Instruction

Z-Wave Node Type Overview and Network Installation Guide

Document No.:	INS10244
Version:	9
Description:	This document describes how to setup a network with regard to inclusion of nodes in the network, operation and exclusion of nodes from the network.
Written By:	ABR;JFR;MVO;PSH;BBR
Date:	2018-03-02
Reviewed By:	JFR;JBU;JSI;BBR
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2018-03-02	15:53:23	JFR	Jørgen Franck	on behalf of NTJ

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20050118	SML JFR	ALL	Revised document. Added SUC and SIS. Primary Controller does not use Node ID 0xEF anymore.
2	20060105	MVO	All	New 1st page/header/footer contents. New Doc No
3	20060426	JFR	All	Fixed page number problem
4	20061212	ABR	All	Fixed various small typos Added clarifying comments Added silent ack description to section on routing Added new sections on Zensor Net basics and Zensor Net features
4	20061221	ABR JFR	All	Updated doc after review – minor typos and clarifications
5	20080626	ABR JFR	All	Added features of Dev. Kit 4.50: Random home ID after exclusion/reset Explorer route resolution and Network-Wide Inclusion
5	20080805	ABR	All	Revised sections on Zensor Net and FLiRS nodes.
6	20130514	MVO	All	Replaced first section and updated headers/footers
6	20130603	PSH	All	Updated document to devkit 6.50
7	20140618	JFR	All	Removed confidential
8	20141127	JFR	All	Removed Zensor Net description
9	20180302	BBR	All	Added Silicon Labs template

Table of Contents

Table of Figures

1 ABBREVIATIONS

Abbreviation	**Explanation**
ACK	Acknowledge
FLiRS	Frequently Listening Routing Slave
GUI	Graphical User Interface
RTC	Real Time Clock
SIS	SUC ID Server
SUC	Static Update Controller
WUT	Wake-up timer
SDK	Z-Wave Software Developer’s Kit

2 INTRODUCTION

2.1 Purpose

The purpose of this document is to provide guidelines for installation, maintenance and operation of a Z-Wave network consisting of controller and slave nodes. Refer to [1] regarding how the functionality is implemented in the described devices.

2.2 Backward compatibility

The latest SDK's contain new features to improve installation flexibility and network topology distribution of a Z-Wave network. Therefore is it important to understand these features in detail to ensure backward compatibility with Z-Wave enabled products built on older Developer's Kit releases.

From SDK v3.3x the Static Update Controller (SUC) was introduced to allow slave and controller nodes to request network topology updates.

From SDK v3.4x the SUC can in addition also function as a node ID server (SIS) to allow other controllers to include/exclude nodes to/from the network. Furthermore, the unique node ID 0xEF for primary controllers was discontinued.

From SDK Kit v4.2x the silent acknowledge mechanism was introduced to reduce routing latency. Refer to section 3.4 for details.

From SDK v5.0x Zensor Net and FLiRS technology was introduced. Supports both 200/300 Series SoCs and discontinued 100 Series SoCs.

From SDK v4.5x (v4.50 came after v5.02) explorer frames were introduced to improve network resilience and inclusion flexibility. Refer to section 5 for details. Zensor Net technology was obsoleted.

From SDK v6.0x the 400 series chip is supported as the only hardware platform and 100kbps baud rate is introduced. The SDK 6.0x has all the same functionality as the 4.5x kit and all node types now supports FLiRS. All 6.0x based nodes are compatible with nodes based on older kits.

From SDK v6.5x the 500 series chip is supported as the only hardware platform. The SDK 6.5x has the same features as version 6.02.00 but includes also functionality from 6.11.01, 5.03.00 and 4.55.00. All 6.5x based nodes are compatible with nodes based on older kits.

2.3 Audience and prerequisites

The audience is considered to be OEM's implementing the Z-Wave technology into their products.

3 Z-WAVE BASICS

This chapter describes the basic building blocks of the Z-Wave technology.

3.1 Network Nodes

The Z-Wave network consists of two different types of network nodes; controllers and slaves. The controller nodes are able to calculate routes (and alternative routes). The second node type is the slave node, which generally acts as input and output units. Both types exist in different versions as described below. The Z-Wave protocol supports networks of up to 232 nodes, which can be freely shared between controller and slave nodes.

3.1.1 Controller Nodes

A controller in the Z-Wave terminology is defined as a unit that has the ability to host a routing table of the entire network and calculate routes on the basis thereof. Moreover, the controller has the ability to pass on routes to slave units, in order to enable them to transmit routed signals.

Z-Wave networks are established around a controller. The controller used to include the first node is by default configured to act as Primary Controller with the capability to include/exclude nodes. The Primary Controller is used to include all subsequent nodes in the network.

Being primary is just a role. Any controller can be primary but only one controller can be primary at a time. The primary controller manages the allocation of node IDs and gathers information about which nodes can reach each other via direct RF links. More Portable Controllers as well as Static Controllers can be added as needed as the network grows and are denominated as secondary controllers. The secondary controllers can get copies of the network information gathered by the primary controller.

A Static Controller can be enabled to become a Static Update Controller (SUC), which adds advanced self-organization functionalities to the network. A SUC can furthermore be enabled to become a SUC Id Server (SIS), which adds more flexibility to the installation process. At the same time, the SUC improves the self-healing properties of the network, as the SUC introduces a redundant representation of the network topology. Thus, a lost or crashed controller may have its topology awareness restored from the information stored in the SUC. The SIS is by default a Primary Controller because it can include/exclude nodes. Furthermore it enables other controllers to include/exclude nodes on behalf of the SIS.

The controller exists in a number of fundamentally different versions, which are described in the following sections.

3.1.1.1 Portable Controller

The Portable Controller has the ability to discover its own position in the network, when it needs to communicate with other nodes. An example of a device using this type could be a remote control unit, e.g. for controlling light or HVAC systems. Because the Portable Controller can be carried around in the network, it is also typically used to include/exclude nodes and maintaining the Z-Wave network. Portable controllers are typically battery powered.

3.1.1.2 Installation Controller

The Installation Controller is essentially a Portable Controller node, which incorporates extra functionality that can be used to implement professional installer tools, which need extended network diagnostics. Like the Portable Controller, An installation Controller is typically also battery powered.

NOTE: The installation controller is not available on the SDK 6.5x kit the Portable Controller should be used instead.

3.1.1.3 Static Controller

The Static Controller is required to remain in a fixed position in the network, meaning that it should not be physically moved when it has been included in the network. Moreover it is required that the Static Controller is always in “listening mode” and it must therefore be mains powered. Other alternatives include mains-powered with battery backup as well as running from a large battery which regularly recharged or replaced.

The “always listening” advantage of the Static Controller allows other nodes to transmit frames to it whenever needed, both for uploading purposes as well as for consulting purposes.

The Static Controller also exists in two variants used for more advanced installations. Both variants are described more thoroughly in the installation example chapters (4, 5):

Static Update Controller (SUC):

When a Static Controller is configured as a SUC, the primary controller automatically sends network updates to the SUC, e.g. when a new node is included to the network. The node will therefore automatically appear in the SUC’s topology map. Other controllers in the network may individually request the SUC for network updates. If no SUC is present the Primary Controller is responsible of updating all controllers in the network, which will typically be a manual process for the end-user. The SUC is capable of creating a new Primary Controller in case the original Primary Controller is lost or malfunctioning. There can only be one SUC in each individual network.

SUC ID Server (SIS):

When a SUC is also configured as a node ID server (SIS) it enables all other controllers to include/exclude nodes. The SIS automatically becomes the Primary Controller in the network when enabled. There can only be one SIS in each individual network. To avoid inconsistency, all node ID allocations are maintained by the SIS.

3.1.1.4 Bridge Controller

The Bridge Controller is essentially a Static Controller node, which has the additional capability of representing devices from other network types like X10 or TCP/IP as virtual nodes in a Z-Wave network. This enables control of Z-Wave nodes from e.g. an X10 controller or vice versa.

3.1.2 Slave Nodes

Slave nodes are devices that do not contain routing tables.

Any slave node can act as repeater for frames going to other nodes. The only requirement for being able to act as repeater is that the node is in listening state. This requires that the node is permanently powered, and in order to limit battery consumption, this means that only mains-powered nodes will act as repeaters in most practical installations.

Battery operated slaves that do not listen continually are disregarded by controllers when calculating routes.

There are several types of slave nodes.

3.1.2.1 Slave

The Slave node type is able to receive frames and reply if necessary. The slave node cannot host pre-configured routes to other nodes. The slave node is typically used for devices that only require input (and report status if polled) and do not generate frames unsolicited. An example of a device using this type could be a power outlet.

NOTE: This node type is not available in SDK 6.5x

3.1.2.2 Routing Slave

The Routing Slave can host a number of routes for reaching other nodes. Such routes are called “Return Routes”[1]. The Routing Slave can either be A/C powered or battery powered. If the Routing Slave is mains powered it is used as a repeater in the Z-Wave network. The Routing Slave functionality is used for devices that need to report unsolicited status or alarms.

