SDK 5 versus SDK 4.5 and the explorer frame –what's this all about?

Version 1.0 (Dec 2011), Dr. Christian Paetz, christian.paetz@gmail.com

All Z-Wave devices run a firmware that consists of two parts: There is a fixed part delivered by Sigma designs that covers all network related functions and there is a vendor – specific part, that the vendors define and implement according to the Z-Wave specification.

The part provided by Sigma is called a System Development Kit (SDK) and has different release numbers. Certain versions of this SDK introduced new functions. These SDKs are always backward compatible but the new functions are then available only for this SDK and subsequent SDK numbers.

The following SDKs were released:

- SDK 3.0x: First Generation of Z-Wave chip ZW0102
- SDK 3.20: introduces Static Update Controller (SUC) in 2003
- SDK 3.40: SUC ID Server (SIS) in 2005
- SDK 4.00: Second Generation of Z-Wave chip ZW0201 in 2005
- SDK 4.20: Silent Acknowledge in 2006
- SDK 5.0x: Third Generation of Z-Wave chip ZW0301 in 2007
- SDK 4.5x: Explorer Frame plus Network Wide Inclusion in 2009
- SDK 6.0x: Fourth Generation of Z-Wave chip ZW0401 in 2010

All SDKs before 3.40 can be considered as obsolete and only very few products are still in the market based on these SDKs.

All SDKs from 4.20 but not 4.5x and the SDK 5.x – typically all referred to as SDK 5 – share the same support for the basic Z-Wave network functions and processes.

The SDKs 4.5x plus all SDKs from 6.x offer an important function called explorer frame that greatly enhances the way the network is self reorganizing in case of changes. All products based on these SDKs are 100 % backward compatible to the older SDKs.

If the network consists of devices of SDK 5 but contains some devices of SDK 4.5x it will still run on the principles of the SDK. If the network only consists of devices with SDK 4.5x this network will self organize on the principles of this SDK and use its enhanced function.

To understand the difference between SDK5 and SDK4.5x some basic principles need to be understood.

Z-Wave uses a so-called source routing. This means that the sender of a message puts the full route to be traveled into the message. The message will then exactly use the route defined by the sender. This means that the sender needs to know about the structure of the network and valid routes to its peers. This structure information is usually referred to as routing table although it's technically an information about neighborhood relations between nodes in the network.

A controller can refresh its own routing table by starting a network healing process that asks all known nodes in the network to determine their neighbors and report them back to the controller.
Other sensors of data, routing slaves or battery wall controllers or remote controls typically don't have this capability but depend on a central instance in the network providing them updated routing information on request. This instance must be a static controller - always available – with the special function of updating nodes on request. This function is called "Static Update Controller = SUC".

A Z-Wave network can operate without SUC but in this case there may be problems with outdated routes in other than the primary controller in the network and in all other nodes that need routes in order to operate properly. Therefore the Z-Wave network will always try to appoint a static controller to the role of SUC to ensure a stable network communication.

SDK 4.5 introduces a function called Explorer Frame. The difference to normal messages in the Z-Wave network is that this explorer frame does not have a predefined route but its supposed to be forwarded by all nodes to all neighbors so that is will eventually reach its desired destination is there is a valid route[1]. The explorer frame can be therefore seen as a routing of last resort or a joker. It's causing some more traffic in the network but it will reach its destination regardless of the quality of the routing information in the nodes of the network.

How to deal with failed routes

The difference of the three different network configurations is shown if a message sending attempt fails due to changes in the network such as moved or disappeared nodes that acted as routers before.

1. Network with SDK 5.x without any SUC in the network

   The sender will receive a failure notification and can do nothing about it. Such a network will only be stable if all nodes are in direct range or the user makes sure that all routing tables - including those in routing slaves – are updated all the time. Whenever there is a static controller in the network that supports the SUC function it will be appointed as SUC

[1] A pruning algorithm makes sure that the explorer frames do not get routed forever saturating the network.

without further user interaction to avoid this version of the network.

2. Network with SDK 5.x with SUC in the network

   The sender will realize that a route failed and will ask the SUC for an update of the route. If the route to the SUC fails this node will try to find a route nearby that can help him to communicate with the SUC for the update desired. This function is done on the network level without any user interaction. As a result of the communication with the SUC the node will receive an update of its routing table. The SUC gets informed about a change in the network and can update its own routing table

3. Network with SDK 4.5x and 6.x

   The sender will realize that a route failed and will send the message as explorer frame. The explorer frame will eventually reach its destination and update the routing table automatically this way.

Its obvious that the explorer frame greatly enhances the stability of the network while even eliminating the need for a SUC. However in order to use explorer frames all other devices in a possible route from the sender to the receiver including the receiver must support these frames as well.

Since devices with support for explorer frames - i.e. SDK 4.5x or SDK 6.x - are only introduced recently there will be mixed networks with different SDKs for a foreseeable time. As said before if there are not sufficient SDK 4.5x devices in a network the network will still use the SDK 5.x –way to deal with changed networks and will therefore still depend on a SUC in the network.

However its not needed that 100 % of the nodes support explorer frame.

- The source node using explorer frame to find a new route needs to support explorer frame.
- Since most of the communication takes place with a static IP gateway, the IP gateway or the Z-Wave USB Stick used by a software must support explorer frames.
- Its not required that all mains powered devices support explorer frames but there should be enough devices with explorer frame support in order to allow to find new routes. The nodes with explorer frame will then become them in routers in the network, since they will volunteer as routers when a explorer frame determines a new route.
- Battery powered devices do not route and can therefore not enhance the stability of a network. However if they don't support explorer frames they will not be able to find new routes automatically and still depend on a SUC to update routes.

The Z-Wave USB Stick dilemma

Static IP Gateways typically use a Z-Wave chip with a serial API to communicate with the Z-Wave network. Unfortunately the current Generation 3 chip used in most USB Sticks and Gateways does not provide enough memory to run a 4.5 Serial API firmware that supports SUC. Therefore most Gateway manufacturers and USB Stick vendors decided to stay with SD K5.x support to be backward compatible.

Z-Wave.Me offers a ZSTICK4 that support SDK 4.5 but does not provide SUC functionality. This device can therefore only be used in a network with SDK 4.5x devices or in case there is a second Static controller with SUC support included.

Devices with 4.5x support

At the moment (December2011) only the following certified devices support the advanced SDK4.5x with explorer frames in Europe:

- All devices from FIBAR (Switch, Dimmer, Motor Control)
- All actuators from Z-Wave.Me (Switch, Dimmer, Motor Control)
- All products from BeNext, Renz GmbH, Vitrum and Fortrezz
- The ZSTICK4 from Z-Wave.Me
- The Wall controller/switch from TKB
- Everspring in wall devices (HAC, HAN)