


A DEEP DIVE INTO THE Z-WAVE BINDING

Chris Jackson



PRESENTATION OVERVIEW

- ▶ What is Z-Wave
 - ▶ Z-Wave protocol overview
 - ▶ Key Z-Wave concepts
 - ▶ openHAB binding overview
 - ▶ Binding roadmap
- 
- A decorative graphic consisting of several parallel white lines of varying lengths, slanted upwards from left to right, located in the bottom right corner of the slide.

Z-WAVE OVERVIEW

- ▶ Z-Wave is...
 - ▶ A wireless home automation protocol
 - ▶ An alliance of companies
- ▶ All Z-Wave hardware uses Sigma chips
- ▶ Z-Wave has been a closed protocol
 - ▶ Devices and protocol managed by Sigma and Z-Wave Alliance
 - ▶ Certified devices undergo a certification scheme to ensure compatibility
 - ▶ This has allowed it to provide a high degree of interoperability
 - ▶ Companies are under an NDA not to disclose information
 - ▶ This means that open source projects needed to rely on reverse engineering the protocol
- ▶ Approx 600 companies and 2200 certified devices



Z-WAVE PLUS


- ▶ Introduced in 2015 and backward compatible with the original standard
- ▶ Mixture of software and hardware changes
- ▶ Improved devices with lower battery consumption, coupled with improved protocol functionality
 - ▶ Increased data rate from 40kb/s to 100kb/s
 - ▶ Lower power consumption for better battery life
 - ▶ Higher output power for better range
 - ▶ Better routing options with addition of Explorer frames
 - ▶ Additional command classes to support new device management



Z-WAVE PUBLIC

- ▶ In 2016 Z-Wave Public was launched (www.zwavepublic.com)
 - ▶ Published standards for some protocol layers
 - ▶ Primarily the compatibility layer (ie command class documentation)
- ▶ NDA still required for full developer information
 - ▶ Does not open source the lower layers (ie Serial API)
 - ▶ Does not open source hardware information
 - ▶ Certification of products

KEY Z-WAVE PROTOCOL FEATURES

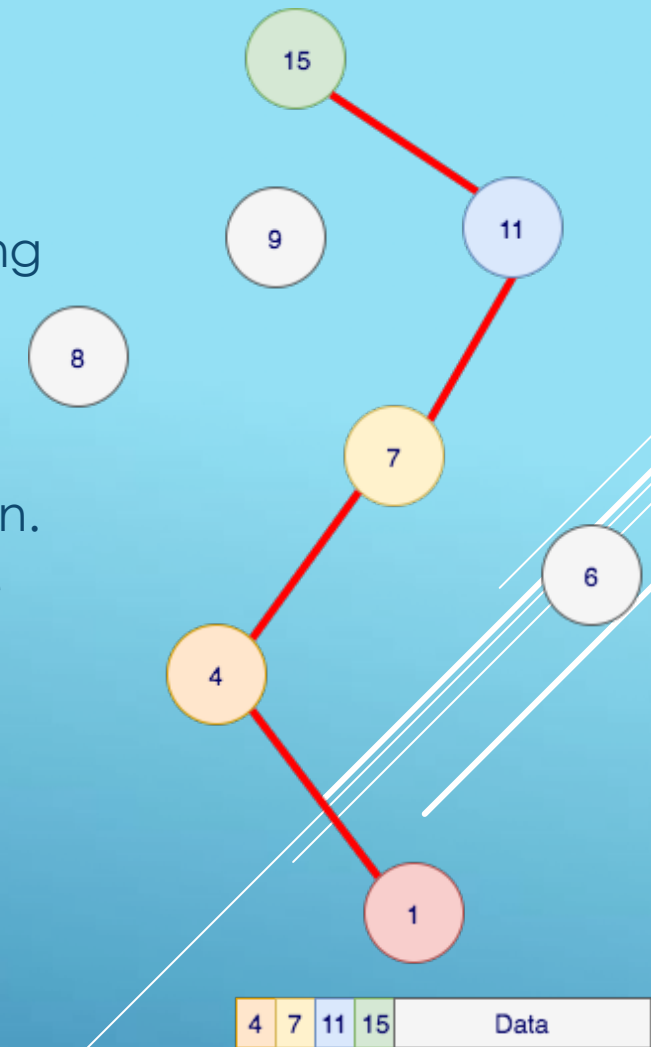
- ▶ Two way communications
 - ▶ Guarantees delivery of data or notification of failure
 - ▶ Mesh networking
 - ▶ Extends the network outside the immediate range of the controller
 - ▶ Immediate status updates
 - ▶ Reduced latency over a polled system
 - ▶ Large number of devices on a network
 - ▶ Supports 232 physical devices
 - ▶ A physical device may provide multiple virtual devices
 - ▶ High security option
- 