An example of a routing slave node could be a thermostat or a Passive Infra Red (PIR) movement sensor. A wall switch might also be a routing slave and could then be used to control small lighting scenes, or to establish a kind of “virtual” 3-way switching.

3.1.2.3 Frequently Listening Routing Slave (FLiRS)

A special case of a battery powered routing slave is the Frequently Listening Routing Slave (FLiRS). This is a routing slave configured to periodically listen for a wakeup beam in every wake-up interval. This enables other nodes to wake up the FLiRS node and send a message to it.

One example of a FLiRS node is as chime node in a wireless doorbell system.

3.1.2.4 Enhanced Slave

The Enhanced Slave has the same basic functionality as a routing slave node, but more software components are available because of more features on the hardware. Enhanced slave nodes have software support for External EEPROM. An example of a device using this type of Slave nodes could be a Thermostat.

NOTE: This node type is not available in SDK 6.5x

3.1.2.5 Enhanced Slave 232

The Enhanced Slave 232 has the same basic functionality as a enhanced slave node, but this node type supports storing return routes of up to 231 destinations.

[1] ”Return Route” is a controller-centric term that was originally referring to a route going back to the controller. Seen from the routing slave, “Controller Assigned Route” might be a better term. However, “Return Route” is the established term in all Z-Wave documentation.

3.2 Home ID

The Z-Wave protocol uses a unique identifier called the Home ID to separate networks from each other. A 32 bits unique identifier may be pre-programmed in all controller nodes at manufacturing. This unique 32 bits identifier is automatically used as Home ID during the installation of the Z-Wave network.. Additional nodes will be assigned that same Home ID when included into the network. This is an automatic process that leaves the initial controller as the Primary Controller and the other controller(s) as Secondary Controller(s). In case a SIS is present in the network will it be allocated the Primary Controller role leaving the remaining controllers as Inclusion Controllers.

All slave nodes in the network will initially have a Home ID that is set to zero. In order to communicate in the network the Primary Controller needs to assign its Home ID. It is only a Primary Controller or an Inclusion Controller that can assign Home ID's to a node.

With the introduction of SDK 4.50, the home ID may still be programmed. However, when excluding or resetting a controller it generates a new random home ID.
The generation of a random home ID not only simplifies production, but also eliminates the risk of creating node ID duplicates because a new home ID is used when creating the network.
With SDK's before v4.50. This would happen when resetting a Primary Controller without first removing all nodes from the network, then the home ID would be reused by the controller and that would lead to nodes with dublicated home ID/node ID.

With the introduction of SDK 4.50, slave nodes also generate a new random home ID after each reset. During Network-Wide Inclusion, the home ID is used to identify the node. Once included, the node takes on the home ID of the primary controller and the slave's random home ID is never used again.

3.3 Node ID

Node ID's are used to address individual nodes in a network. A Node ID is an 8 bit value and it is assigned to slave and controller nodes by a Primary Controller or an Inclusion Controller. The Node ID assigned to a device is only unique within a network defined by the unique Home ID. The application must not assume a default Node ID allocation when a network is created because the Node ID allocation algorithm depends on a number of factors.

See Figure 1 for a graphical representation of the Home ID and Node ID of four different nodes before and after installation. The original Home ID (0x00002222) is preserved when the Portable Controller gets the secondary controller role but not used as long it is a part of the network with Home ID 0x00001111.

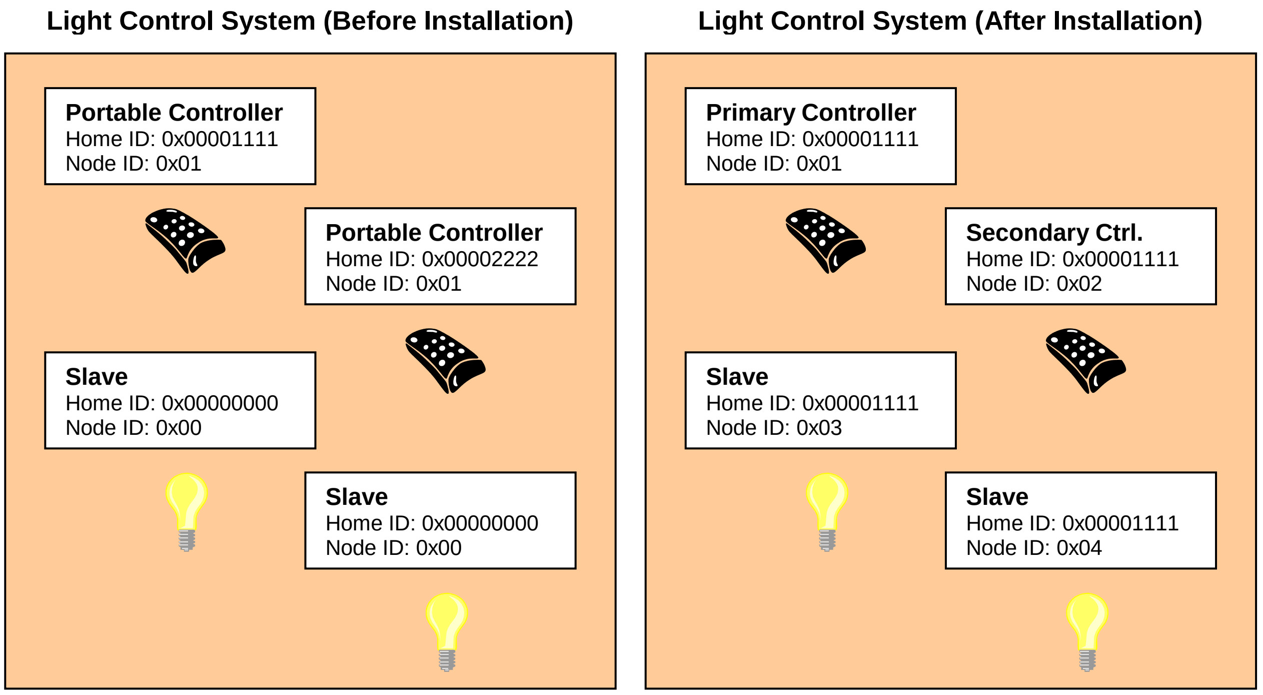


Figure 1. Assigning Home ID and Node ID

See Figure 2 for an illustration of how two adjoining networks will have different Home ID's because different controllers, with each their own unique Home ID's, were used to do the installation.

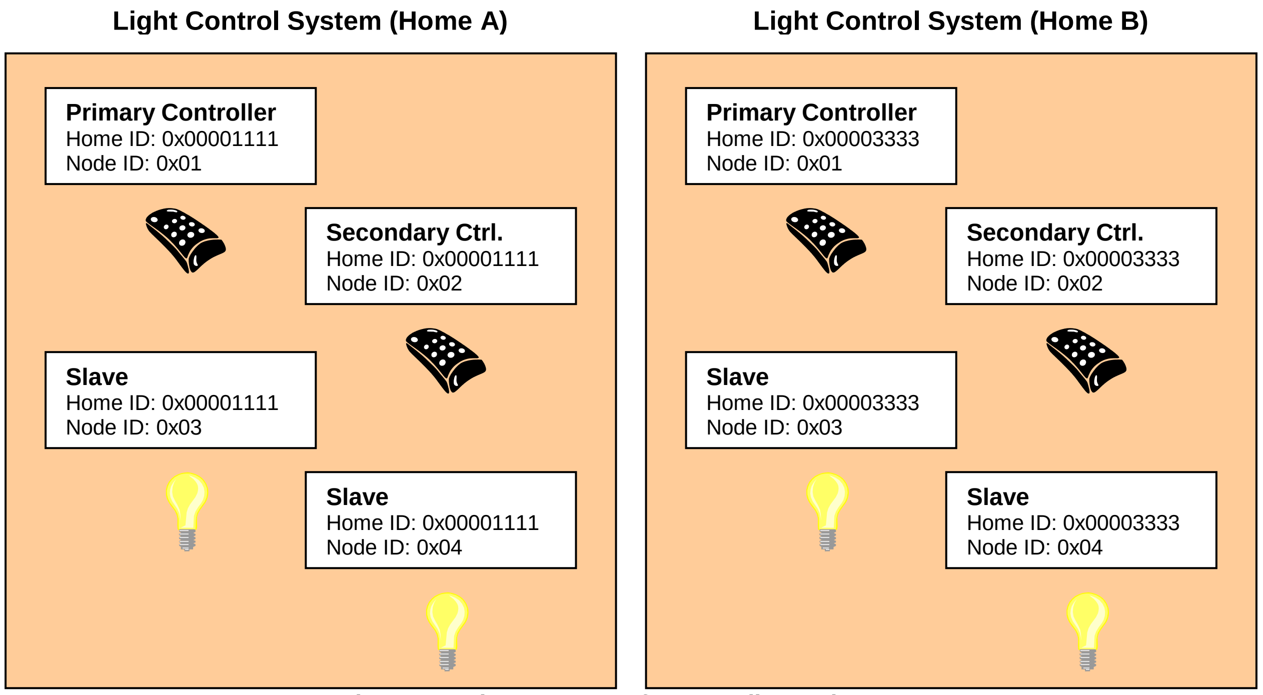


Figure 2. Unique Home ID for two adjacent homes

While the inclusion process is different when performing network-wide inclusion via explorer frames, the end result is the same. All nodes assume the home ID of the primary controller and individual, unique node IDs.

