UG103.9: ZLL Fundamentals

This document compares the ZLL stack and network with the EmberZNet PRO stack and network, with notes about considerations when implementing a ZLL solution. It includes a basic description of ZLL configuration and commissioning, notes about the interoperability of ZLL and non-ZLL devices.

Silicon Labs' *Fundamentals* series covers topics that project managers, application designers, and developers should understand before beginning to work on an embedded networking solution using Silicon Labs chips, networking stacks such as EmberZNet PRO or Silicon Labs *Bluetooth*®, and associated development tools. The documents can be used as a starting place for anyone needing an introduction to developing wireless networking applications, or who is new to the Silicon Labs development environment.

KEY POINTS

- ZLL network commissioning process
- ZLL and non-ZLL devices
- Differences between ZLL and Zigbee PRO
- ZLL considerations
- AppBuilder configuration

1. Introduction

The Zigbee Light Link (ZLL) profile is a Zigbee application profile based on Zigbee PRO intended for use in over-the-counter, consumer lighting applications. The stated purpose of the ZLL profile according to the specification is as follows:

- To provide an evolutionary consumer experience for lighting devices in which further purchases enhance the overall system.
- To develop a simple and sensible Zigbee specification for over-the-counter lamps and luminaries in the consumer market space.
- To develop a solution, fully in line with consumer market boundary conditions on commissioning, security, ease of use, network scale, cost, etc.
- To be able to address non-installer consumer lighting related features that are not addressed in the Zigbee Home Automation (ZHA) profile.

ZLL has been incorporated into Zigbee 3.0.

2. ZLL Network Commissioning Process

ZLL network commissioning takes place via a process called *touchlinking*. There are two initial phases of touchlinking, *device discovery* and *identifying*, followed by a number of scenarios dependent upon the states of the devices involved. Touchlinking involves two parties, an “initiator”, who is the device (usually a controller of some sort) initiating the discovery process, and a “target”, who is the device (usually a light) being discovered. A device that has not engaged in any touchlinking since leaving the network or forming a new network is known as “factory new” and behaves differently in certain scenarios than a different that is not factory new. These differences, where applicable, are described below.Touchlinking allows for a means of discovering and joining together two devices in close proximity into the same ZLL PAN.

2.1 Device Discovery

Only “address assignment capable” devices, meaning those devices that have the ability to maintain a list of used and available network addresses for allocation by other devices, initiate device discovery. On the initiator side, device discovery begins with a broadcast of eight inter-PAN *scan request* command frames, with the *touchlink initiator* sub-field of the ZLL information field set to 1 and with a nominal output power of 0dBm. The transmission proceeds as follows:

- The initiator switches to the first primary ZLL channel (for example, channel 11) and broadcasts five consecutive *scan request* inter-PAN command frames.
- The initiator then switches to the remaining three primary ZLL channels and broadcasts a single *scan request* command frame on each channel.
- After each transmission, the initiator waits a fixed amount of time to receive any responses.
- If an extended scan is required (such as for a reset to factory new or if the initiator is part of a non-ZLL network), the initiator switches to each of the secondary ZLL channels and broadcasts single *scan request* command frames on each channel, and waits to receive any responses.

If any responses are received with a valid transaction identifier, the initiator may request information about the sub-devices of a target by unicasting a *device information request* inter-PAN command frame; this is, however, unnecessary if a target has only one sub-device. If any responses are received with a valid transaction identifier and the touchlink priority request bit of the ZLL information field of the *scan response* inter-PAN command frame is set to 1, the initiator may consider giving priority processing to those devices.

The device discovery operation may be aborted at any time. If, during its scan, a non-factory new initiator receives another scan request inter-PAN command frame from a factory new target, it is ignored.

On the target side, if the target is an end device, it may first need to be woken so that it can enable its receiver and respond to the scan from the initiator. This is done by application-specific means. If a factory-new target initiates a scan, it defers to a *scan request* inter-PAN command frame from a non-factory-new initiator and responds accordingly. Devices receiving the *scan request* command frame may choose whether or not to respond. If they respond, the device unicasts a *scan response* inter-PAN command frame back to the initiator on the same channel on which the *scan request* command frame was received. If the initiator successfully received the response frame, the target will ignore successive *scan request* frames from the same initiator that have the same transaction identifier.

