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

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

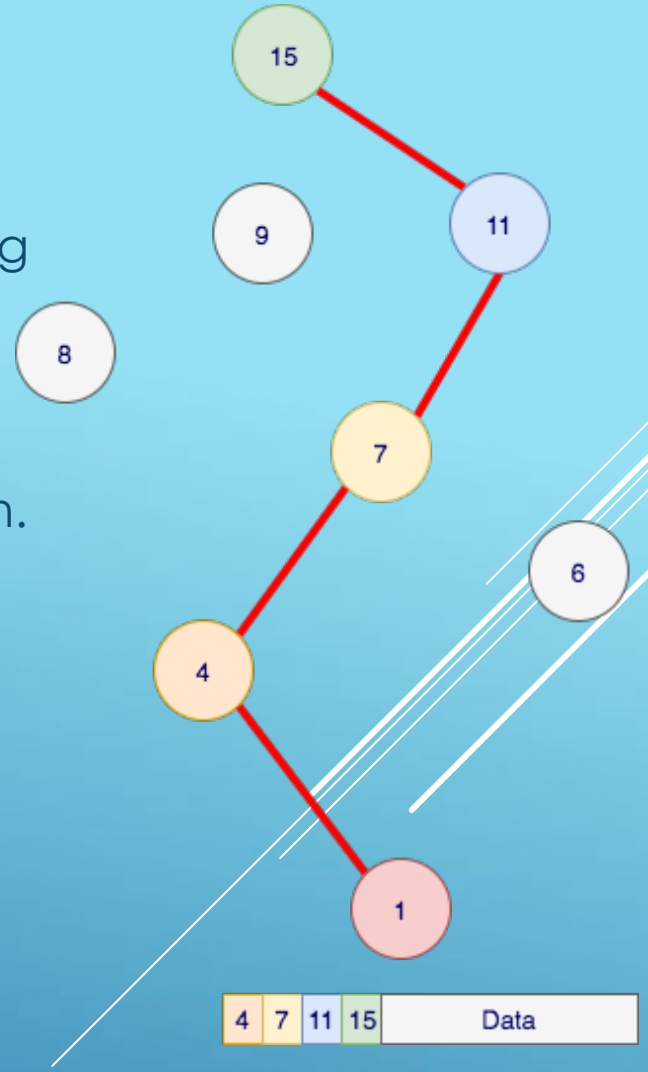
- Two way communications
  - Guarantees delivery of data or notification of failure
- Mesh networking
  - Extends the network outside the immediate range of the controller
- Immediate status updates
  - Reduced latency over a polled system
- Large number of devices on a network
  - Supports 232 physical 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