3.4 Routing

All controllers maintains a routing table, which enables the controller to calculate routes in the Z-Wave network. Slave nodes can not initiate transmission of routed frames, unless the controller has provided one or more return routes to the Routing Slave or Enhanced Routing Slave.

The forwarding mechanism for routed frames changed significantly from version 4.2x. Before that version, every frame transmission was acknowledged before the frame was forwarded to the next node in the path.
Version 4.2x makes use of the fact that the frame forwarded to the next node in the path is just as easy to hear from the originator as an acknowledgement frame would be. This trick is nicknamed “Silent Ack”. By using silent ack, airtime is freed up and protocol latency is reduced.

Before version 4.2x: Full acknowledgement

The figure below shows an example of the frame flow when a frame is sent from a controller, repeated by slave 1 to slave 2.

1. The controller sends a frame for slave 2 routed via slave 1
2. Slave 1 confirms the reception of the frame by sending out an ack.
3. Slave 1 repeats the frame.
4. Slave 2 confirms the reception of the frame by sending out an ack.
5. Slave 2 sends out a routed ack for end-to-end confirmation
6. Slave 1 confirms the reception of the frame by sending out an ack.
7. Slave 1 repeats the routed ack.
8. The controller confirms the reception of the frame by sending out an ack.

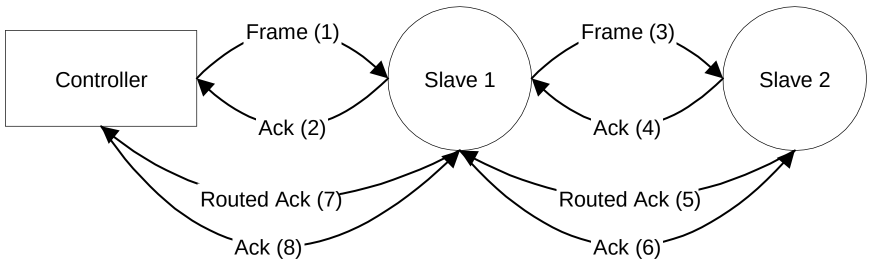


Figure 3. Routed frame flow

From version 4.2x: Silent ack

The figure below shows an example of the frame flow when a frame is sent from a controller, repeated by slave 1 to slave 2.

1. The controller sends a frame for slave 2 routed via slave 1
2. Slave 1 repeats the frame.
   The controller receives the frame and interprets this as an acknowledgement
3. Slave 2 sends out a routed ack for end-to-end confirmation
   Slave 1 receives the routed ack and interprets this as an acknowledgement
4. Slave 1 repeats the routed ack.
   Slave 2 receives the routed ack and interprets this as an acknowledgement

The controller confirms the reception of the frame by sending out an ack frame

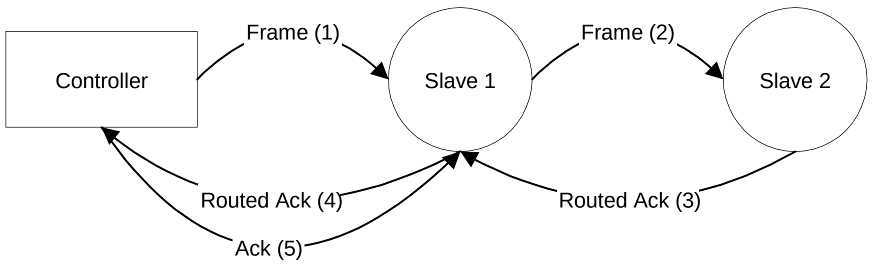


Figure 4. Routed frame flow w. silent ack

3.5 Initiators

The Z-Wave system is relying on the ability to include, exclude and operate nodes in the network, as described below. This requires that each node has a number of “Initiators”. In the rest of this document an initiator does not necessarily mean a physical button. An Initiator can either be a physical button, a special activation of a button (e.g. activating a button for 2 seconds), a combination of buttons (e.g. activating two buttons simultaneously) or an item in a menu system.

3.5.1 Slave Initiators

The slave nodes only need one Initiator since all the intelligence is placed in the controller nodes. The slave initiator will be used for inclusion/exclusion, association/disassociation and operation depending on whether the initiators on the controller are activated or not. If additional features need to be supported, additional initiators may be required.

3.5.2 Controller Initiators

The controller nodes need at least the following initiators:

Include Initiator:

The include initiator is used when Primary and Inclusion Controllers include nodes in the network. When both the include initiator on the Primary/Inclusion Controller and a slave initiator have been activated simultaneously, the new node will be included to the network if the node was not included previously.

Exclude Initiator:

The exclude initiator is used by Primary Controllers to exclude nodes from the network. When the exclude initiator and a slave initiator are activated simultaneously, it will result in the slave being excluded from the network (and reset to Home ID zero). Even if the slave was not part of the network it will still be reset by this action.

Associate Initiator:

The associate initiator is used to associate nodes when both initiators on two nodes have been activated, the two nodes will be associated. This has the effect that when activating the operate initiator, all devices associated with this node is being operated.

Operate Initiator:

The operate initiator is used for controlling the functionality of the associated devices (e.g. a portable controller turning light on/off).

Return Route Initiator:

The controller assigned route initiator is only needed in systems where the Primary Controller is required to generate return routes for Routing Slaves. The initiator is used to associate a Routing Slave with Static Controllers and other slaves.

Static Route Initiator:

The static route initiator is used when creating associations between slaves and static controllers out of direct reach. Using a portable controller, a two-step process is used to create an association. Refer to Figure 15, page 23, for details.

If additional features need to be supported, additional initiators may be required.

3.6 Node Information Frame

When the Include Initiator in a node is activated the node will issue a node information frame. This frame is part of the Z-Wave protocol, and advertises the capabilities of the node. These capabilities announced include the node type, whether the node is able to repeat frames, and other protocol relevant issues. The node information frame also contains the Home ID and the Node ID.

It is possible for the application to ask for the Node Information Frame from all nodes in the network and hence enabling any node to acquire information regarding any other nodes features in the network at any given time.

4 Z-WAVE NETWORK FEATURES

This chapter contains information about functionalities in a Z-Wave Network.

4.1 Include/exclude Process

All nodes are required to take part in the include/exclude process. Only the Primary Controller and Inclusion Controllers in the network are able to include and exclude nodes. This strategy is chosen to ensure that the network stays consistent with respect to the Node ID's allocated.

From the Installer viewpoint all Z-Wave nodes (regardless of their node type) will use the same installation process.

4.1.1 Including Nodes

When a node is to be included in the network, the Primary Controller or Inclusion Controller will grant the request if there is Node ID's available. The network can consist of maximum 232 nodes, including the primary controller.

4.1.1.1 Including slaves

The Include process is initiated by activating both the include initiator on the controller and the slave initiator. This causes the slave node to send out its node information frame. When the node information frame is received from an un-initialized node the controller will assign Home ID and Node ID to the slave. If a slave has already been assigned a Home ID and Node ID in atother network, it will not accept the inclusion. The Node will have to be excluded (or reset) before the node can be included again.

4.1.1.2 Including controllers

The inclusion process is initiated by activating the include initiator on the Primary Controller or Inclusion Controller and the include initiator on the new controller that should be included into the network. The Primary Controller or Inclusion Controller will assign Home ID and Node ID to the new controller, which will then automatically become a Secondary or Inclusion Controller.

As part of the inclusion of additional controllers into the network, a replication of the routing tables and optionally other information will automatically take place. This ensures that the new controller have the newest information available. At later stages, when the Primary Controller has been updated with new nodes (or nodes have been deleted), a new replication can be initiated by activating the include initiators on both controllers as desired above. This network information updating can also be done automatically using the Static Update Controller functionality.

4.1.2 Exclude Nodes

The exclusion of Slave nodes happens when the exclude initiator on the Primary Controller and the slave initiator are activated. When nodes are excluded it will result in the node being reset and the topology information in the Primary Controller or Inclusion Controller being updated to reflect the change. This information is then passed on to the SUC if one is present.

When a slave node is reset, it will assume Home ID zero again. When a controller is reset, it will assume the Home ID it was originally programmed with during the manufacturing process. This applies for SDK's prior to v4.5x.

For SDK from version 4.5x and onwards, a reset results in a new random Home ID in both slaves and controllers respectively.

