European Commission

Information Society and Media Directorate – General

Protocol SW of a SmartCoDe node, initial version

SmartCoDe

Project No.:	ICT-2009-247473
Deliverable No.:	D-2.4
Deliverable Title:	Protocol SW of a SmartCoDe node, initial version
Due Date:	June 30th 2011

Nature:	Report
Dissemination Level:	public
Authors:	Juraj Hájek, Juraj Špánik, Edgar Holleis, Markus Damm
Lead Beneficiary No.:	6
Lead Beneficiary:	ADO

SmartCoDe has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013]) under grant agreement n°247473

Table of Content

1 Introduction

1.1 Purpose

The purpose of this document is to describe the implementation of the SmartCoDe protocol software as a result of D-2.4 - Protocol SW of a SmartCoDe node, initial version.

The initial protocol software builds on top of knowledge gained on several tasks in work package 1 and 2:

- Task 1.3 Energy Generation Forecasting
- Task 1.4 Demand side management in local grids combining regenerative energies and household/office appliances
- Task 1.5 Automatic power management
- Task 2.2 Optimization and adaptation of the ZigBee standard.doc
- Task 2.3 System design of a SmartCoDe node

The knowledge and information gained through the work described here is a basis for several ongoing and upcoming endeavours in the SmartCoDe project mainly:

- D-2.4, M30: Architecture and protocol SW of a SmartCoDe node, revised

In Section 2, we present design goals and scope of the initial and final version of the protocol software. Section 3 describes ZigBee extensibility model used for definition of SmartCoDe protocol stack, high level architecture and each network layer in optimized profile. Sections 4 and 5 contain technical details about development tools and documentation. Section 5 presents conclusion and future work.

1.1 Requirement Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document, when in a requirements block (as indicated by the leading Req[]), are to be interpreted as described in [RFC 2119].

1.2 Typography Conventions Used

Example protocol requests, protocol responses, and attributes are presented in `fixed-width Courier New font, with a size of 10.`

2 About the SmartCoDe Profile Protocol Specification

2.1 Design Goals

The following considerations have shaped the final design of the SmartCoDe protocol:

- Offer straightforward support for the energy management approach outlined in the WP 1 deliverables
- Leverage existing technology (ZigBee, ZigBee Home Automation and ZigBee Smart Energy) as much as practical
- Provide clear interfaces and boundaries between the realms of ZigBee Home Automation, ZigBee Smart Energy and SmartCoDe

The last point is a systematic difference between the SmartCoDe project and what was achieved in eDIANA project.[1]

The eDIANA protocol was defined by mapping previously defined use-cases onto ZigBee, embracing existing ZigBee features that can support those use cases and defining a manufacturer specific ZigBee cluster for mapping the remaining use-cases. eDIANA use-case analysis remains valid for SmartCoDe and therefore did not have to be repeated at that level of detail. The approaches differ in two important aspects:

- The energy management approach described in the WP1 deliverables is fundamentally different in that SmartCoDe follows a partially distributed energy management approach where control is exercised at the periphery of the network and the EMU is restricted to coordination.
- Instead of embracing ZigBee Home Automation and ZigBee Smart Energy, the SmartCoDe profile complements those protocols: Use ZigBee HA if you want to switch on and off, use SmartCoDe if you want to coordinate energy usage.

The consequence of that can be observed in the verbs used by the respective protocols. E. g.:

- eDIANA: “Raise fridge / freezer temperature temporarily” (EMU exercising direct control)
- SmartCoDe: “Send cost function”, “Send power estimate” (EMU and EuP participating in a distributed algorithm)

By analogy with modern web-service protocols, the SmartCoDe Profile follows a RESTful architecture [REST]. Several specific design principles are important for the application protocol:

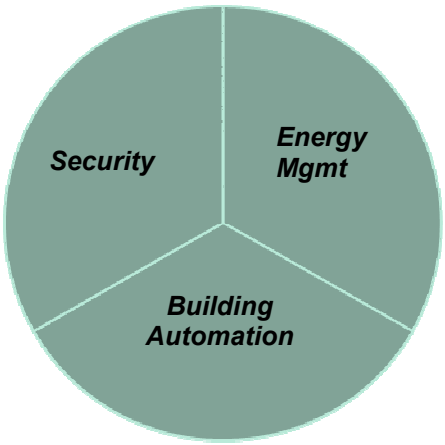
- While devices may maintain state, interfaces should be stateless
- Keep implementations simple, minimize code space and memory requirements
- Minimize the number of transactions required to achieve a given function.

The protocol was defined by performing the following steps:

1. Define the services required to implement the SmartCoDe energy saving algorithm based on the EuP and LED classification from Deliverable D-1.1.1
2. Define clusters to support the required functionality of the services
3. Break down each cluster into commands and attributes
4. Assign command IDs and frame format for commands

[1] http://www.artemis-ediana.eu/

2.2 Scope of the Initial Version



The SmartCoDe Profile is comprised of three innovative parts:

- Energy Management – innovative decentralized approach
- Building Automation – innovative commissioning modes for large installations
- Security – state of the art, smart-card level security scaled down into SoC package

The initial version focuses on energy management aspects. This is because of timing, especially interdependence with respect to WP4, the demonstrator. While security and commissioning modes can be demonstrated at any later stage of the project, demonstrating energy savings takes time to achieve the required statistical significance.