INCLUSION AND EXCLUSION

- ▶ Z-Wave devices must be “included” into the network before they can be used
- ▶ The **inclusion** process allocates the “Node ID” to the device and tells the device what network it is part of (the “Home ID”)
- ▶ The primary controller (or SIS) will allocate the next unused node ID (1 - 232)
- ▶ Nodes that are not included into the network will not work
- ▶ Inclusion in itself is not secure, so devices can send messages on a network without being included
 - ▶ Secure inclusion prevents this with only a small window of opportunity to be hacked during the inclusion
- ▶ **Exclusion** will remove the device from the controller and reset it

ROUTING

- ▶ Z-Wave uses a “*source routing*” mesh network
 - ▶ This means that the sender (and controller) is responsible for defining the routes.
 - ▶ Up to four routers can be traversed between the source and destination
- ▶ The controller uses “*Explorer Frames*” to derive a route to the destination.
 - ▶ For each frame that is transmitted, the controller will make multiple attempts.
 - ▶ Try the last known working route
 - ▶ Derive a route using the Explorer frame
- ▶ Routes are defined by the controller, and static routes between nodes are configured during the heal

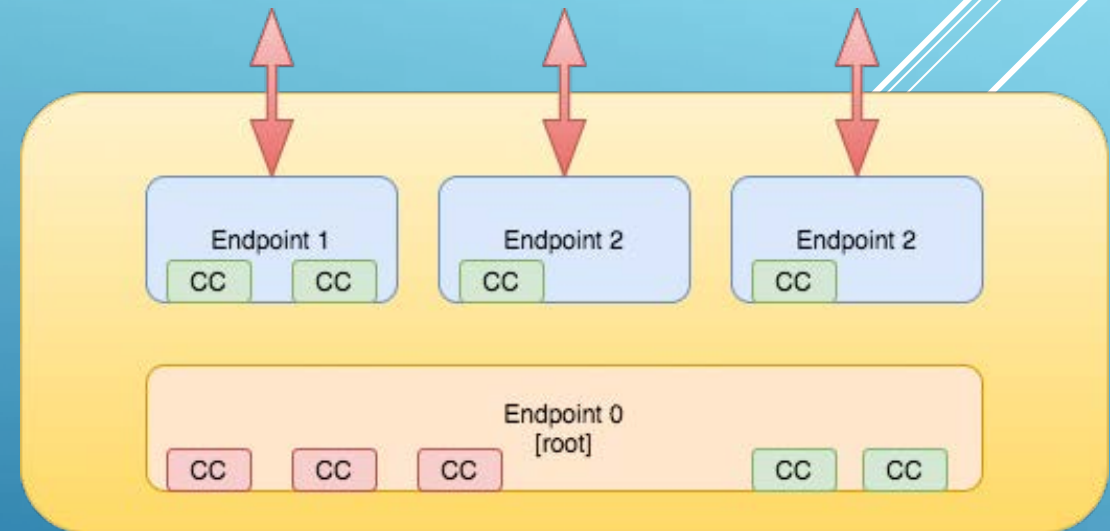


COMMAND CLASSES

- ▶ Command Classes are the heart of the operability layer
 - ▶ They define groups of functionality that is implemented in a standard way
 - ▶ A command class will normally define a number of commands related to the overall function the class implements
 - ▶ Normally includes functions to describe the device
 - ▶ As the standards evolve, the classes increase in version
 - ▶ Generally, backward compatibility is maintained
 - ▶ Z-Wave certification ensures that these functions are implemented correctly, and therefore ensures that different devices are compatible
- ▶ Currently approximately 120 command classes are defined

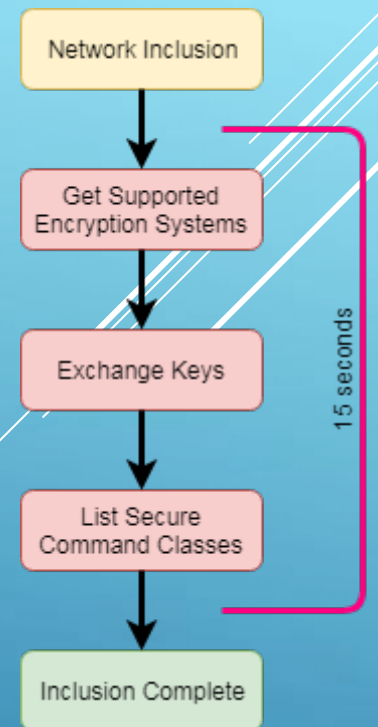
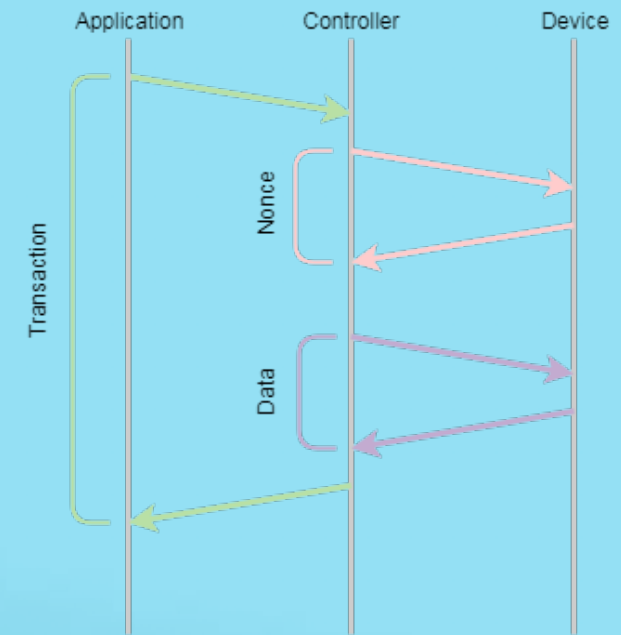
THE ANATOMY OF A DEVICE