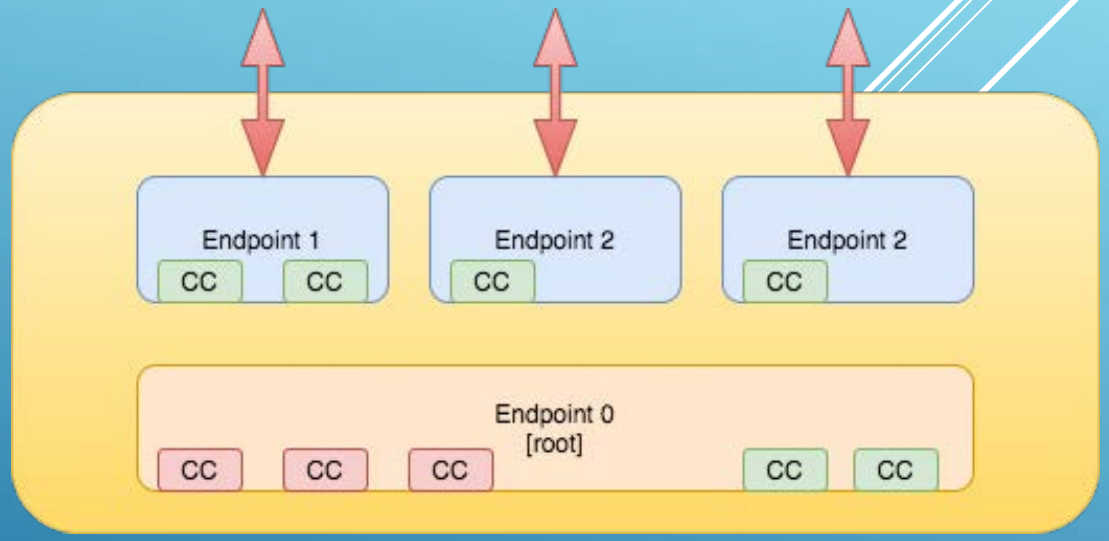


COMMAND CLASSES

- 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

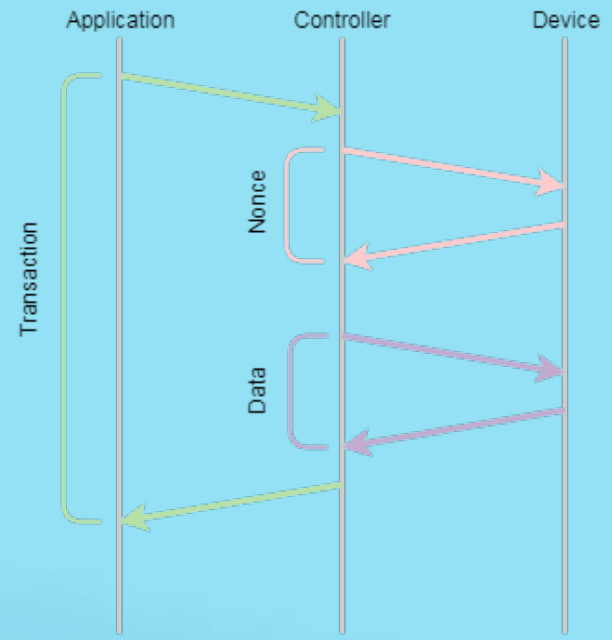
THE ANATOMY OF A DEVICE

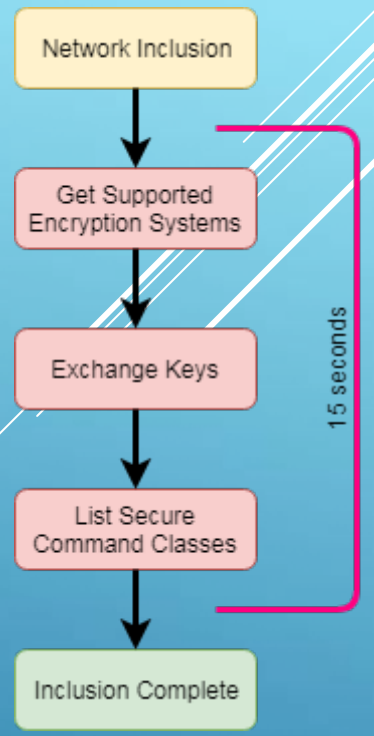
- A Z-Wave device contains a number of logical endpoints
  - Always a root endpoint (Endpoint-0)
  - Normally a number of other endpoints
- The root endpoint provides management functions and primary functionality
  - E.g. Encapsulation methods, security, maintenance…
  - Basic control capabilities
-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