Prospective targets will only respond to *scan request* frames if its RSSI is above a certain manufacturer-specific threshold and the link initiator sub-field of the ZLL information field is set to 1. The *scan response* frame is then transmitted with a nominal output power of 0 dBm. The target sets the RSSI correction field of the *scan response* frame to a certain product-specific RSSI correction value in order to compensate for RF signals losses between the radio and the outer side of a product. The initiator can then use this value from each discovered target to select an appropriate device.

If the target device wishes to be considered as a priority by the initiator during touchlinking, the touchlink priority request bit of the ZLL information field is set to 1. The target sets the response identifier field to a random (non-sequential) value. If not factory-new, the target sets the extended PAN identifier, network update identifier, logical channel, PAN identifier, and network address fields to the corresponding values determined by the router during its startup phase from NVM. All future inter-PAN communication between the initiator and the target is conducted on the channel indicated in the logical channel field. On receipt of a *device information request* inter-PAN command frame, a target will respond to it by unicasting a corresponding *device information response* inter-PAN command frame.

2.2 Identify

The device discovery operation can result in a list of potential devices from which the application can select one or more devices for further processing. In order to support a user confirmation, the initiator has the possibility to have the selected devices identify themselves.

Once within a device discovery transaction, the initiator generates and transmits an *identify request* inter-PAN command frame to the appropriate discovered target device. The *identify request* frame must contain the same *inter-PAN transaction identifier* that was used in the scan request inter-PAN command frame during device discovery. If the initiator wishes to send further *identify request* frames, it must do so within the active transaction time. Otherwise, device discovery must be repeated.

On receipt of an identify request inter-PAN command frame with a valid transaction identifier the target identifies itself in an application specific way (for example by flashing a lamp) for a time depending on the value of the identify time field. If the value of this field is 0x0000, the device immediately stops its identify operation. If the value of this field is 0xFFFF, the device identifies itself for an application specific time. For all other values, the device identifies itself for a time equal to this value in units of 1 second. If an *identify request* frame is received with an invalid transaction identifier, the frame is ignored. No response is generated to an *identify request* frame. The identify request is non-blocking.

2.3 Form and Join scenarios

As outlined in section 2.2 Identify, at this point the initiator may form a new network and bring an identified target into the network. If the initiator has already formed a network, it may join routers or end devices to its network. The primary difference between *network start request* frames and *network join router / network join end device* frames lays in the absence of Initiator IEEE address and Initiator network address fields in the latter frames. Additionally, the network parameters should already be present and should be included in the join frames. Finally, no network rejoin request is necessary on the initiator side for the latter join scenarios.

3. ZLL and Non-ZLL Devices

In addition to touchlink commissioning, classical Zigbee commissioning is supported on ZLL devices (under application control.) Since the classical Zigbee commissioning mechanisms do not support the concept of a range of addresses, even if the device is address assignment-capable it will only be allocated a single address. Network and group address ranges can only be assigned when the end device is touchlinked with another ZLL device. The application may request primary network discovery, wherein the device will do so over the primary ZLL channel set. If any suitable networks are found, an end device attempts to join the network through the classical Zigbee join mechanism. If the join is successful, the device is authenticated and waits to receive the network key from its parent. Upon receipt of the network key, ZLL address and group attributes are set to 0x0000. If a suitable network is not found or the join is unsuccessful, secondary network discovery is initiated over the secondary ZLL channel set, and the join and authentication procedures proceed as above. If no suitable network is found over the secondary channels, the join is unsuccessful, or authentication fails, the ZLL device selects one of the primary ZLL channels at random and tunes to it. If it is an end device, it enters a low power dormant state and performs no further processing. If it is a router, it enables its receiver and waits for a touchlink command.

An application may request that a router enable its permit joining flag for a fixed duration to allow non-ZLL devices to join if required using the standard Zigbee join procedures. If a device attempts to join using classical Zigbee commissioning, the router allocates an address using classical Zigbee mechanisms. The router will then authenticate the device: if it is on a trust center protected network, the router generates and transmits an APS update device command frame, secured with the trust center link key, to the trust center to allow it to handle the new device. Otherwise, the router generates and transmits an APS transport key command frame, secured with the ZLL certification pre-installed link key to the joining device.