- ▶ A Z-Wave device contains a number of logical endpoints
 - ▶ Always a root endpoint (Endpoint-0)
 - ▶ Normally a number of other endpoints
- ▶ The root endpoint provides management functions and primary functionality
 - ▶ E.g. Encapsulation methods, security, maintenance...
 - ▶ Basic control capabilities
- ▶ Other endpoints provide specific functionality
 - ▶ Switches
 - ▶ Temperature
 - ▶ Notifications



SECURITY

- ▶ Z-Wave provides a security layer
 - ▶ Original “S0” security command class
 - ▶ Newly defined “S2” security command class
- ▶ Provides 128 bit AES encryption
- ▶ A “secure inclusion” is required to perform the key exchange
 - ▶ This must complete within 15 seconds of the network inclusion
 - ▶ Uses a “well known” key to transfer the network key
 - ▶ Minor weakness in the security
- ▶ If the security key exchange fails, a device *MUST* be excluded from the network before it can be re-included
- ▶ Each application transaction requires two communications with the device
 - ▶ A NONCE (Number – used ONCE) which is only valid for 10 seconds
 - ▶ The encrypted command message (encrypted with the NONCE)

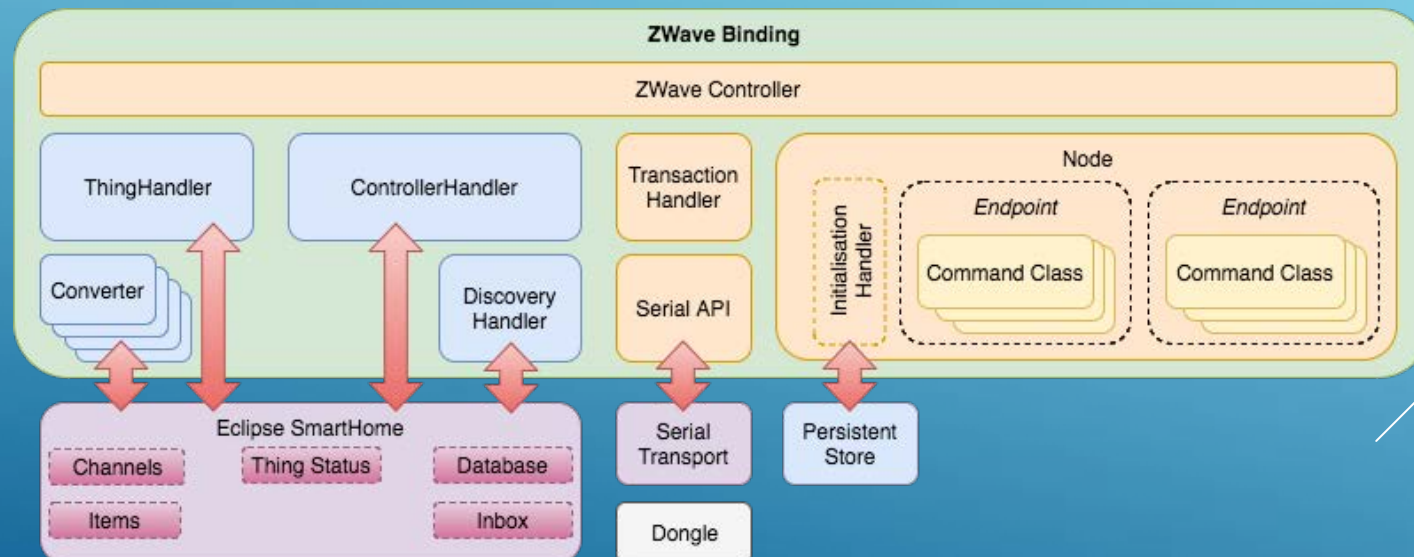


OPENHAB BINDING OVERVIEW



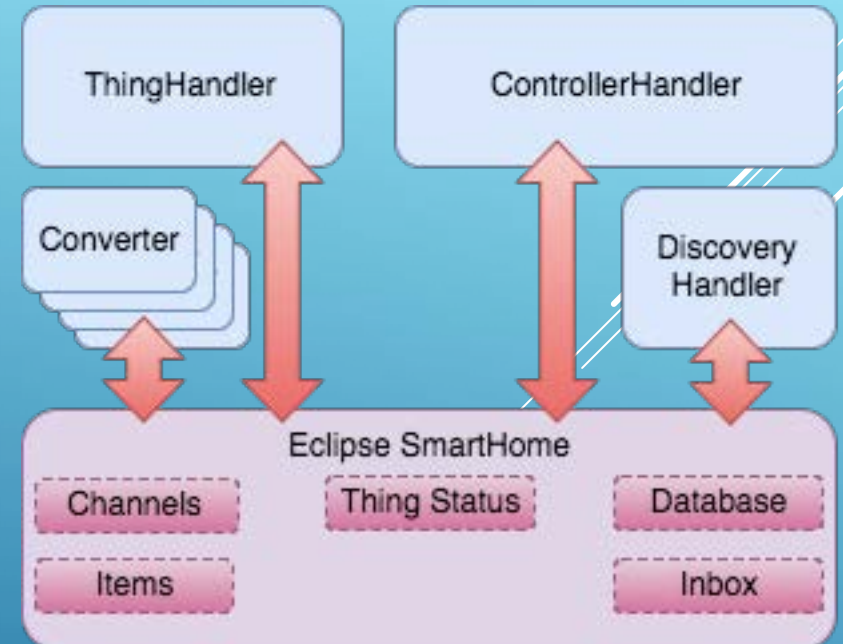
BINDING ARCHITECTURE

- ▶ The Z-Wave binding is broadly split into two parts
 - ▶ Z-Wave protocol stack
 - ▶ Handles all the protocol layers, commands etc
 - ▶ ESH/OH interface handlers
 - ▶ Converts Z-Wave data to ESH data types
 - ▶ Manages thing types and system notifications, discovery...