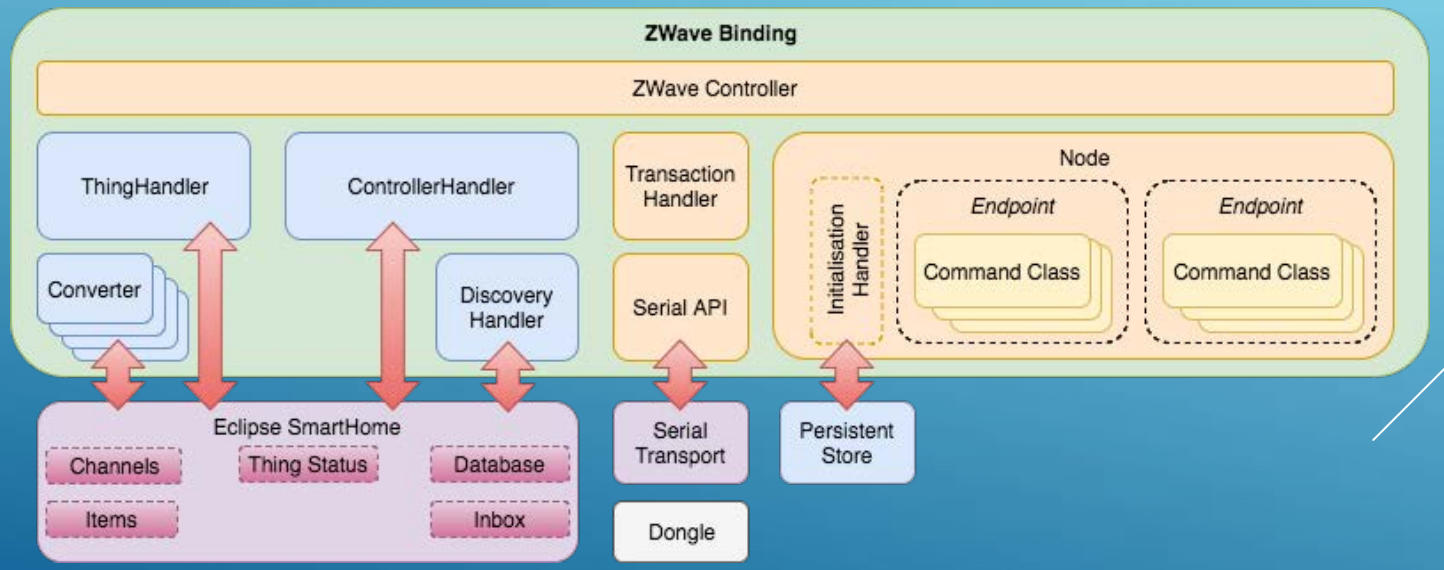




OPENHAB BINDING OVERVIEW

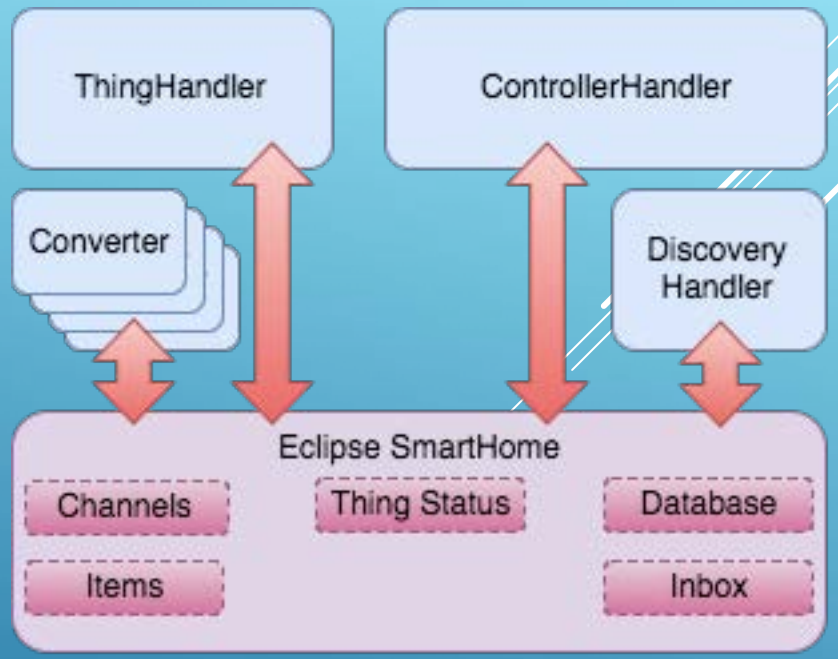
BINDING ARCHITECTURE

- 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

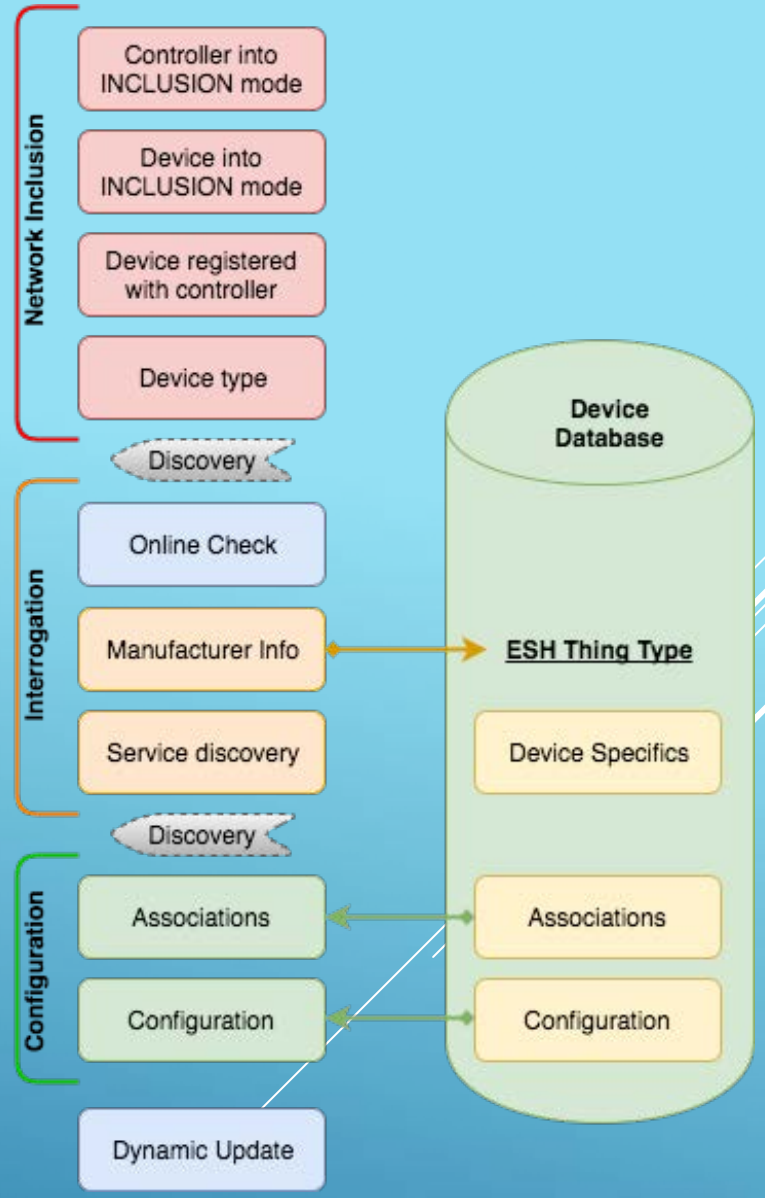
- The openHAB specific layers provide an interface between the stack, and the ESH framework
- Mostly, these are thin interface layers to perform conversions or other housekeeping for ESH
  - Controller Handler: One per controller
  - Discovery: Linked to controller
  - Thing Handlers: One per Z-Wave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