4.2 Association of nodes

When two nodes have been included into the network it is possible to associate these with each other. The following describes the different possibilities for associating nodes.

Convention: “A is associated with B” means that A is under the control of B.

4.2.1 Associating a Slave with a Controller

A slave node may be associated with a controller by activating both the associate initiator on the controller to make it go into receive mode and the slave initiator on the slave to make it transmit its node information frame. The controller will receive the node information from the slave node and add the slave node to its internal associated nodes list maintained by the application layer.

Example: The slave is an outlet dimmer module and the controller is a remote control

4.2.2 Associating a (Routing) Slave with a Routing Slave

A (Routing) Slave may be associated with a Routing Slave by using a Controller. The association of the nodes takes place by first activating the return route initiator on the Controller and the initiator on the destination Node (node that should be controlled). Secondly the controller assigned route initiator on the Controller and the initiator on the source node (node that should be controlling) are activated. The Controller will then generate one or more routes and transmit these to the source node. This functionality can typically be supported by a GUI wherefrom the source and destination can be selected in a list and finally push out routes to the source via RF.

Example: The slave is an outlet switch module and the routing slave is a movement sensor.
The controller used for association might be a portable controller.

4.2.3 Associating a Static Controller with a Routing Slave

A Static Controller may associated with a Routing Slave by use of a Controller. The association of the nodes takes place by first activating the controller assigned route initiator on the Controller and the associate initiator on the destination Static Controller (node that should be controlled or informed about events). Secondly the controller assigned route initiator on the Controller followed by the initiator on the source node (Routing Slave node that should be controlling or sending event information) are activated. The Controller will then generate one or more routes and send them to the source node.

Example: The static controller is a wireless wall switch and the routing slave is a wall mounted.
scene controller. The controller used for association might be a portable controller.

Note: The static controller could set up this association by itself if it is somehow possible to communicate to it who the source node is (it is assumed that they are not within direct range in which case the situation would be similar to the “Associate a Slave with a Controller” example above). By using a controller for this job, the same procedure can be followed whether a routing slave is to control a slave, a routing slave or a static controller. Even more important, this procedure makes it possible for an end-user to set up his wall mounted scene controller without having to bother with whether this product is based on the controller or the routing slave libraries.

5 EXPLORER FEATURES

With the introduction of SDK 4.50, a new class of services was added to the Z-Wave protocol layer. All these services make use of Explorer frames. Explorer frames are broadcasted Z-Wave frames, which carries a special explorer header and optionally an embedded Z-Wave command to be executed by a target addressed inside the explorer header.

Special rules specify how explorer-supporting nodes should forward copies of explorer frames.

5.1 On-demand Route Resolution

If failing classic routing, a node may issue an explorer SearchRequest frame.
All nodes supporting explorer frames may forward the request in accordance with the forwarding rules.

Once the explorer frame reaches the target of the search, the target looks for an embedded command to execute and returns an explorer SearchResult to the requester.

The SearchResult frame carries a Stop flag to prevent all neighbor nodes from sending more explorer frame copies.
The flag is copied from the SearchRequest, which carries the flag from the requester to the target.

5.2 Network-Wide Inclusion

Explorer-supporting nodes may issue an explorer InclusionRequest frame. The InclusionRequest may be forwarded by all other Explorer-supporting nodes and the frame is used as a trigger for including out-of-direct range nodes.

5.3 Network migration

Implementers may choose to create products that benefit from new explore features while maintaining full compatibility with existing designs.

5.3.1 Compatibility

Explore-supporting products are backwards compatible with earlier versions. This means that a explore-supporting slave may be included with a classic Z-Wave controller and a classic Z-Wave slave may be included with a explore-supporting controller. In both cases, classic inclusion will be used.
With classic inclusion, a controller asks the included slave to perform a neighbor discovery. A explore-supporting controller also asks the included slave to perform a neighbor discovery. This allows a classic Z-Wave secondary controller included subsequently to calculate routes to explore-supporting slaves.

5.3.2 Route resolution strategy

In order to support classic Z-Wave nodes, a explore-supporting controller always uses the following strategy:

1. Try last working route
2. Try a number of routes calculated from the routing table
3. Issue an explorer search request

Thus, in a predominantly static infrastructure, explorer frames may only be used during network-wide inclusion, as the route table stays valid.

Now, if the nodes are moved around, earlier versions may fail calculating alternative routes.
Explore-supporting nodes will also fail trying the last working route and calculation of alternative routes. If this happens, explore-supporting node may send out an explorer SearchRequest frame.

For an explorer frame to reach its target, a number of repeater nodes must be present. The sufficient number depends on the actual network topology and the amount of RF noise in the environment. This observation applies to route resolution as well as network-wide inclusion.

In short, explore-supporting provides the same level of network robustness as previous versions plus improved discovery features if a sufficient number of explore-supporting nodes are present.

5.3.3 SIS functionality for improved plug and play experience

Network-wide inclusion via explorer InclusionRequest makes use of a central primary controller. The primary controller assigns a unique node ID to each new node included.
explore-supporting nodes may be included from anywhere in the network; also when out of direct reach.

By implementing SIS functionality in a central controller based on explore-supporting SDK, one may get the best of both worlds. Classic Z-Wave Nodes may be included via any inclusion controller in the network.

Explore-supporting nodes may also be included via any inclusion controller in the network. In addition, the user may activate network-wide inclusion in the primary controller, e.g. via a web page. In this mode, explore-supporting nodes may be included from anywhere in the network.

6 BASIC INSTALLATION EXAMPLE

This chapter contains some examples of how to install elements of a Z-Wave system in a sequential order. It also describes how nodes are associated.

6.1 Initial Setup

The most basic setup is one controller node and one slave node, as depicted in Figure 5.

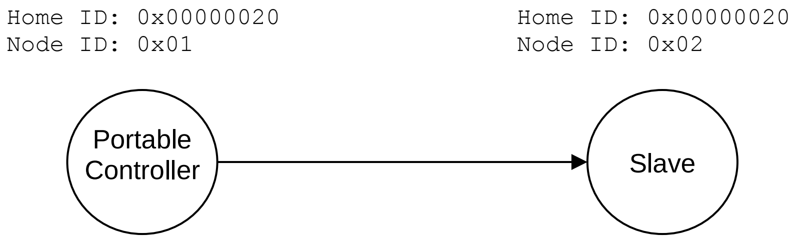


Figure 5. Including a Slave node.

Initially both the controller and the slave node are reset. The controller will have a predefined Home ID = 0x00000020) and a Node ID =0x01. The slave have a predefined Home ID = 0x00000000 and Node ID = 0x00. Being the first controller in the system and therefore determining the Home ID, the controller defaults to become the Primary Controller and takes node ID = 0x01.

When the include initiator on the controller and the slave Initiator is activated simultaneously, the include procedure is initiated. The controller assigns the Home ID = 0x00000020 and the Node ID = 0x02 to the slave node. The slave is now part of the Z-Wave network, but no specific association has been made between the two nodes (the inclusion and the association process can be done as one interlinked action from the end-user's perspective).

6.2 Associating Nodes

The two nodes in Figure 6 are included in the same Z-Wave network, but have not yet been associated.

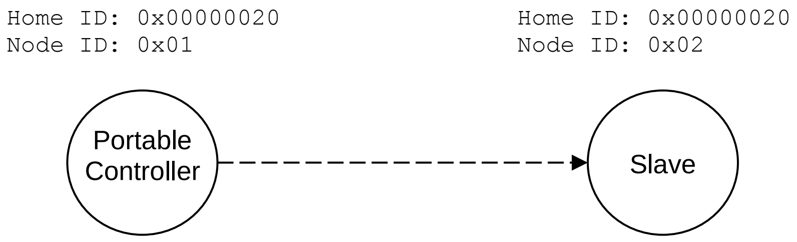


Figure 6. Associating a Slave Node with a Controller.

If the two nodes should be associated, this can happen by activating the associate initiator on the controller and the slave initiator simultaneously. This will cause the controller to listen for a node to associate. When the slave sends a node info frame (specified by the Z-Wave protocol and prompted by activating the slave initiator), the controller will include the slave in the associated slaves list.

Every time the operate initiator on the controller is activated, the controller will send a message to the slave node, which will cause the slave node to perform the specified action.

6.3 Adding Slave Nodes

New slave nodes can be included in the Z-Wave network as the network is gradually expanded. In Figure 7, the example is extended to include another slave node. The inclusion happens the exact same way as for the first slave node. The second slave node will be assigned the Home ID = 0x00000020 and the Node ID = 0x03. The second slave node can also be associated with the controller by activating the associate initiator on the controller and the slave initiator simultaneously. The result is depicted in Figure 7.