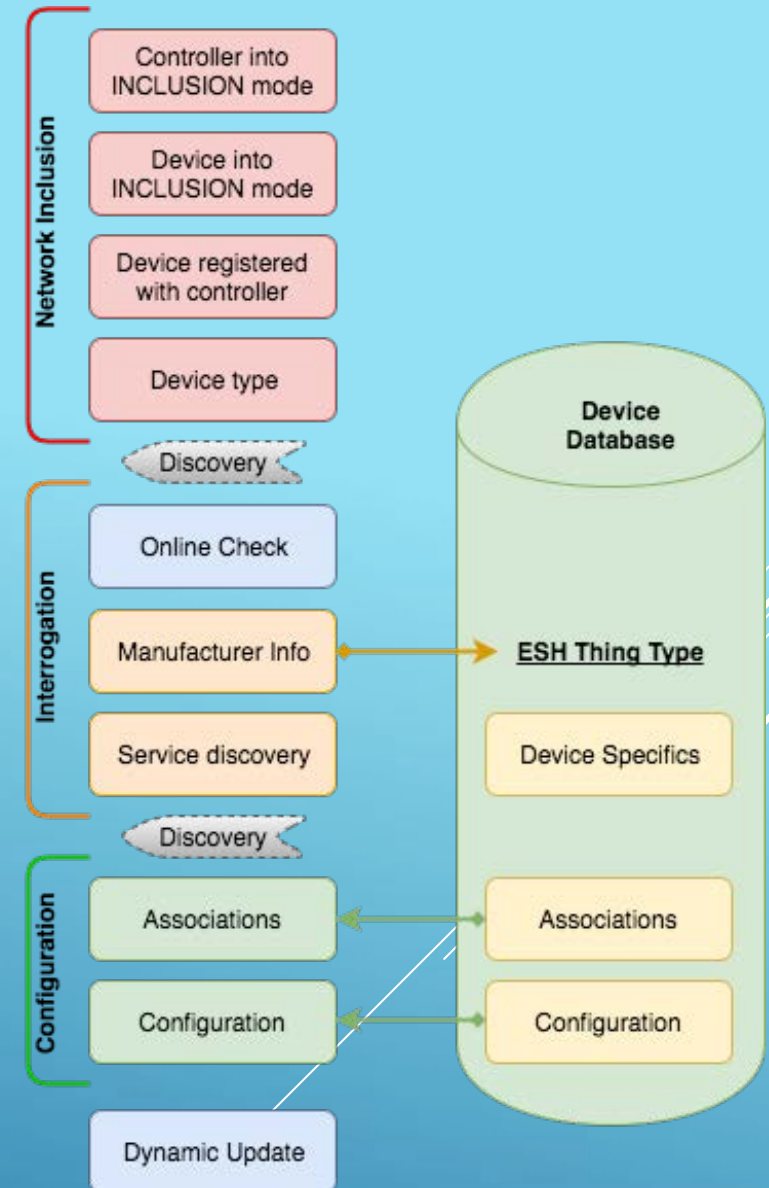
OPENHAB INTERFACE LAYERS

- ▶ The openHAB specific layers provide an interface between the stack, and the ESH framework
- ▶ Mostly, these are thin interface layers to perform conversions or other housekeeping for ESH
 - ▶ Controller Handler: One per controller
 - ▶ Discovery: Linked to controller
 - ▶ Thing Handlers: One per Z-Wave node
 - ▶ Converters: One per channel




DEVICE INITIALISATION

- ▶ There are three main phases to initialise a device
 - ▶ Network inclusion including secure inclusion
 - ▶ Device interrogation
 - ▶ Configuration
- ▶ Two “Discovery Points” are used
 - ▶ Discovery when the device is included, but unknown
 - ▶ Updated once the services and thing type are known
- ▶ Manufacturer ID, device type and device ID
 - ▶ Links the device to the database
- ▶ Config only performed if “master controller” is set

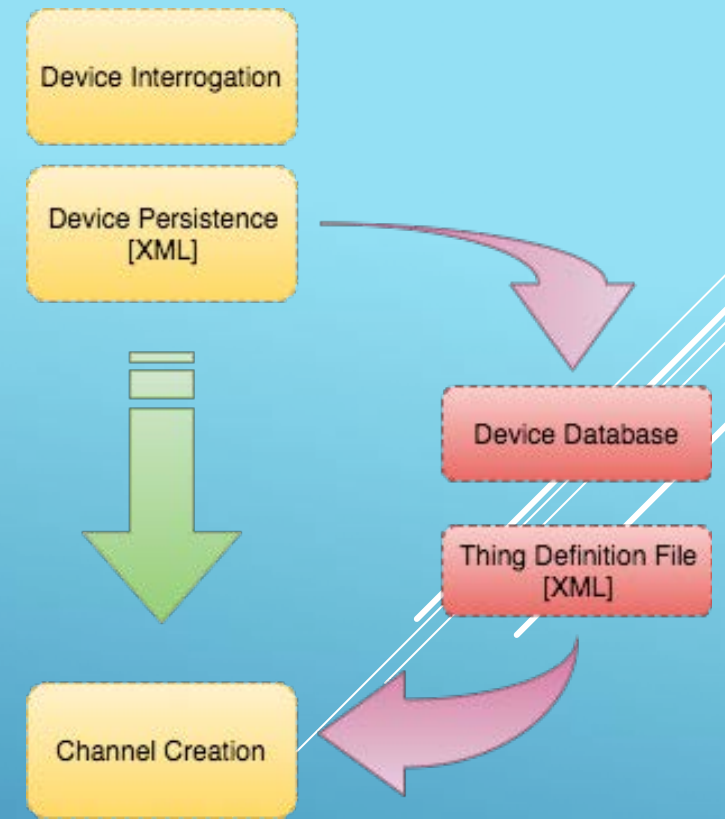


DEVICE INTERROGATION PHASE

- ▶ The purpose is to read information from the device to find out what the device is, and services it supports
 - ▶ All command classes are read to see what features they support
 - ▶ Command versions are checked
 - ▶ The interrogation is only performed when the device is initially installed
 - ▶ Information from the interrogation is saved by the binding in an XML file
 - ▶ When the binding restarts, the information is read from the XML rather than performing the interrogation
- 

THING DEFINITION

- ▶ Thing definition is currently a multi stage process
 - ▶ Firstly the device is discovered
 - ▶ This discovery data is imported into the database
 - ▶ Additional meta data is added to the database
 - ▶ A thing definition file is then exported into the binding
 - ▶ Channels can then be created
- ▶ This workflow was required by ESH when the binding was written
 - ▶ There was no ability to define channels outside of the XML
 - ▶ This is also necessary for most devices
 - ▶ Definition of configuration and association data
 - ▶ Options to resolve device bugs and workarounds
- ▶ Future plan to bypass this where possible



DATABASE OVERVIEW

- ▶ There are thousands of Z-Wave devices on the market
 - ▶ The binding currently contains definitions for around 620 devices (and growing quite quickly!).
- ▶ Z-Wave devices are largely self describing in their functionality
 - ▶ Currently the binding saves this information into an XML file
 - ▶ The XML is used to generate channels in the database
 - ▶ This in turn creates the ESH Thing definitions
- ▶ Many devices contain configuration parameters
 - ▶ These need to be defined so they can be presented to the user

