### Instruction

Z-Wave DLL v5 User Guide

<table>
<thead>
<tr>
<th>Document No.</th>
<th>INS13113</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>2</td>
</tr>
<tr>
<td>Description</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Written By</th>
<th>JFR;SRO;BBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>2018-04-05</td>
</tr>
<tr>
<td>Reviewed By</td>
<td>JRM</td>
</tr>
<tr>
<td>Restrictions</td>
<td>Public</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approved by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>2018-04-05</td>
</tr>
</tbody>
</table>

This document is the property of Silicon Labs. The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction.
# REVISION RECORD

<table>
<thead>
<tr>
<th>Doc. Rev</th>
<th>Date</th>
<th>By</th>
<th>Pages affected</th>
<th>Brief description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20141217</td>
<td>SRO</td>
<td>All</td>
<td>Initial version. Based on INS10250-08</td>
</tr>
<tr>
<td>2</td>
<td>20180405</td>
<td>BBR</td>
<td>All</td>
<td>Added Silicon Labs template</td>
</tr>
</tbody>
</table>
Table of Contents

1 ABBREVIATIONS ................................................................................................................................. 1

2 INTRODUCTION ................................................................................................................................... 1

2.1 Purpose .............................................................................................................................................. 1

2.2 Audience and prerequisites ................................................................................................................ 1

3 WHAT IS Z-WAVE DLL ........................................................................................................................ 2

4 ARCHITECTURE AND IMPLEMENTATION ......................................................................................... 3

5 DATA FLOW ......................................................................................................................................... 3

5.1 Application Data Flow ........................................................................................................................ 5

5.2 Session Data Flow ............................................................................................................................ 5

5.3 Frame Data Flow ............................................................................................................................. 8

5.4 Transport Data Flow .......................................................................................................................... 9

6 GETTING STARTED .............................................................................................................................. 10

6.1 Check the prerequisites .................................................................................................................... 10

6.2 Limitations ........................................................................................................................................ 10

6.3 Required Z-Wave hardware and libraries ......................................................................................... 10

7 CREATION OF A Z-WAVE DLL BASED APPLICATION ..................................................................... 11

7.1 Create a .NET Windows Application project .................................................................................... 11

7.2 Add references to Z-Wave DLL components ................................................................................... 12

7.3 Example of the Inclusion node ......................................................................................................... 12

7.4 Example of the Inclusion with both nodes connected ...................................................................... 13

8 REFERENCES .................................................................................................................................... 15

List of Figures

Figure 1: Positioning of Z-Wave DLL within custom solution ................................................................. 2

Figure 2: Layers ........................................................................................................................................... 3

Figure 3: "Z-Wave DLL" Data Flow Diagram ............................................................................................ 4

Figure 4: Application Data Flow ................................................................................................................ 5

Figure 5: "Execute" Data Flow Diagram .................................................................................................. 6

Figure 6: "Cancel" Data Flow Diagram .................................................................................................... 7

Figure 7: "Token Expired" Data Flow Diagram .......................................................................................... 7

Figure 8: "Handle Frame" Data Flow Diagram .......................................................................................... 8

Figure 9: Frame Data Flow Diagram ........................................................................................................ 9

Figure 10: Transport Data Flow ................................................................................................................ 9

Figure 11: Create a new Visual C# application ......................................................................................... 11

Figure 12: Add ZWave Dll and BasicApplication projects ..................................................................... 12

Figure 13 Add Reference to the libraries ............................................................................................... 12
1 ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CD</td>
<td>Compact Disc</td>
</tr>
<tr>
<td>COM</td>
<td>Serial port interface on IBM PC-compatible computers</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>PC</td>
<td>Personal computer</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus, a serial bus standard to interface devices</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>WPF</td>
<td>Windows Presentation Framework</td>
</tr>
</tbody>
</table>

2 INTRODUCTION

2.1 Purpose

This document describes the architecture and functionality of the Z-Wave DLL that supports Z-Wave enabled PC based applications.

This document includes conceptual overviews, step-by-step procedures, and information about samples.

More and up-to-date information on Z-Wave DLL API references and practical implementation can be found in Error! Reference source not found..

2.2 Audience and prerequisites

The audience is Z-Wave partners and Silicon Labs developers. It is assumed that partner developers already have and are familiar with the Z-Wave Developer's Kit.
3 WHAT IS Z-WAVE DLL

“Z-Wave DLL” is a name of the framework library for building Z-Wave PC Applications (WinForms, WPF or Console) for Microsoft .NET Framework platform.

Z-Wave PC Applications features:

- Can send/receive data packets to/from one or many Z-Wave devices
- Parse received data into data frames
- Store received data packets
- Synchronize send and receive using acknowledges
- Build actions/operations and synchronize them

List of supported Z-Wave Devices and Firmware:

- Zniffer firmware (“ZnifferApplication” API)
- Serial API firmware (“BasicApplication” API)
- Z/IP Gateway (“ZipApplication” API)
- USB, UART programmer interfaces firmware (“ProgrammerApplication” API)

Custom Z-Wave enabled .NET application, which is built on top of Z-Wave DLL, should be thin GUI-oriented WinForms, WPF or console application with reference to this DLL in the solution.