Regarding trust center protected networks, initiators may touchlink to a target device on a trust center protected network whether or not it is on the same trust center protected network; however, touchlinking may not be performed between an initiator on a trust center protected network and a target device that is either factory new or connected to a different network.

3.1 Clusters

The ZLL profile ID is 0xC05E. However, for the purposes of interoperability, ZCL commands addressing clusters other than *ZLL commissioning* or those commands in the commissioning utility command set of the *ZLL commissioning* cluster, use the ZHA profile identifier, 0x0104. The *ZLL commissioning* cluster has a cluster ID of 0x1000.

3.2 Device IDs

The following table contains the device descriptions and identifiers as listed in the ZLL specification, along with their respective ZLL commissioning cluster orientations. ZLL lighting devices do not need to support the commissioning utility functionality of the ZLL commissioning cluster.

Table 3.1. Devices and ZLL Commissioning Cluster Orientations

	Device Description	Device ID	Commissioning Server	Commissioning Server / Client	Commissioning Client
Lighting devices					
	On/off light	0x0000	Yes	Yes	No
	On/off plug-in unit	0x0010	Yes	Yes	No
	Dimmable light	0x0100	Yes	Yes	No
	Dimmable plug-in unit	0x0110	Yes	Yes	No
	Color light	0x0200	Yes	Yes	No
	Extended color light	0x0210	Yes	Yes	No
	Color temperature light	0x0220	Yes	Yes	No
Controller devices					
	Color controller	0x0800	Yes	Yes	Yes
	Color scene controller	0x0810	Yes	Yes	Yes
	Non-color controller	0x0820	Yes	Yes	Yes
	Non-color scene controller	0x0830	Yes	Yes	Yes
	Control bridge	0x0840	Yes	Yes	Yes
	On/off sensor	0x0850	Yes	Yes	Yes

4. Differences Between ZLL and Zigbee PRO

While ZLL is built upon the Zigbee PRO stack, there are differences between the two. The following sections will outline the differences.

4.1 Security Differences

At the Zigbee network layer, ZLL security functions in the same manner as classical Zigbee security, although ZLL security does not require a centralized security controller (known as a “Trust Center” in Zigbee PRO) to authenticate incoming devices. The major difference between the Zigbee PRO and ZLL security schemes lies in security key exchange. During ZLL network commissioning, known as “Touchlinking,” a fixed secret key known as the ZLL key encrypts the exchanged network key. The ZLL key is stored in each ZLL device.

The transfer of the network key is handled differently depending on the type of network commissioning being performed. During classical Zigbee commissioning where a non-ZLL device is being joined to a ZLL network without a trust center, a pre-installed link key is used to secure the transfer of the network key when authenticating. All certified ZLL devices share the secret pre-installed link key, however, prior to certification, a certification key is used for testing purposes. If APS decryption fails through use of the certification pre-installed link key, ZLL devices try to decode the APS message using a known default trust center link key, which is the same default trust center link key used by the Zigbee Home Automation (ZHA) profile and defined in the ZHA specification.

In the case of ZLL touchlink commissioning, sixteen possible algorithms can be utilized to encrypt the network key during the transfer between an initiator and possible target (at present, three algorithms have been allocated.) Within its *scan response* inter-PAN command frame, the possible target indicates in the key bitmask field which key encryption algorithms are supported. When the initiator receives a *scan response* inter-PAN frame from the potential target, it compares its internal key bitmask with the received key bitmask field. If no common key is found through this comparison, the initiator will not select this target for further commissioning. Otherwise, the initiator will set the key index to the bit position corresponding to the matching key with the highest index, encrypt the network key using the appropriate algorithm, and include both the index and the encrypted key in the key index and encrypted network key fields, respectively, of the *network start request, network join router,* or *network join end device* inter-PAN command frames.

Listed below are descriptions of the various ZLL key encryption algorithms:

- **Key index 0 (Development Key)**

  This algorithm encrypts the network key with AES in ECB mode in one step. The associated AES key takes the form “PhLi” || TrID || “CLSN” || RsID, where “PhLi” and “CLSN” are character strings to be converted to their hexadecimal equivalents, TrID is the transaction identifier field of the original *scan request* command frame passed between the initiator and target, and RsID is the response identifier of the *scan response* command frame passed between the target and the initiator (both TriD and RsID are random 32-bit integers). This algorithm is not to be used or supported within Commercial ZLL products.
- **Key Indices 4 and 15 (Master and Certification Keys)**

  The algorithm itself for key encryption using the master and certification keys is the same; the difference comes in the keys used in Network encryption. Key index 4 indicates the usage of the ZLL master key, which is a secret key shared by all certified ZLL devices, whereas key index 15 indicates the usage of a fixed, predetermined key for use during the certification phase known as the ZLL certification key.