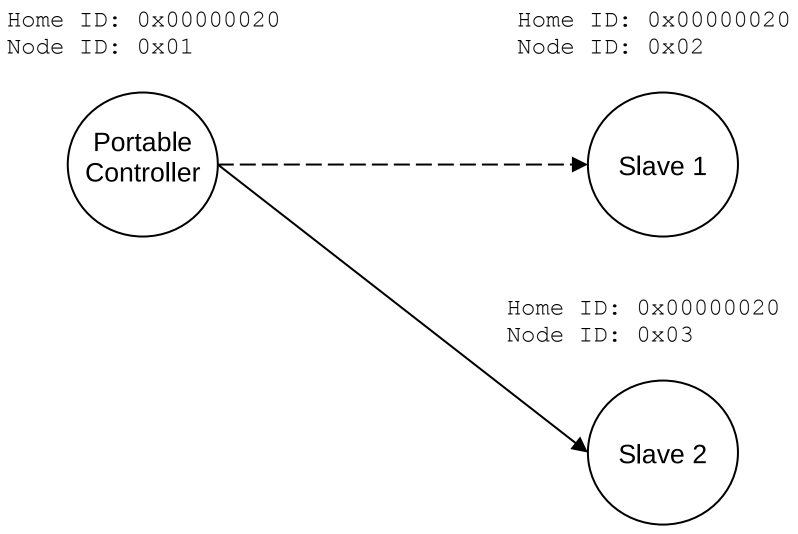


Figure 7. Including Slave node 2.

The second slave node can also be associated with the controller by activating the associate initiator on the controller and the slave initiator simultaneously as shown on Figure 8.

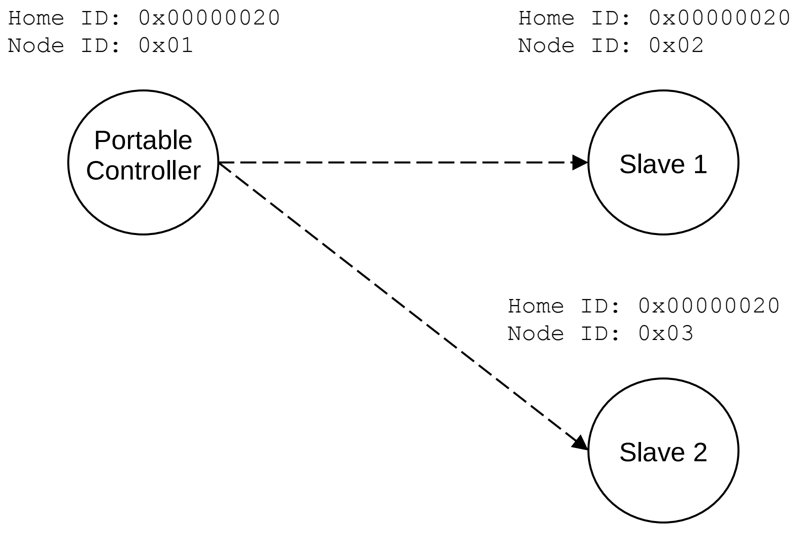


Figure 8. Associating Slave Node 2 with the Controller.

6.4 Adding Controllers

The Z-Wave network can also be extended to include multiple controllers. Figure 9 shows a system with 2 controllers.

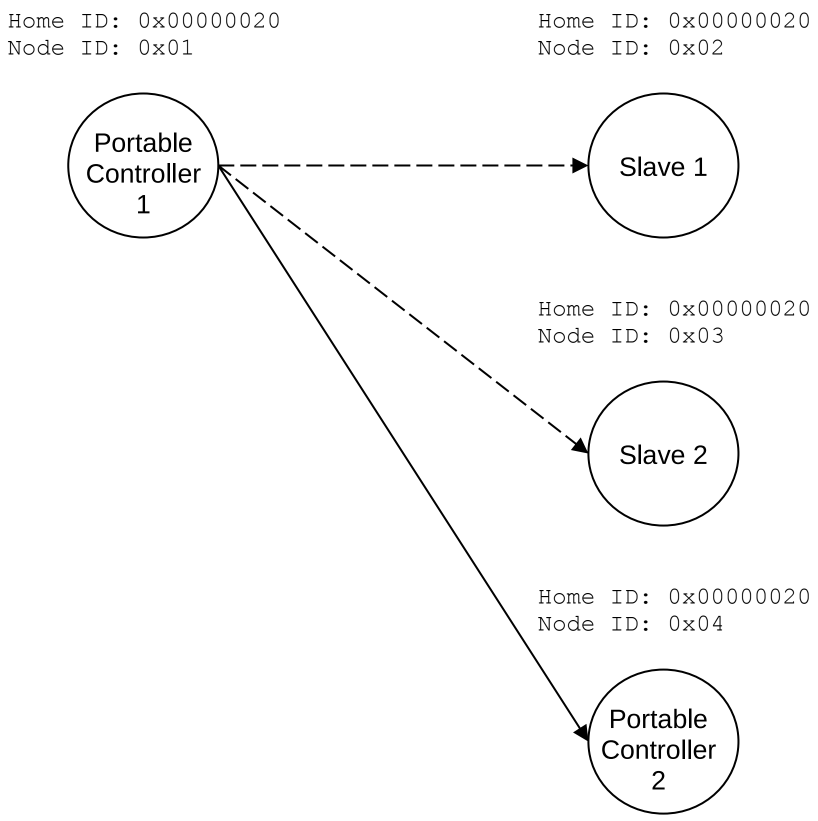


Figure 9. Including a second Controller.

The second controller to be added comes with another unique default Home ID, for example 0x00000030 and Node ID = 0x01. When the Include initiator is activated on both controllers, the include process is initiated. The first controller, which acts as Primary Controller for the Z-Wave network, will assign the Home ID = 0x00000020 and the Node ID = 0x04 to the second controller, which now assumes the position as a Secondary Controller.

Following the exchange of network identifiers a replication will take place. During the replication process, all routing information will be transferred from the Primary Controller to the Secondary Controller. The Secondary Controller is now able to perform all the same functionalities as the Primary Controller except from including/excluding nodes in the Z-Wave network.

Additional slave nodes can be assigned to the Z-Wave network by the Primary Controller and associated with either controller 1, controller 2 or both. In Figure 10 below two slave nodes has been included with the Primary Controller and associated with controller 2.

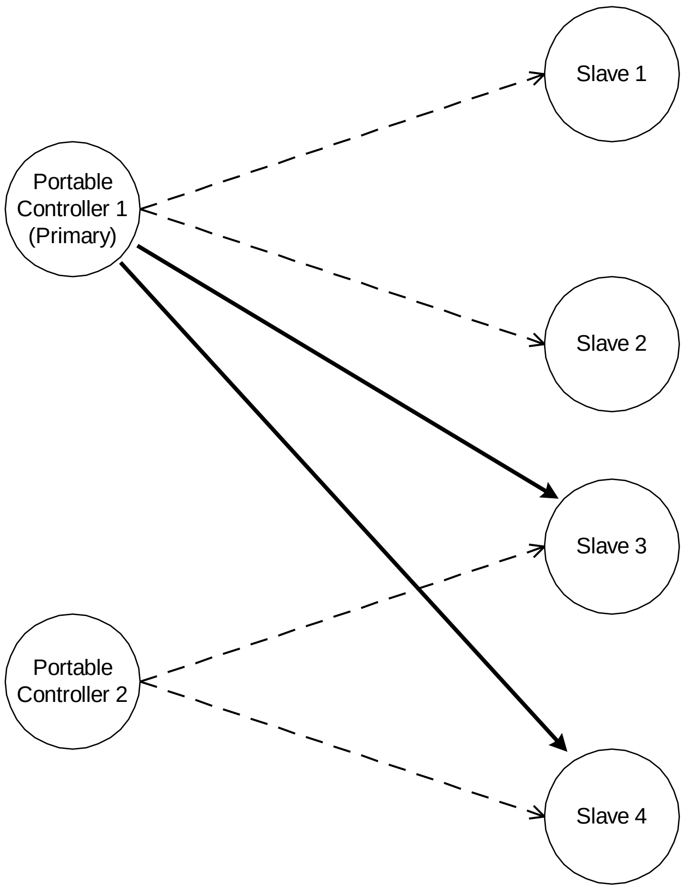


Figure 10. Including and associating additional Slave nodes.

The Portable Controller 2 can be associated with slave 3 and slave 4. The routing table of Portable Controller 2 will not be updated before Portable Controller 1 performs a controller replication.

6.5 Removing Slaves

A slave node can be removed from the Z-Wave network by activating the exclude initiator on the Primary Controller and the slave initiator. This action will cause the controller to send Home ID = 0x00000000 and Node ID = 0x00 to the slave, thereby resetting the slave. The controller will also remove the routing information from the routing table.

6.6 Removing Secondary Controllers

A Secondary Controller can be removed by activating the exclusion initiator on both the Primary Controller and the Secondary Controller.

6.7 Recover from Controller Node Error

If a controller fails, recovery depends on whether it is a Primary Controller or a Secondary Controller. If it is a Secondary Controller it can just be removed, a new can be included and the routing information can be replicated.