Manufacturer	Fibergroup
Manufacturer ID	010F
Name	FGMS001
Device Description	Motion Sensor
References (Type ID)	0800.1001,0800.2001,0800.3001,0801.1001,0801.2001
Firmware Versions	All up to 2.8
Protocol Version	3.067
Library Type	LIB_SLAVE_ENHANCED
Always Listening	No
Frequently Listening	No
Supports Routing	Yes
Supports Beaming	Yes
Supports Security	No
Maximum Data Rate	40000
Device Overview	
Inclusion Information	
Exclusion Information	
Wakeup Information	
Basic Class	BASIC_TYPE_ROUTING_SLAVE
Generic Class	GENERIC_TYPE_SENSOR_BINARY
Specific Class	SPECIFIC_TYPE_ROUTING_SENSOR_BINARY
Unique Reference	fgms001



Parameter	8																		
Bitmask																			
Label	PIR Sensor Operating Mode																		
Description	The parameter determines the part of day in which the PIR sensor will be active.																		
Word Size	1																		
Default Value	0																		
Allowable Range	0 to 255																		
Overview	The parameter determines the part of day in which the PIR sensor will be active. This parameter influences only the motion reports and associations. Tamper, light intensity and temperature measurements will be still active, regardless of this parameter settings. 0 - PIR sensor always active 1 - PIR sensor active during the day only 2 - PIR sensor active during the night only. Default setting: 0 Parameter size: 1 [byte]																		
Options	<table><tr><td>Option Value</td><td>0</td></tr><tr><td>Option Label</td><td>PIR sensor always active</td></tr><tr><td>Option Overview</td><td></td></tr><tr><td>Option Value</td><td>1</td></tr><tr><td>Option Label</td><td>PIR sensor active during the day only</td></tr><tr><td>Option Overview</td><td></td></tr><tr><td>Option Value</td><td>2</td></tr><tr><td>Option Label</td><td>PIR sensor active during the night only</td></tr><tr><td>Option Overview</td><td></td></tr></table>	Option Value	0	Option Label	PIR sensor always active	Option Overview		Option Value	1	Option Label	PIR sensor active during the day only	Option Overview		Option Value	2	Option Label	PIR sensor active during the night only	Option Overview	
Option Value	0																		
Option Label	PIR sensor always active																		
Option Overview																			
Option Value	1																		
Option Label	PIR sensor active during the day only																		
Option Overview																			
Option Value	2																		
Option Label	PIR sensor active during the night only																		
Option Overview																			

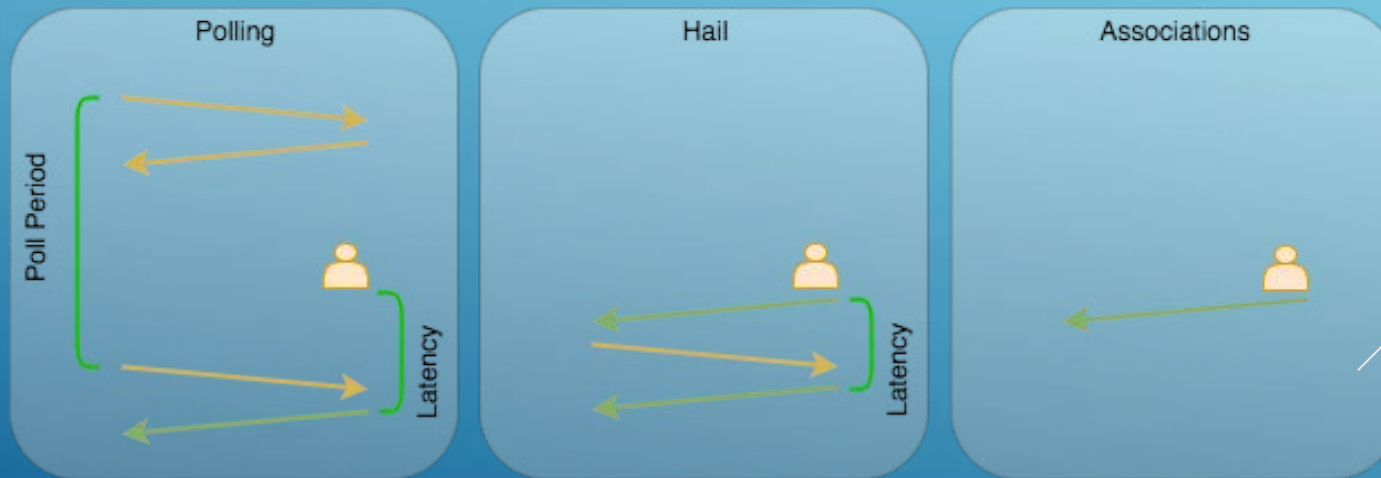
WORKING WITH DEVICE “FEATURES”

- ▶ Many devices have “features” that make them operate differently than expected by the Z-Wave standards
 - ▶ The Z-Wave certification program should catch these, but it doesn't always!
 - ▶ The binding should work with as many devices as possible
 - ▶ Workarounds have been coded into the binding to work around these “features”
 - ▶ These workarounds are normally enabled through configuration options in the database
 - ▶ This keeps the code as clean as possible, allowing these changes to be enabled in a configuration file.