  As outlined in the development key algorithm, the transaction and response identifiers are exchanged during the touchlink procedure, and for both encryption and decryption, they are expanded to form 128 bit nonces of the form Transaction identifier || transaction identifier || response identifier || response identifier. For encryption, a transport key is calculated by performing 128-bit AES encryption with the aforementioned nonce as *plaintext*, and the ZLL master or certification key (as the case may be) as *key*. The initiator encrypts the network key using the calculated transport key and AES ECB mode, and then transmits the encrypted network key as part of the touchlinking process. The target receives the encrypted key and, using the calculated transport key, decrypts with AES ECB mode. The target then stores the received network key in the NIB parameter of the ZLL target.

4.2 Network Formation Differences

During the device discovery phase of touchlinking, an initiator will have found an appropriate target. To start a new network, the initiator generates a *network start request* inter-PAN command frame as follows:

- The initiator may, if it desires, specify the PAN ID, extended PAN ID, and logical channel for the new network within the command frame. Otherwise, they are set to zero and are determined by the target.
- The initiator sets the key index and encrypted network key fields of the command frame accordingly to describe the Zigbee network key to be used for securing the network.
- The initiator sets the network address field of the command frame to the selected network address with which the target shall operate on the network. If the beginning of the free network address range is equal to 0x0000, the initiator stochastically generates an address according to the classical Zigbee mechanism. If not, the initiator gives the target the stored beginning address and increments the value. The network address is not changed by the target unless it leaves the network and joins another, or if it is required to do so to resolve an address conflict.
- If during the device discovery phase the target requested a set of group identifiers and the beginning of the free group ID range is not equal to 0x0000, the initiator allocates a range of group identifiers for the target and set the group identifiers begin and group identifiers end fields of the command frame accordingly. If instead it is equal to 0x0000, the aforementioned fields in the command frame are set to 0x0000.
- If during the device discovery phase the target indicated that it was address assignment capable and the beginning network address is not 0x0000, the initiator allocates a range of network addresses and group identifiers that the target can use for its own purposes and set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the command frame accordingly. If instead the beginning network address is 0x0000, the aforementioned fields in the command frame are set to 0x0000.
- The initiator sets the initiator IEEE address and initiator network address fields of the command frame to its IEEE address and the network address it will use on the new network, respectively.

Once the *network start request* inter-PAN command frame has been generated, the initiator unicasts it to the selected target. It then enables its receiver and waits for a pre-specified amount of time or until a *network start response* inter-PAN command frame is received with a valid transaction identifier. If the wait exceeds this duration or the response frame is received with a non-zero status parameter value, the initiator terminates the operation with the target in question. If there are no further targets, the operation at large is terminated and no further processing is performed. Upon receipt of a *network start response* frame with a valid transaction identifier from the desired target, the initiator first copies the network parameters to its network information base if the network parameters were to be determined by the target, then waits to allow the target to start the network correctly. The initiator then issues a network rejoin request to the NWK layer. If the rejoin is successful, the initiator broadcasts a device announcement.

On the target side, when the *network start request* command frame is received with a valid transaction identifier, the target decides via application specific means whether or not to join the network. If it decides not to join the network, it generates a *network start response* command frame with a status indicating failure. It then performs no additional processing. If the target decides to join the network, it checks the PAN ID, extended PAN ID, and logical channel fields. For each field, if the value is equal to zero, the target determines an appropriate value. In order to verify the uniqueness of the PAN and extended PAN IDs, the target issues a network discovery request to the NWK layer over the primary ZLL channels and waits for a confirmation. The target then sets the trust center address to 0xFFFFFFFFFFFFFFFF. The target then generates a *network start response* command frame and unicasts it to the initiator. If the target is not factory new, it leaves its old network, resets its network parameters to the default values, then copies the new network parameters to its network information base and start operating on the new network by issuing a start router request to the NWK layer. After the router has successfully started, it broadcasts a device announcement. In order to allow direct communication via the Zigbee network between the initiator and the target, the target finally performs a Zigbee direct join procedure in order to create an entry in the neighbor table with the IEEE address and the network address of the initiator.

