent malformed Image Block Response: imageData is now an int8u array rather than an octet string. Due to this change, callers of this macro must specify the length of the response data twice: once for the dataSize field for the payload and once as imageDataLen for the API so it knows how much to copy.
- Case 14831: SE 1.1.1 Test Case 15.42: Certificate subject EUI64 in CBKE should be verified against partner's EUI. (Fixed in 4.7.0.)
- Case 14815: Syntax for "zcl ota server notify" CLI command is incorrectly documented in Application Framework API Guide's CLI reference.
- Case 14851: Singleton attributes and those attributes with same ID in both client and server cluster are not restored correctly from non-volatile (token) storage.

4.2 Known Issues

- Case 15133: Poll Control Client: timeout value not printed properly
- Case 15131: Poll Control Client: Parameter types are not correct in CLI parser for "plugin poll set-long" and "set-short"
- Case 14887: New Price Server attributes from SE 1.1.1 need to be properly reflected in in ami.xml
- Case 14729: "zcl ota server reload" doesn't properly reload image info when using OTA Simple Storage plugin
- Case 14766: Wrong definition for OTA_HW_VERSION_BIT_MASK in OTA Common plugin
- Case 14723: OTA plugin: Some OTA Upgrade cluster attributes not initialized correctly until client state machine is started
- Case 13220: High-RAM concentrators should send source-routed ACKs with setSourceRoute, sendReply
- Case 13064: OTA - add plugin option to deal with deletion of downloaded images
- Case 12887: ECC Library Path doesn't support whitespace in filenames
- Case 14962: Custom Include File Paths UI widget should not be displayed for pre-4.7 stack configurations
- Case 14992: API command for the permit joining state

- Case 12603: Add EMBER_SERIAL1_RTSCTS define to BOARD_HEADER when flow control enabled in AppBuilder
- Case 11712: DEBUGPRINT: emberAfGuaranteedPrintln behaves differently from emberSerialGuaranteedPrintf
- Case 14117: AppBuilder must permit duplicate cluster IDs across different mfg IDs
- Case 14822: Fragmentation plugin should account for indirect transmission timeout when scheduling emAfFragmentationAbortReception
- Case 12864: Framework doesn't use ezspUtilInit to configure NCP
- Case 14728: Messaging Client plugin should differentiate between Cancel Msg command and timed out / replaced message
- Case 13646: Framework should avoid sending unicast loopback messages with APS security since stack doesn't support this
- Case 14512: Framework should filter outgoing interpan packets unless explicitly allowed by application
- Case 14379: AppBuilder adds extraneous/deprecated APP_BTL define to EM35x SoC projects with App Bootloader
- Case 14311: Some generated structs in af-structs.h are not padding-safe
- Case 14202: AppBuilder's Endian-ness should be consistent with that of the chip
- Case 13983: ZCL devices picklist should filter out devices introduced in later spec versions than the one currently selected
- Case 14056: Single-quote character not escaped properly in endpoint_config file
- Case 14018: AppBuilder should warn if RX queue size for Application Serial Port is very small and CLI is Minimal or Full
- Case 13384: Price Cluster: Consider requesting SummationDelivered and PriceLabel for current tier upon receiving PublishPrice
- Case 11958: DRLC Cancel command can cause new events to be scheduled incorrectly
- Case 11383: When DRLC table is full, device should ignore new DRLC command instead of sending a reject command