Time	Node	Entry
18:24:38.131	13	AddNodeToNetwork ADD_NODE_ACK HIGH POWER
18:24:38.140	13	AddNodeToNetwork ADD_NODE_STATUS_LEARN_READY
18:24:44.777	13	AddNodeToNetwork ADD_NODE_STATUS_LEARN_READY
18:24:44.924	13	AddNodeToNetwork ADD_NODE_STATUS_LEARN_READY
18:24:44.960	13	AddNodeToNetwork ADD_NODE_STOP
18:24:45.015	13	AddNodeToNetwork ADD_NODE_STATUS_DONE
18:24:45.035	13	Stage advanced to IDENTIFY_NODE
18:24:45.040	13	IdentifyNode
18:24:45.045	13	IdentifyNode ROUTING_SLAVE SWITCH_REMOTE_ANALYSE LISTENING FLUSHED FLUSHED ROUTING BEAMING
18:24:45.050	13	Stage advanced to MANUFACTURER
18:24:49.681	13	ApplicationCommandHandler MAKE_UP_MODIFICATION
18:24:49.721	13	Node is NAME
18:24:49.726	13	SendData 132 MANUFACTURER_SPECIFIC_GET
18:24:49.728	13	SendData 132 ACCEPTED BY CONTROLLER
18:24:49.748	13	SendData 132 ACK RECEIVED from device in 27ms
18:24:49.763	13	ApplicationCommandHandler MANUFACTURER_SPECIFIC_REPORT 00000000
18:24:49.806	13	Stage advanced to SECURITY_REPORT
18:24:49.810	13	Stage advanced to ZPP_VERSION
18:24:49.820	13	SendData 132 MANUFACTURER_SPECIFIC_GET
18:24:49.828	13	SendData 132 ACCEPTED BY CONTROLLER
18:24:49.847	13	SendData 132 ACK RECEIVED from device in 27ms
18:24:49.856	13	ApplicationCommandHandler MANUFACTURER_SPECIFIC_REPORT 00000000

STATUS UPDATES

- ▶ Z-Wave binding supports 3 methods for status update
 - ▶ Polling: High latency, high occupancy
 - ▶ Hail, Low latency, low occupancy
 - ▶ Association: Lowest latency, lowest occupancy
- ▶ Polling in the binding has the lowest priority to avoid saturating the network
- ▶ Best compromise is long polling period for lifeline, and associations for reporting




BINDING ROADMAP



DEVELOPMENT VERSION

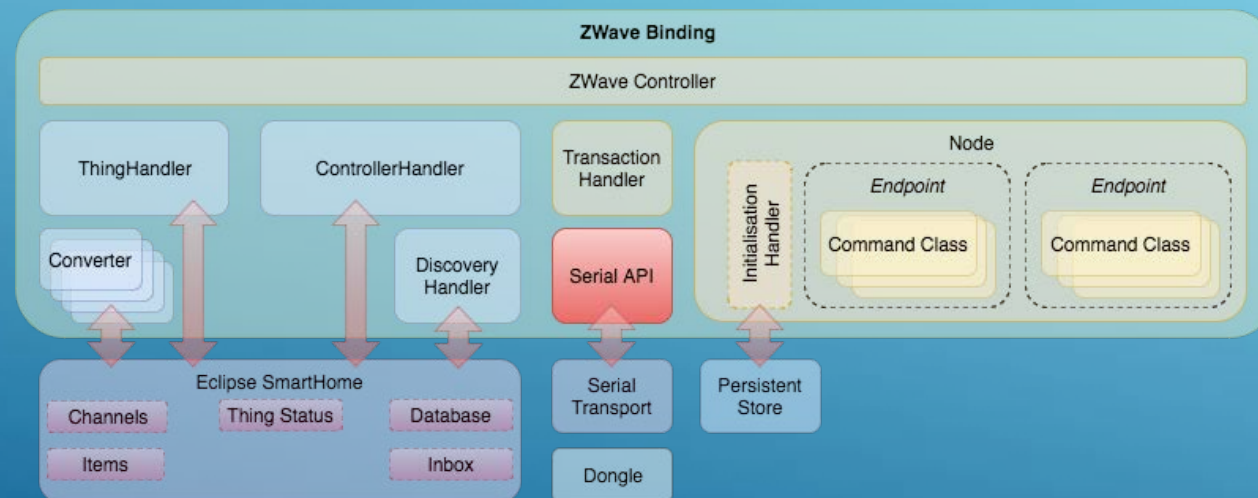
- ▶ Currently there is a “Development Version” of the binding
 - ▶ Includes some significant changes from the current master
 - ▶ S0 Security, improved associations
 - ▶ This has been aligned with the Z-Wave Standards and has a number of “breaking changes” wrt the current maser
 - ▶ As further breaking changes are planned, this has not been merged to reduce the pain to those using the binding
 - ▶ Maybe it is best to merge and live with multiple “breaking” versions?