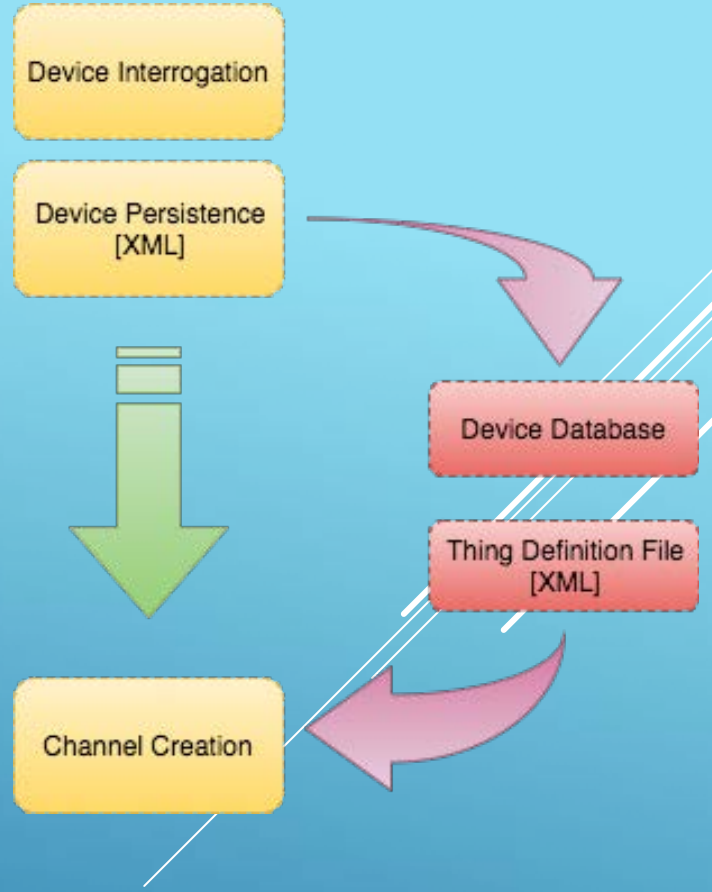


DEVICE INTERROGATION PHASE

- 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

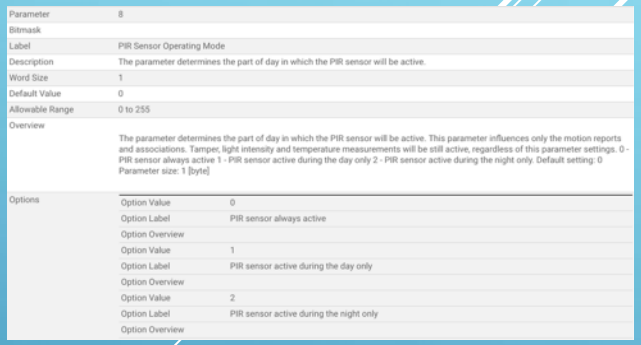
THING DEFINITION

- 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



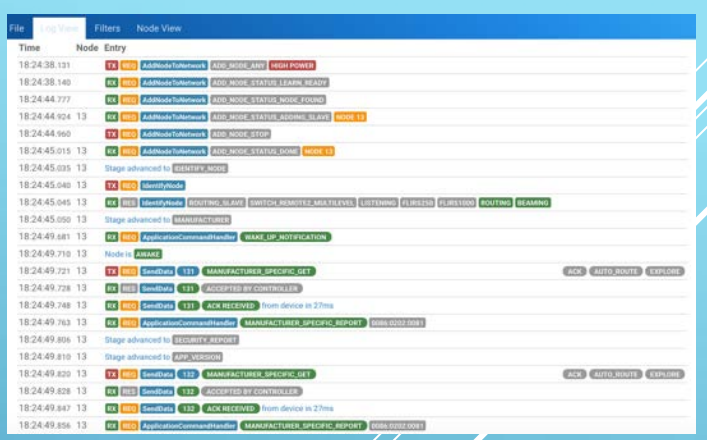
DATABASE OVERVIEW

- 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



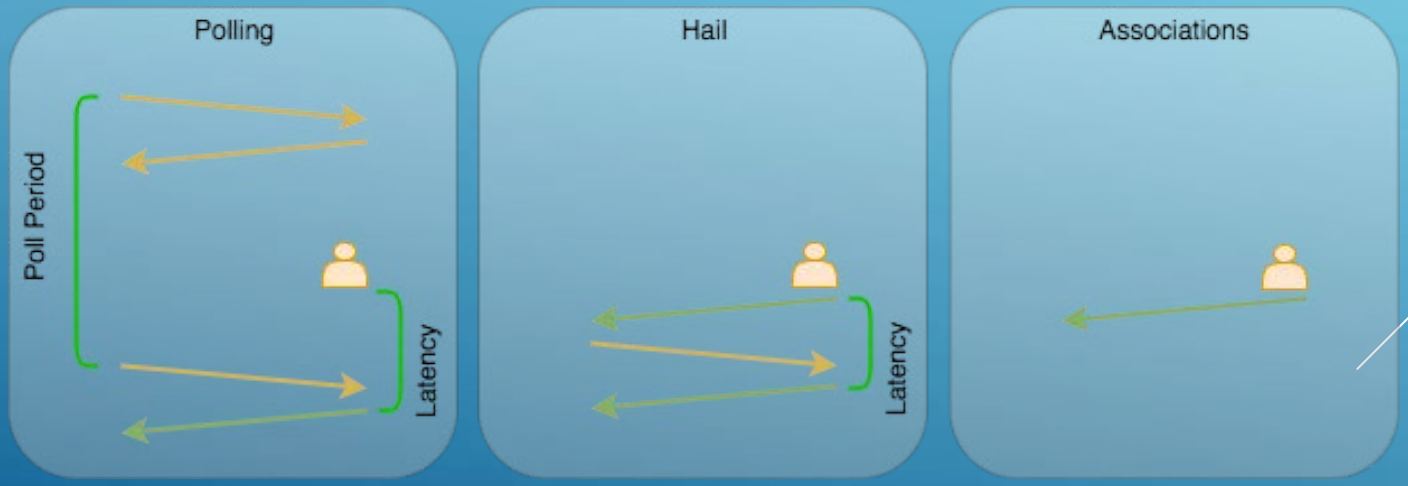
WORKING WITH DEVICE "FEATURES"

- 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.