![Figure 1: Positioning of Z-Wave DLL within custom solution](image-url)
4 ARCHITECTURE AND IMPLEMENTATION

Z-Wave DLL consists of such layers:

- Transport (provides methods for send/receive data packets from communication source)
- Frame (provides methods for guaranteed send (using acknowledges) and parsing received data packets)
- Session (is a message broker, provides methods for synchronizing actions/operations, message validation, message transformation and message routing)
- Application (provides API specific methods, callbacks)

Each Z-Wave DLL layer is a factory for clients. Each connected Z-Wave device represented by its own Application, Session, Frame and Transport clients.

5 DATA FLOW

Data flow diagram is a graphical representation of the "flow" of data through Z-Wave DLL, modeling its process aspects. It shows how input data is transformed to output results through a sequence of functional transformations.
Figure 3: “Z-Wave DLL” Data Flow Diagram
5.1 Application Data Flow

After application startup Create Application Client flow executed. It creates set of clients for serving each attached Z-Wave Device.

![Application Data Flow Diagram]

Figure 4: Application Data Flow

When User select method for execute Action (or Operation) is created. Created Action passes to the session client which returns Action Token. If method is asynchronous Action Token returned to the User, otherwise Wait Completed blocks running thread until Action Completed and Action Result returned to the User. Asynchronous methods have completed callback also they can be cancelled by the User.

5.2 Session Data Flow

Data types:

- Action – ActionBase(OperationBase) base class for creating operations
- Action Handler Result – ActionHandlerResult class. Contains data packets to send or derived Actions to run
- Data Frame – DataFrame base class. Represents data packet to send/receive.
- Signal – signal for unblock waiting thread.

Components:

- Substitute Manager – implements ISubstituteManager interface. Can substitute action on another action (example SendDataOperation->SendDataSecureTask). Can substitute incoming DataFrame on another DataFrame (decrypt incoming DataFrame).
- Action Check Point – blocks running thread until previous exclusive Action completes. If Action is not exclusive it runs in parallel.
- Timeout Manager – maintains Action timeouts.
- Action Buffer Block – queue for derived Actions

Major flows:

- “Execute” flow
• “Cancel” flow
• “Token Expired” flow
• “Handle Frame” flow

After “Execute” flow starts Action may be substituted by Substitute Manager. For example in BasicApplication API Substitute Manager is Security Manager and it replaces AddNode, SendData, etc. with its secure analogues. In ZipApplication, ProgrammerApplication, ZnifferApplication Substitute Manager is not implemented.

Next is the Action Check Point. It blocks running thread until previous exclusive Action completes. If Action is not exclusive it runs in parallel. This prevents overlapping of the critical operations. For example: SendDataOperation in BasicApplication API.

After the Start method Action gives Action Handler Result containing Command Messages. Command Message contains Data Frame for sending and timeout. After processing Action Handler Result state of the Action checked and if it is completed Set Completed signal issued.

“Cancel” flow starts by user action. Try Handle Stopped returns corresponding Action for the given Action Token. Get Stop Unit returns Action Handler Result with Command Message for the last command to the Z-Wave Device (for example NodeStop message).
Figure 6: "Cancel" Data Flow Diagram

After sending data frame of the Command Message Set Completed signal issued, then Action Check Point allows next Action.

"Token Expired" flow like "Cancel" flow, but it is initiated by the Timeout Manager when corresponding Action Token timeout expires

Figure 7: "Token Expired" Data Flow Diagram

"Handle Frame" flow describes how incoming data frames change Actions flow. It starts after Frame Client executes Receive Frame Callback.
5.3 Frame Data Flow

Frame Data Flow has two major flows:

- "Send Frames" flow
- "Handle Data" flow
5.4 Transport Data Flow

Transport Data Flow has two major flows:

- **“Work thread” flow**
- **“Write Data” flow**

**Figure 10: Transport Data Flow**

“**Work thread**” flow checks Communication Source for any new data. It reads data when underlying source has new data. After that Receive Data callback executed and Data Chunk passes to the Frame Client (see “Handle Data” flow).

“**Write Data**” flow started when Frame Client executes Send Data Callback it writes binary data to the underlying Communication Source.
6 GETTING STARTED

6.1 Check the prerequisites

The following components should be installed on the computer that you need to develop, build, test, and deploy Z-Wave DLL based .NET application:

- Microsoft .NET Framework, version 3.5 or later
- The Microsoft Visual Studio 2008 development environment
- Windows Installer 3.0 (Windows Installer 3.1 or later is recommended).

**Important:** Make sure you have the latest service pack and critical updates for the version of Windows that you are running. To find the recent security updates, visit Windows Update.

6.2 Limitations

The setup has only been tested on Windows 7 and Windows 8, but may work on other versions as well.