4.3 Addressing

Devices that are address assignment capable assign network addresses. All network addresses must be unique. The method ZLL uses to ensure this is to assign subdivisions of the available address space to devices that join the network and that are address assignment capable. Since Zigbee reserves the network address 0x0000 for the coordinator and the address range (0xFFF8…0xFFFF) for broadcast, the total ZLL network address space is defined in the range (N_{min}=0x0001…N_{max}=0xfff7). Address assignment capable ZLL devices keep track of their current free address range; when such devices are factory new, the address range is (0x0001…0xFFF7). When a factory-new initiator device, which is address assignment capable, has just formed a new network, it assigns itself the network address N_{min} (i.e., 0x0001) and then increments N_{min}. When a device is joined to an existing network, it is assigned the first (that is N_{min}) network address from the free network address range of the initiator through which it is joining. The initiator that started the network then increments N_{min}. If a device cannot be assigned a network address, it is not permitted to operate on the network. If a device that is address assignment capable joins the network, it also receives its own free network address range (N'_{min}… N'_{max}). The initiator splits its own free network address range at an implementation specified point and the upper range (that is, highest in value) is assigned to the new address assignment capable device. If after splitting the free network address range the resulting two address ranges are smaller than an implementation specific threshold, the new device is not joined to the network.

ZLL also supports group identifiers in a similar fashion to its treatment of network addressing. Group identifiers are used when addressing a subset of devices using broadcast mechanisms and they are typically used by a controller application residing at a certain endpoint. The group identifiers need to be unique in the network and their range is (0x0001…0xFEFF). Group identifier 0x0000 is reserved for the default group in the ZCL **scene** cluster, and group identifiers in the range (0xFF00…0xFFFF) are reserved. The number of group identifiers needed by an application residing on an endpoint is given in the device information table. Since group identifier assignment is linked to network address assignment, the total number of group identifiers needed by all endpoints on a node is reported in the *scan response* command frame. Group management is handled as above, where $G_{\neg min} \sim N_{\neg\neg\neg min}$, $G_{\neg max} \sim N_{max}$, $G'_{\neg min} \sim N'_{min}$, and $G'_{max} \sim N'_{max}$.

4.4 Other

ZLL devices are able to operate on all channels available at 2.4 GHz, numbered from 11-26. The **primary** ZLL channels are defined to be 11, 15, 20, and 25. All other channels constitute the **secondary** ZLL channels. Additionally, ZLL supports a channel change mechanism in an application-defined way. When the channel change mechanism is instigated, the device broadcasts a network update with the scan channels field set to indicate the ZLL channel on which to begin operating. Routers receiving the update request update their NIB and execute their channel change procedure. Routers that miss the request can be brought back into the network through a touchlink procedure. If a touchlink initiator wants to bring a router back into the network, it sends a unicast inter-PAN *network update request* command frame. If the touch-link initiator is an end device, it executes a re-join procedure.

5. ZLL Considerations

- Given ZLL address allocation mechanisms, the address spaces of ZLL devices may be exhausted quickly if not managed properly.
- Touchlinking is limited by device proximity (product dependent.)
- ZHA and ZLL commands have different commands and can potentially have device ID conflicts.
- ZHA devices cannot join to ZLL networks because they do not have the requisite ZLL keys.
- For interoperability purposes, ZLL devices must maintain both ZHA and ZLL keys.

6. Application Configuration

Zigbee EmberZNet SDK 7.0 introduced a new component-based architecture, along with a Project Configurator and other tools to replace AppBuilder and plugin configuration. In general, the new software components are comparable to the plugins. For more information, see *AN1301: Transitioning from Zigbee EmberZNet SDK 6.x to SDK 7.x*.

To configure a ZLL application in EmberZNet 7.0 and above, install the **ZLL Commissioning Common** component as well as the **ZLL Commissioning Client** and/or the **ZLL Commissioning Server** component, depending on your application needs. Note that this can also be done by enabling the ZLL Commissioning cluster from the Zigbee Cluster Configurator, which will automatically install the necessary component(s).