If the failed controller is the Primary Controller the situation is different. Basically the entire network can be reset and then reconfigured with a new controller. This controller will then become the new Primary Controller. The unique 32 bits identifier of the new Primary Controller will now be the new Home ID for the entire Z-Wave network.

A more advanced method is to use the Network management features of the Z-Wave Network.

If a Static Controller is configured as a Static Update Controller (SUC), it will keep an updated version of the routing table. All controllers in the system may request updates from the SUC. The primary controller is required to send updates to the SUC when new nodes have been included or old nodes have been reset. If the Primary Controller malfunctions, a secondary controller can be requested to be primary by making replication from the SUC.

A SUC basically implements a mirror service for the primary controller, allowing the user to reconstruct all topology settings and inclusion data in a new primary controller; should the primary controller break down.
Note that associations and other application-level data is not stored in the SUC. The SUC only holds protocol-level data. Thus, the SUC increases the network robustness but it is not a general backup service.

7 STATIC CONTROLLER AND ROUTING SLAVE EXAMPLE

This chapter extends the previous example to also include Static Controllers and Routing Slaves. The starting point for this example is a network with a Portable Controller and two slave nodes.

7.1 Including Routing Slaves

The first step in this scenario is to include a Routing Slave, see Figure 11.

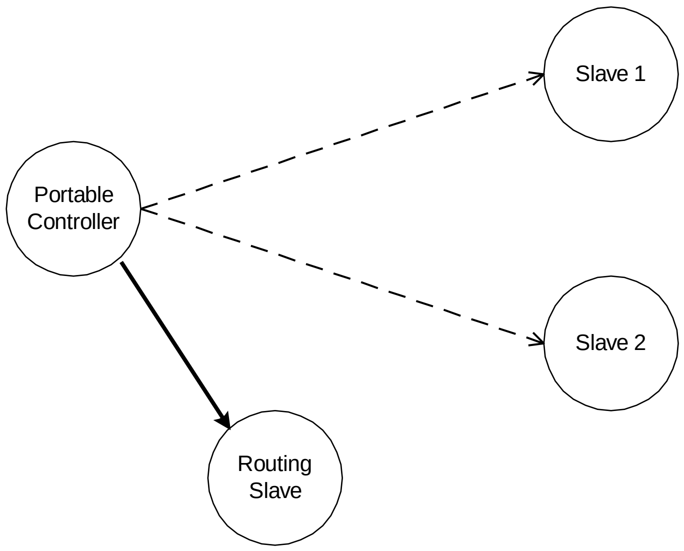


Figure 11. Including a Routing Slave.

The Routing Slave is included in the network in the exact same way as any other slave node. When the include initiator on the controller and the initiator on the Routing Slave are activated, the controller will assign Home ID and Node ID to the Routing Slave.

The Routing Slave can now be associated with a controller or another slave. The controller is an important part of facilitating the association process although it is not part of the final association.

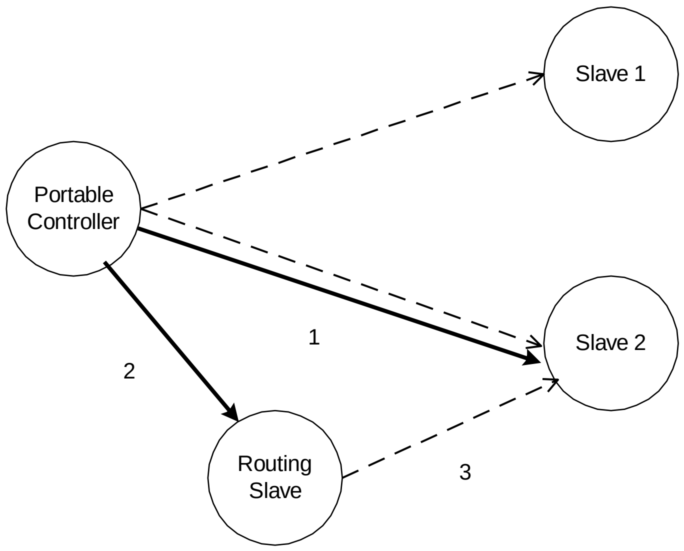


Figure 12. Associating a "Simple" Slave with a Routing Slave.

The controller needs an extra initiator compared to the previous example. This initiator is called a controller assigned route initiator. The controller assigned route initiator on the controller and the initiator on slave 2 (the device to be controlled) are activated. The procedure is repeated for the Routing Slave. Upon doing the second part of the association, the controller will calculate one or more routes between the Routing Slave and Slave 2. These routes will be sent to the Routing Slave, which will use the routes when communicating with Slave 2.

7.2 Including Static Controllers

This part of the example describes a situation where the network initially consists of one controller and three Routing Slaves. This network is depicted in Figure 13.

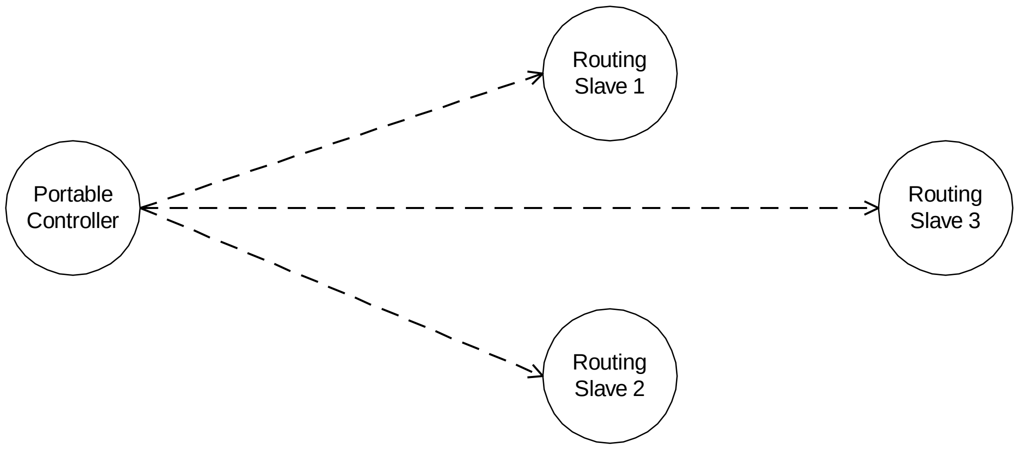


Figure 13. Controller with three associated Routing Slaves.

The following figure shows the previous example extended with a Static Controller. A Static Controller is included in the same way as a Portable Controller. The Static Controller will be assigned a Home ID and a Node ID, and the topology information is transferred. The main difference in the inclusion procedure is that the Static Controller automatically investigates which neighbours it is able to see during the include process. Figure 14 illustrates the network after the Static Controller has been included.

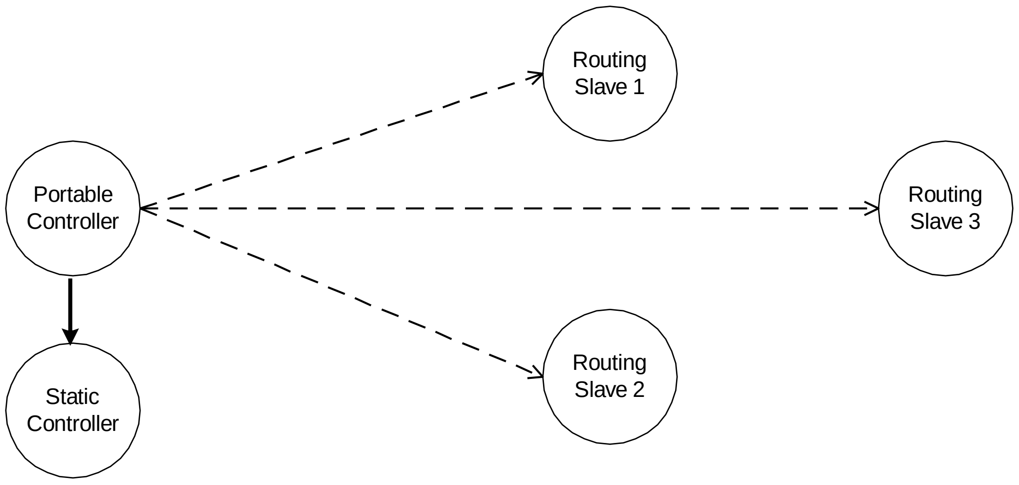


Figure 14. Including a Static Controller.

The Portable Controller now contains the full routing information, including the knowledge of which nodes that can be reached directly from the Static Controller.

In this scenario it is now possible to create an association that allows a Routing Slave to communicate with the Static Controller.

In the following example we want to associate the Static Controller with the Routing Slave 3. In the example it is assumed that the Static Controller can only reach Routing Slave 1 and Routing Slave 2 directly. Also for the Routing Slave 3 it is assumed that it is able to reach Routing Slave 1 and Routing Slave 2 directly, but not the Static Controller.