6.3 Required Z-Wave hardware and libraries

Any Z-Wave enabled application requires the appropriate Z-Wave hardware to be connected:

- Program the Z-Wave device with appropriate HEX file, usually Serial API Controller, Installer, Bridge or Slave depending on the Windows application.
- Establish serial communication to the Z-Wave module.
- Z-Wave Dll 5.39 or newer library dll files or sources
- BasicApplication library (for serial API applications) dll file or sources
7 CREATION OF A Z-WAVE DLL BASED APPLICATION

This section describes the creation of the new Z-Wave DLL based .NET Windows Application and its customization.

The following steps are examined:

- Create a new Z-Wave DLL based .NET Windows Application project.
- Obtain available Serial Port Interfaces.
- Implement “Detect Device” functionality.

Attention! The steps below are based on a C# project. However you can use the same steps for a Visual Basic project, except for specifics such as file name extensions and code.

7.1 Create a .NET Windows Application project

In this section, you create a solution with .NET Windows Application project in Visual Studio.

2. In the File menu, point to New, and then click Project.
3. In the New Project dialog box, expand Visual C#, and then click Windows. Under Visual Studio installed templates, select Console Application. Select Create Directory for Solution. In the Name field, type MyApplication. And finally enter the location for the solution (D:\Internal\DK66000\SDK\PC\Source\SampleApplications for example).

![Figure 11: Create a new Visual C# application](image)
To continue, click OK.

7.2 Add references to Z-Wave DLL components

1. Use the Add Existing Project dialog to add libraries to your solution.
2. Browse to folder where Z-Wave DLL sources located, add 3 projects Utils, ZWave, ZWaveXml
3. Add BasicApplication project from the folder where BasicApplication source located

![](image1)

Figure 12: Add ZWave Dll and BasicApplication projects

4. Add References to the added Z-Wave Dll and BasicApplication libraries

![](image2)

Figure 13 Add Reference to the libraries

7.3 Example of the Inclusion node

```csharp
using ZWaveBasicApplication;
using ZWaveLayers.Session;
using ZWaveLayers.Transport;
using ZWaveBasicApplication.Devices;
using ZWaveLayers;
using ZWaveBasicApplication.Operations;
using ZWave.Enums;
namespace MyApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            BasicApplicationLayer AppLayer = new BasicApplicationLayer(
                new SessionLayer(),
                new BasicFrameLayer(),
                new SerialPortTransportLayer());
            Controller Ctrl = AppLayer.CreateController();
            if (Ctrl.Connect(new SerialPortDataSource("COM13", BaudRates.Rate_115200)) == CommunicationStatuses.Done)
            {
                VersionResult res = Ctrl.GetVersion();
                if (res
```
7.4 Example of the Inclusion with both nodes connected

```csharp
    class Program {
        static void Main(string[] args) {
            BasicApplicationLayer AppLayer = new BasicApplicationLayer(
                new SessionLayer(),
                new BasicFrameLayer(),
                new SerialPortTransportLayer());
            Controller Ctrl = AppLayer.CreateController();
            Slave Slave = AppLayer.CreateSlave();
            if (Ctrl.Connect(new SerialPortDataSource("COM13", BaudRates.Rate_115200)) == CommunicationStatuses.Done &
                Slave.Connect(new SerialPortDataSource("COM16", BaudRates.Rate_115200)) == CommunicationStatuses.Done)
            {
                VersionResult verResCtrl = Ctrl.GetVersion();
                VersionResult verResSlave = Slave.GetVersion();
                if (verResCtrl && verResSlave)
                {
                    // reset to default settings
                    Ctrl.SetDefault();
                    Slave.SetDefault();
                    // async method for start inclusion process with 10 seconds timeout
                    ActionToken addToken = Ctrl.AddNodeToNetwork(Modes.NodeAny, 10000, null);
                    // wait for learn ready callback
                    Ctrl.WaitNodeStatusSignal(NodeStatuses.LearnReady, 10000);
                    // async method for start learn mode process with 10 seconds timeout
                    ActionToken learnToken = Slave.SetLearnMode(LearnNodes.LearnModeClassic, 10000, null);
                }
            }
        }
    }
}
```
// wait for AddNode completed
AddRemoveNodeResult addRes = (AddRemoveNodeResult)addToken.WaitCompletedSignal();
// wait for LearnMode completed
SetLearnModeResult learnRes = (SetLearnModeResult)learnToken.WaitCompletedSignal();
if (addRes && learnRes)
{
    // node added
    // addRes.Id - added node Id
    else
    {
        // inclusion failed
    }
} else
{
    // method failed check that device programmed with
    // SerialAPI firmware (ControllerStatic and SlaveEnhanced for example)
    Ctrl.Disconnect();
    Slave.Disconnect();
} else
{
    // connect failed. Port busy or doesn't exists
    Ctrl.Dispose();
    Slave.Dispose();
}
8 REFERENCES
Disclaimer
Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information
Silicon Laboratories Inc., Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32G, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world’s most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOModem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, Z-Wave and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.