To configure a ZLL application in AppBuilder (EmberZNet 6.x or lower), you need to only choose the platform and enable the ZLL Commissioning plugin.

ZLL can be used in either SOC or NCP / host context. Additionally, a ZLL application can determine at runtime whether to operate as a ZLL or a ZHA device, depending on which type of network it occupies.

6.1 Sample ZLL Light / Remote Scenario

The following figures present a sample ZLL Light / Remote Scenario. Node 3 is a ZLL Color Scene Remote and nodes 1 and 2 are ZLL Color Lights. The figures illustrate the touchlinking process and some simple network interactions among the lights and the remote. Repetitive elements (for example, multiple touchlinks) are omitted.

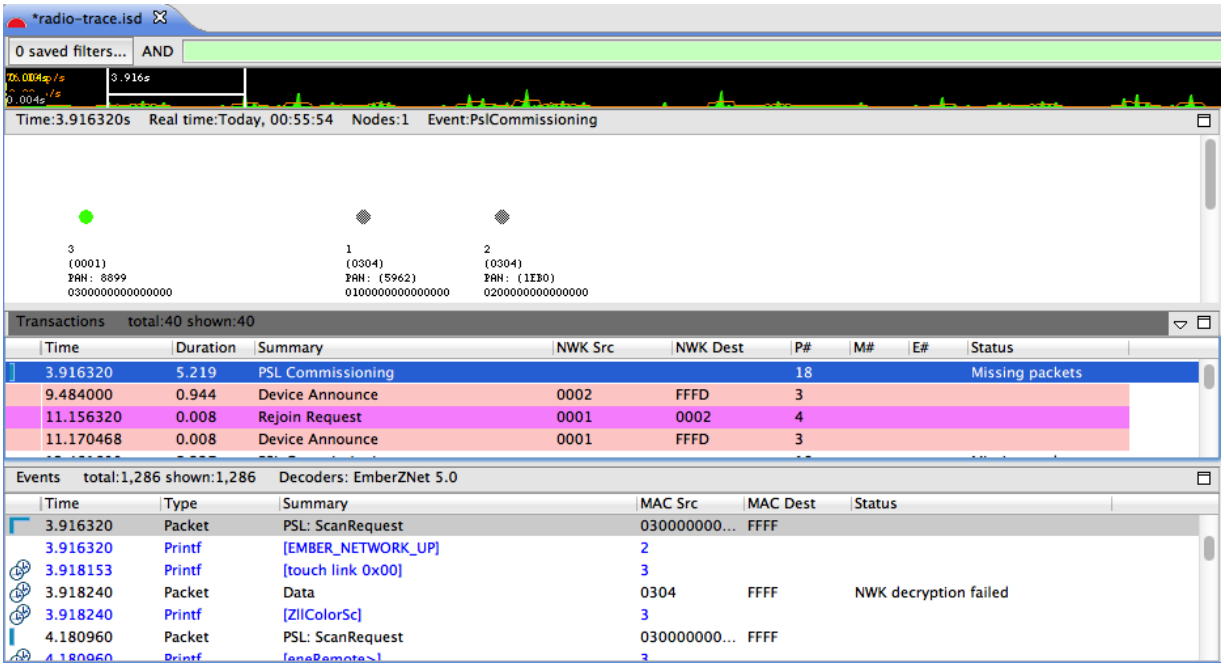

Figure 6.1. Node 3 Initiates a Touchlink and Discovers Node 2

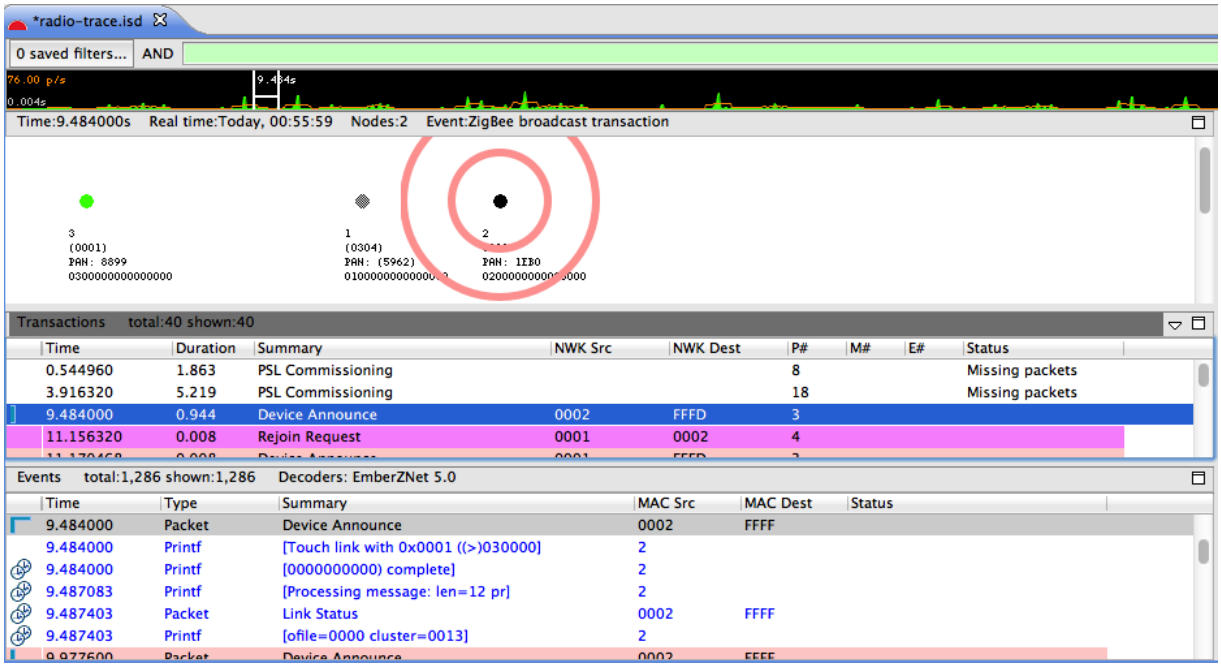

Figure 6.2. Node 2 Announces Itself