The Association is done as described in the previous example, by first moving the Portable Controller within range of the Static Controller, and then activating the static route initiator on the Portable Controller and the associate initiator on the Static Controller. Secondly the Portable Controller is moved closer to Routing Slave 3. When the associate initiator on the Portable Controller and the initiator on slave 3 have been activated, the Portable Controller will calculate routes from Routing Slave 3 to the Static Controller. In this example the Portable Controller will come up with two alternative routes. The first route is to reach the Static Controller through Routing Slave 1. The second route is to reach the Static Controller through Routing Slave 2.

Figure 15 shows the Z-Wave network after the Static Controller has been associated with Routing Slave 3.

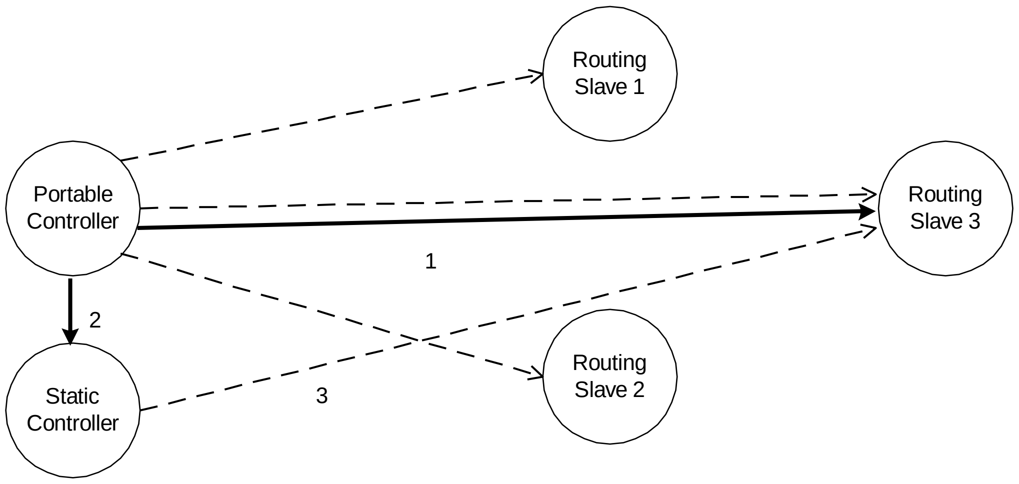


Figure 15. Associating a Static Controller with a Routing Slave.

8 STATIC UPDATE CONTROLLER (SUC) EXAMPLE

When a Static Controller is configured as a Static Update Controller (SUC) then it will automatically receive all changes in the network topology from the Primary Controller. Other nodes in the network can acquire this information.

Note that only one SUC can be present in each individual network.

8.1 Including a SUC to the network

The following step describes the process of including a SUC to the network:

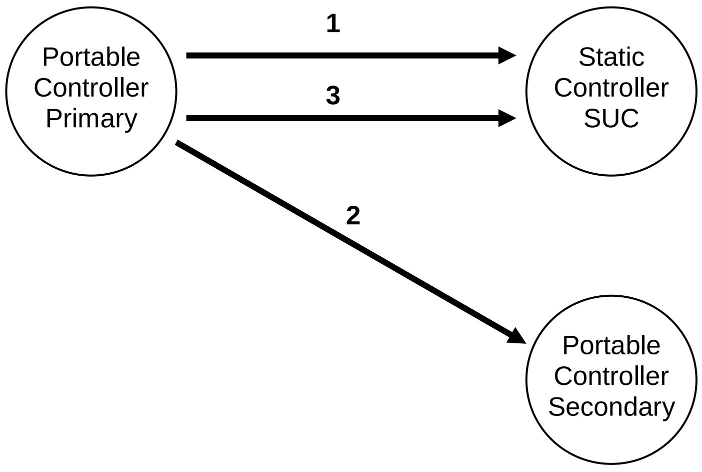


Figure 16. Including a SUC to the network.

1) Include the Static Controller as described in the previous chapter by replicating the information from the Primary Controller to the Static Controller. The Static Controller is now included to the network and is secondary. Afterwards will the application on the Primary Controller assign the Secondary Static Controller to become the SUC.

2) The Primary Controller includes an additional Portable Controller node to the network in same way as described in previous chapters.

3) The Primary Controller will automatically send the updated network topology for the included Secondary Portable Controller to the SUC. From now on every time a new node (controller or slave) is included to the network the SUC will get the updates automatically.

Association is done exactly as described in previous chapters.

It is recommended to include the SUC as the first node in the network because new nodes can then be advised about its presence when included.
Adding a SUC to an existing network will cause battery-operated nodes, such as portable controllers, to be unaware of the SUC. Such nodes will never be able to request topology updates from the SUC.

9 SUC ID SERVER (SIS) EXAMPLE

The SUC may be further configured with node ID server (SIS) functionality to enhance installation flexibility. The SIS enables other controllers to include/exclude nodes to/from the network. When SIS functionality is enabled the controller also takes the role as the Primary Controller because it has both latest network topology and allocated node IDs. All the other controllers are called Inclusion Controllers because they can include/exclude nodes to/from the network. Until the SIS is included to the network the installation process is as described in the previous installation chapters.

Note that only one SIS can be present in each individual network.

9.1 Including a SIS to the network

The following step describes the process of including a SIS to the network.

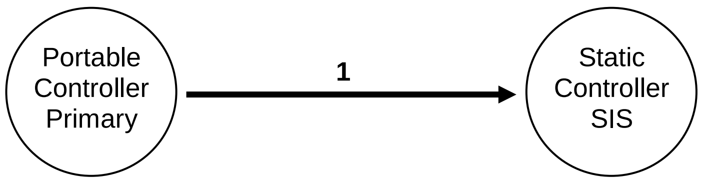


Figure 17. Including a SIS to the network.

Portable Inclusion Controller 1
Static Controller Primary SIS

Figure 18. After SIS is included to the network.

1. Include the Static Controller as described in the previous chapter by replicating the information from the Primary Controller to the Static Controller. The Static Controller is now included to the network and is secondary. Afterwards will the application on the Primary Controller assign the Secondary Static Controller to become the SIS. The SIS becomes the Primary Controller after inclusion and the old Primary Controller becomes an Inclusion Controller.

It is recommended to include the SIS as the first node in the network because new nodes can then be advised about its presence when included.
Adding a SIS to an existing network will cause battery-operated nodes, such as portable controllers, to be unaware of the SIS. Such nodes will never be able to request topology updates from the SIS.

9.2 Adding inclusion controllers to the network

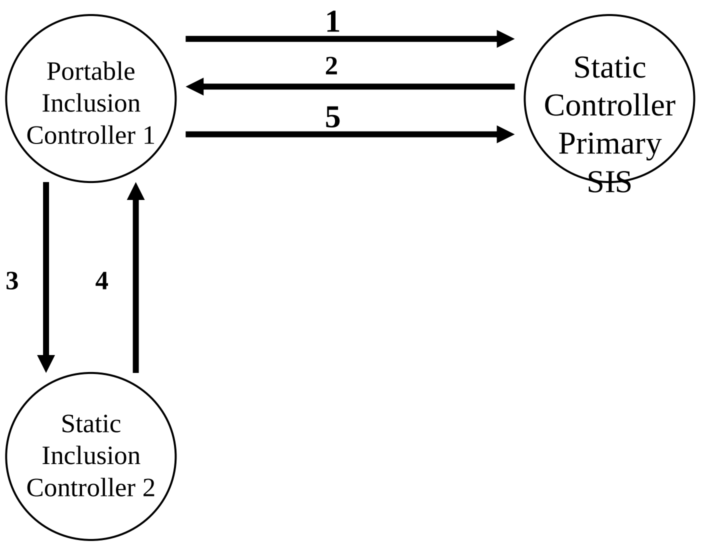

1
2
5
Portable Inclusion Controller 1
Static Controller Primary SIS
3
4
Static Inclusion Controller 2

Figure 19. Including an Inclusion Controller to the network.

1. Inclusion Controller 1 requests an available node ID for the new Static Inclusion Controller 2 and the latest network topology from the SIS

2. The SIS provides a node ID and the latest network topology to the Inclusion Controller 1.

3. Inclusion Controller 1 now assigns the allocated node ID to Static Inclusion Controller 2. Based on the latest network topology the Static Inclusion Controller 2 check for neighbors within direct range. (Check for neighbors within direct range is not done in case the Inclusion Controller 2 is portable)

4. The Inclusion Controller 2 returns neighbor information to Inclusion Controller 1.
   In case Controller 2 is a static controller based on an old library without Inclusion Controller capability then it becomes a Secondary Static Controller.

5. The Inclusion Controller 1 updates the SIS with the latest network topology discovered during the inclusion of Inclusion Controller 2.

10 NETWORK-WIDE INCLUSION EXAMPLE

This chapter discusses inclusion of nodes using network-wide inclusion via explorer frames.