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

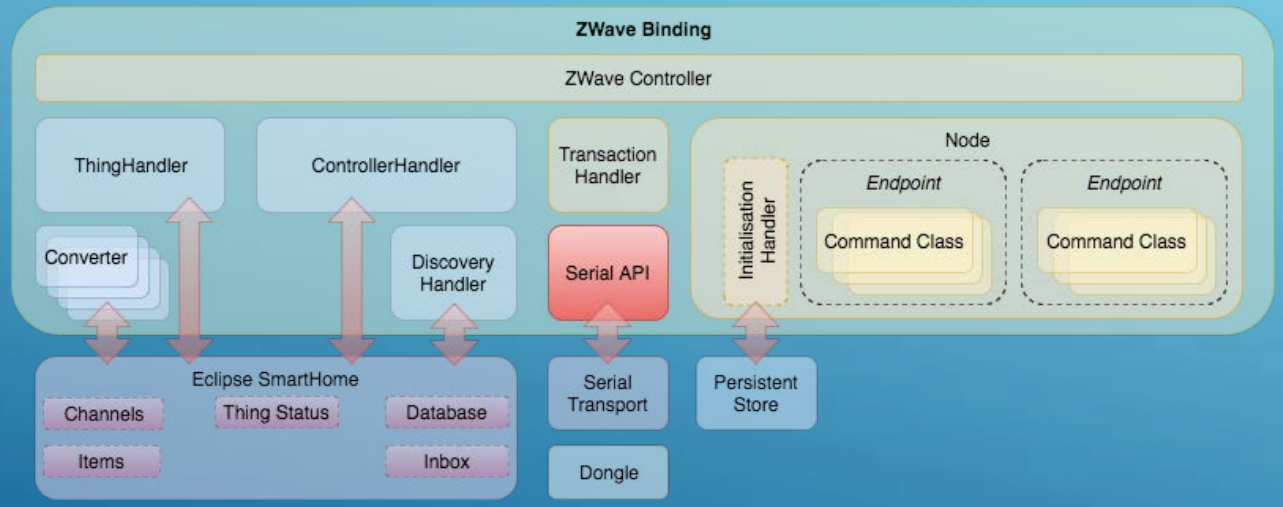
- Currently there is a “Development Version” of the binding
  - Includes some significant changes from the current master
    - S0 Security, improved associations
  - This has been aligned with the Z-Wave Standards and has a number of “breaking changes” wrt the current maser
  - As further breaking changes 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

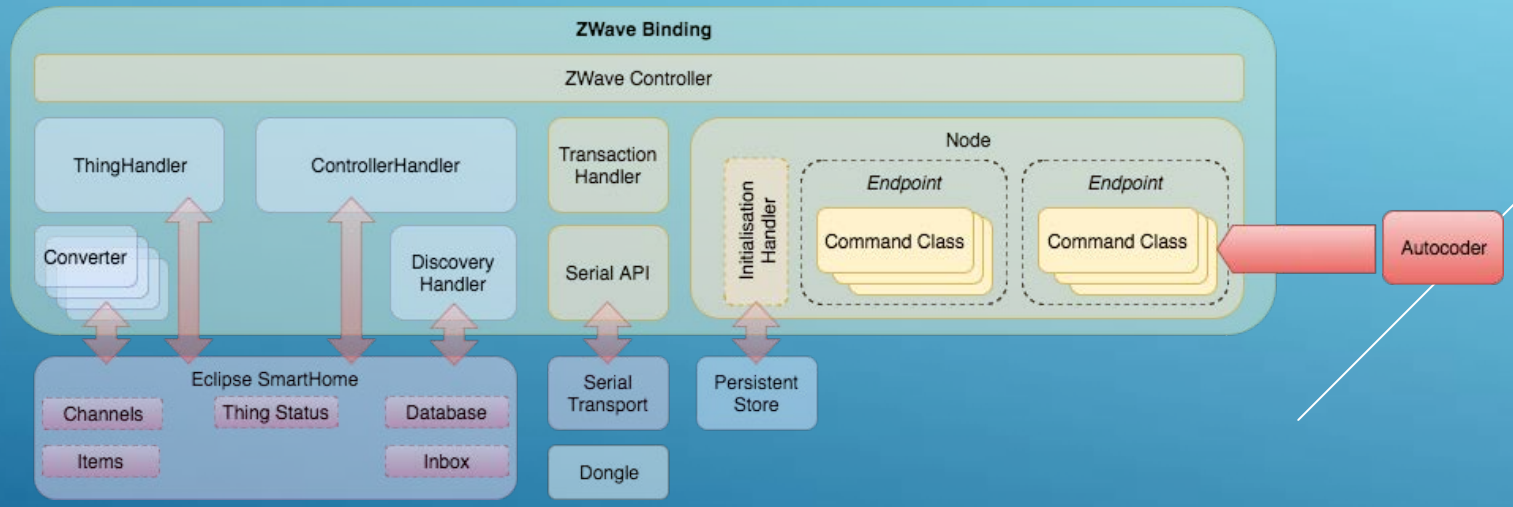
SEPARATION OF TRANSPORT LAYER

- 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