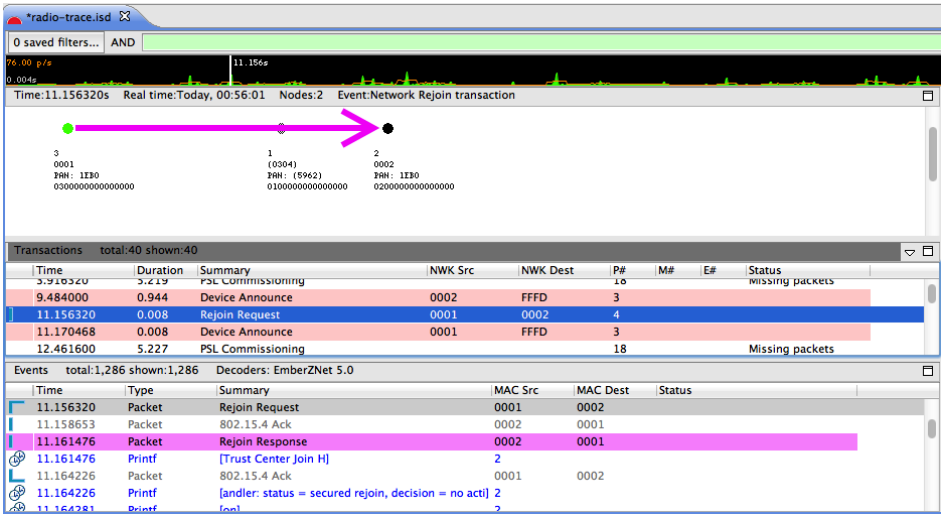

Figure 6.3. Node 3 Issues a Rejoin Request

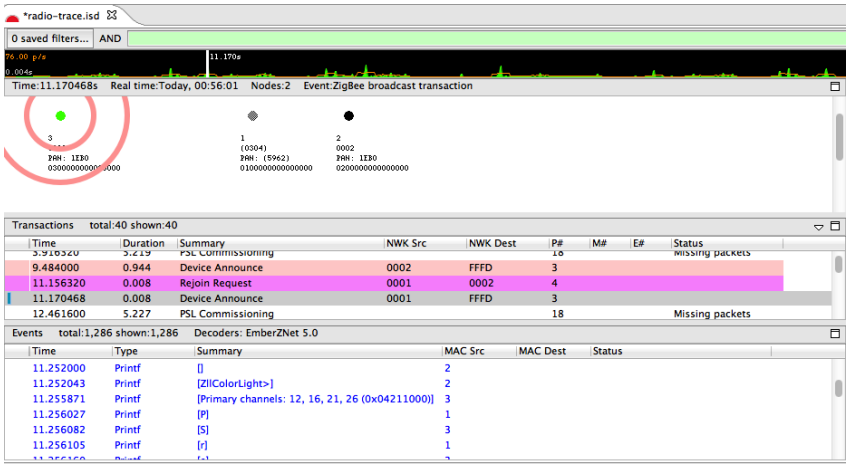

Figure 6.4. Node 3 Announces Itself

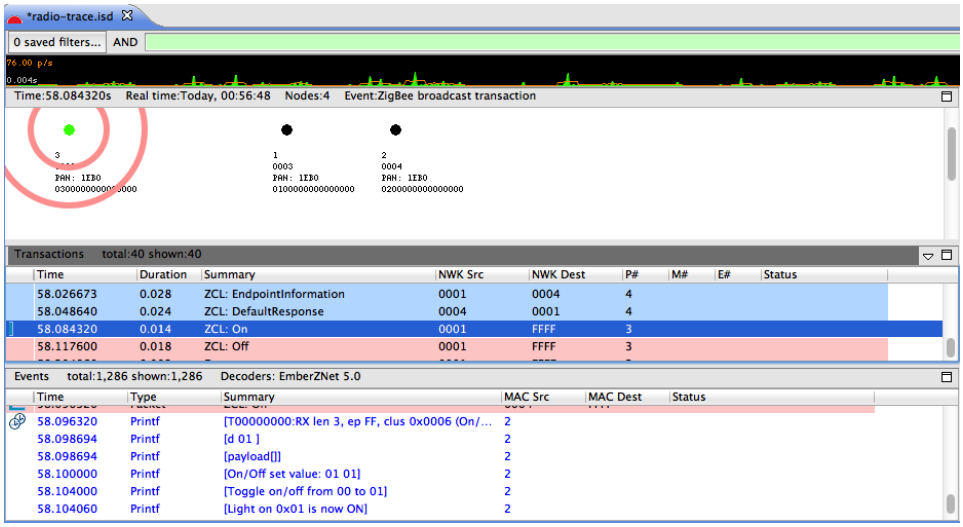

Figure 6.5. Node 3 Broadcasts a ZCL On Command

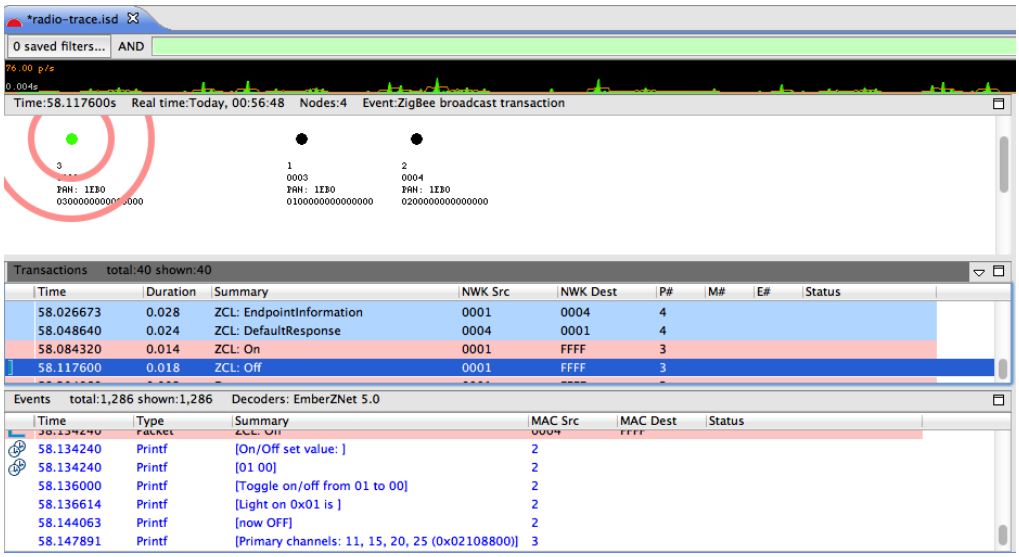

Figure 6.6. Node 3 Broadcasts a ZCL Off Command

Smart. Connected. Energy-Friendly.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com