10.1 Initial Setup

Inclusion mode is enabled in the primary controller. Typically this is done via a web page or other sort of GUI.

When in inclusion mode, the controller accepts one inclusion requests, if multiple nodes . Inclusion mode must time out after a period of time when no new inclusion requests have been received.

The most basic setup is one controller node and one slave node, as depicted in Figure 20.

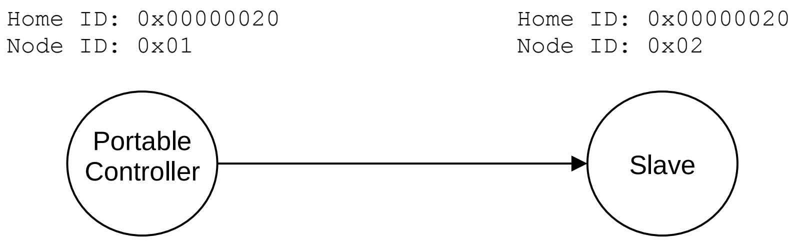


Figure 20. Including a Slave node.

Initially both the controller and the slave node are reset. The controller may have a predefined Home ID (e.g. 0x00000020) and a Node ID =0x01. The slave may have a predefined (non-zero) Home ID and Node ID = 0x00. Being the first controller in the system and therefore determining the Home ID, the controller defaults to become the Primary Controller and takes node ID = 0x01.

Inclusion mode is enabled in the controller. Auto-inclusion is enabled in the slave. This may be done by activating a special key sequence in the slave or simply resetting the slave. The slave chooses a new random home ID after each reset.

The controller assigns the Home ID = 0x00000020 and the Node ID = 0x02 to the slave node. The slave is now part of the Z-Wave network, but no specific association has been made between the two nodes.

10.2 Adding additional nodes

New slave nodes can be included in the Z-Wave network as the network is gradually expanded. In Figure 21, the example is extended to include another slave node.

If the new node is within direct range of the controller, the inclusion happens the exact same way as for the first slave node. If out of range, the new node cannot reach the controller. Then, an explore frame is issued.

The explorer frame carrying the InclusionRequest is forwarded by the already included slave node. The following inclusion process resembles classic Z-Wave inclusion with the minor difference that the new node uses a non-zero home ID and the controller AssignId message may be routed rather than direct range.

The second slave node will be assigned the Home ID = 0x00000020 and the Node ID = 0x03.
The result is depicted in Figure 21.

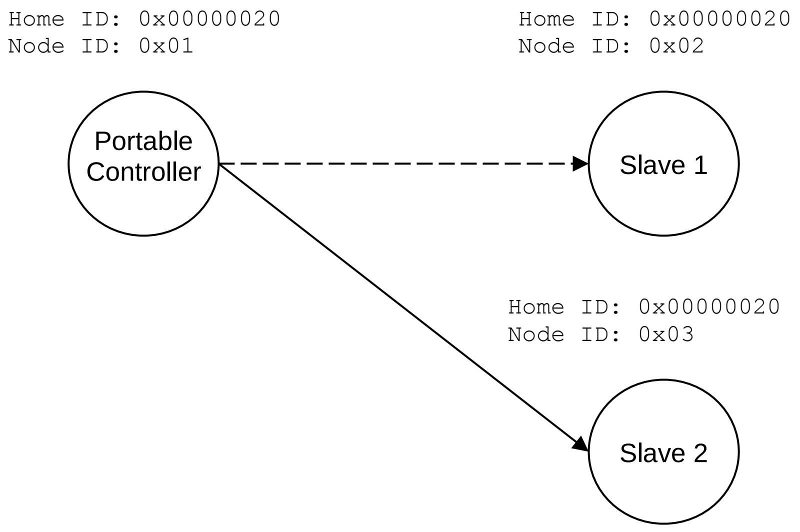


Figure 21. Including additional nodes.

10.3 Removing Nodes

A node can be removed from the Z-Wave network by activating the exclude initiator on the Primary Controller and the node initiator. This action will cause the controller to send Home ID = 0x00000000 and Node ID = 0x00 to the node, thereby resetting the node. The controller will also remove the routing information from the routing table.

10.4 Recover from Controller Node Error

If operating a mixed network containing non-explorer Z-Wave nodes, refer to section 6.7.

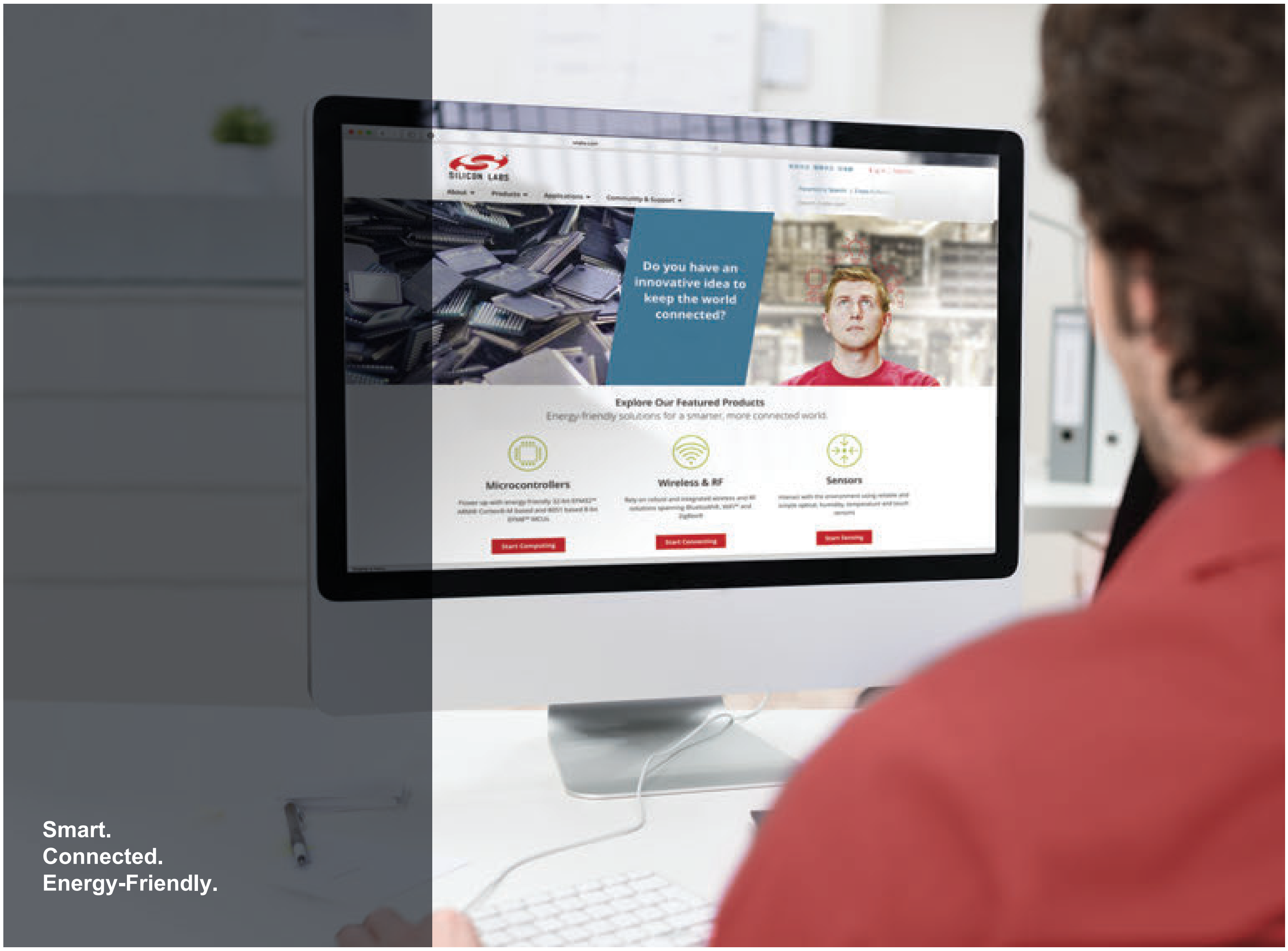
If all nodes support explorer route resolution, reconstruction of the route map is not important. Any node may send out an explorer SearchRequest frame to locate the destination node in question.

However, since the distribution of node IDs are managed by the primary controller, the loss of the primary controller still means that the network has to be re-built.

Every explorer-supporting controller reset operation causes the controller to choose a new homeID every time. This means that there is no risk of duplicate nodeIDs; even if the controller is just reset and used again.

11 REFERENCES

[1] Silicon Labs, INS12308, Instruction, Z-Wave 500 Series Appl. Prg. Guide v6.51.0x


Smart.
Connected.
Energy-Friendly.
Do you have an innovative idea to keep the world connected?
Explore Our Featured Products
Microcontrollers
Wireless & RF
Sensors

Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

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

http://www.silabs.com