MAJOR UPCOMING FEATURES

- ▶ With the release of parts of the standard, “reverse engineered” implementations are being improved
 - ▶ Many new features are planned for the binding -:
 - ▶ Backup / restore of dongle configuration
 - ▶ Statistics / routing information
 - ▶ IMA (Installation Maintenance Application)
 - ▶ Network health
 - ▶ Link quality information between devices
 - ▶ Packet loss statistics
 - ▶ Reduced database dependency
- 

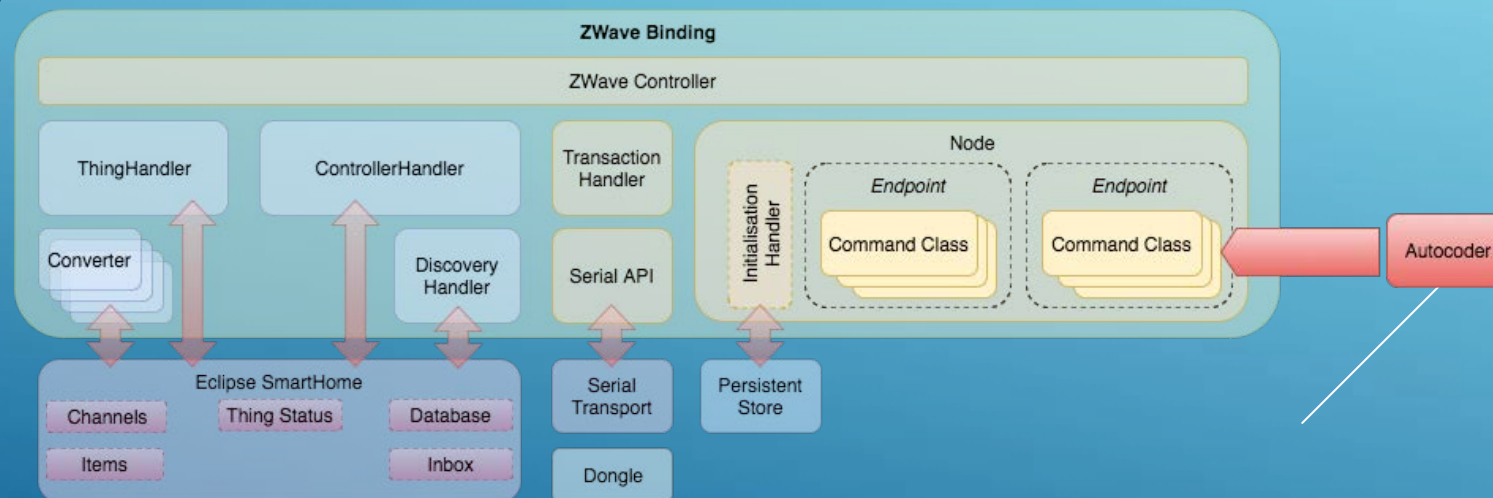
SEPARATION OF TRANSPORT LAYER

- ▶ The Serial API will not be made public. To allow the binding to implement features that are not public, the (small) Serial API component should be separated
 - ▶ This would result in two versions of the transport element - an open source version as now, and a closed source version.
 - ▶ Closed source transport should still be freely available within openHAB.



AUTO CODE GENERATION

- ▶ Much of the Z-Wave protocol is defined by the Command Classes
 - ▶ Detailed, and complex formatting of commands
 - ▶ Errors here are difficult to find and directly impact compatibility
- ▶ Generation of command classes can be automated from the public documentation
 - ▶ This would ensure 100% coverage of the protocol and compatibility with the standard



- ▶ Z-Wave provides a mature, reliable home automation environment with many options for devices from multiple manufacturers
- ▶ openHABs Z-Wave binding has support for over 600 devices with a wide range of functionality
- ▶ Further developments are planned that will further improve the system for users

THANK YOU

A decorative graphic consisting of several parallel white lines of varying lengths, slanted upwards from left to right, located in the bottom right corner of the slide.