The presented initial version of the SmartCoDe profile is co-developped on the SmartCoDe simulation environment. The immediate goals are:

- validate completeness and compliance of protocol software regarding to the requirements of effective energy management
- provide an executable specification as an communication tool which allows rapid prototyping as a way how to express new ideas
- allow to validate high-level integration of different system components in the simulation environment (communication protocol, energy optimization algorithms).

The following aspects **are not** within scope of the initial version:

- porting of software into target platform
- optimization of algorithms on implementation level (protocol SW is optimized on design level, optimization of elementary algorithms is closely related to porting to HW platform)
- implementation of cryptographic primitives, because they will be provided by HW platform

3 Optimization and adaptation ZigBee standard

The SmartCoDe protocol is an extension and modification of the ZigBee standard. ZigBee specification is followed wherever possible until differences are not defined explicitly. The main advantages of the ZigBee solution over fully proprietary solutions are:

- interoperability and vendor independence
- easy adaptation or retrofitting of existing products
- reusability of concept and availability of existing stack which allows to focus on new energy management algorithms instead of reinventing underlying framework

The interoperability of different vendors is achieved through application profiles. Profiles are currently available or in development for the following types of applications:

- Smart Energy
- Home Automation
- Telecom Applications
- Commercial Building Automation
- Personal Home and Hospital Care

Non proprietary profiles are defined by the ZigBee Alliance and provide a description of the devices supported for a specific application together with the messaging scheme used by those devices for communication.

Each *device* defined by profile is implemented as application object and application object is connected to the rest of the ZigBee stack by an *endpoint*, which is an addressable component within a device. Messaging scheme is defined by *communication clusters.* Communication cluster specifies features of the network on the high level and concrete commands and parameters on the low level. For example, the Home Automation profile contains a cluster dedicated to the control of lighting subsystems including simple on/off or luminance levelling.

Each cluster has two ends:

- The client/output requests and manipulates the data.
- The server/input has the source data.

In the case of SmartCoDe project, optimization and adaptation of ZigBee standard consists of:

- definition of the new application profile which extends existing profiles and adds new features to network
- definition of services; SmartCoDe is focused more on specification of services provided by devices rather than specification of devices
- definition of clusters (commands and parameters) required to implement services

Detailed description about optimization and adaptation is provided in sections bellow.

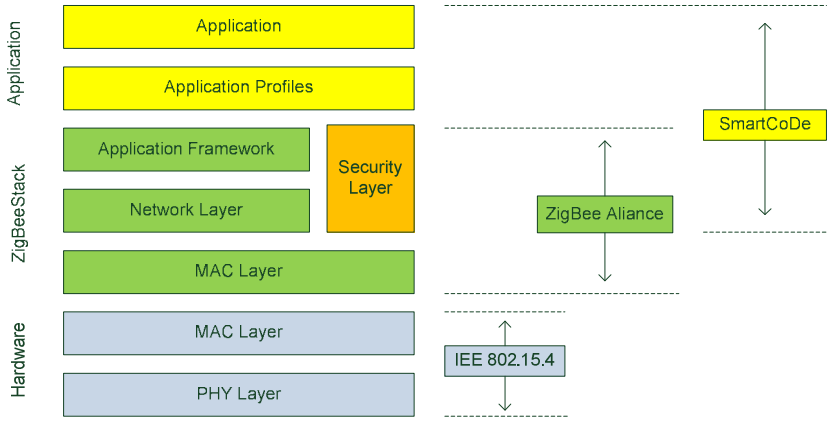


Figure 1: SmartCoDe profile in context of ZigBee stack

3.1 Physical and Media Access Control Layer

The SmartCoDe protocol **is PHY/MAC agnostic.** Wireless transmission is not mandatory with respect to the energy saving algorithms.

For the wireless implementation, the protocol builds upon the physical layer and medium access control defined in IEEE 802.15.4-2003. The same standard is referenced in ZigBee, with the exception of the ZigBee Smart Energy 2.0 profile (still available only as a draft), which will be also PHY/MAC agnostic.

ZigBee PHY/MAC layers already address requirements for reliability and noninterference of network motioned in DoW. ZigBee products have access to 16 separate, 5MHz channels in the 2.4GHz band. Several of these do not overlap with US and European versions of Wi-Fi™. ZigBee incorporates an IEEE 802.15.4 defined CSMA-CA protocol that reduces the probability of interfering with other users, plus ZigBee uses automatic retransmission of data to ensure network robustness.[2]

No other optimization or adaptation was required.

3.2 Network Layer

ZigBee uses Ad hoc On-Demand Distance Vector (AODV) routing protocol designed for mobile ad hoc networks and other wireless ad-hoc networks. ZigBee's addressing scheme is capable of supporting more than 64,000 nodes per network, depends on which frequency band is selected, how often each device on the network needs to communicate, and how much data loss or retransmissions can be tolerated by the application.[3]

Routing and network formation capabilities of ZigBee network layer are sufficient to achieve the project goals and no other optimization or adaptation was required.

3.3 Application Layer

The core of SmartCoDe functionality is defined at the application layer. The SmartCoDe defines a custom ZigBee profile, in accordance with the standard ZigBee extension model.

Within the SmartCoDe profile it is possible to reuse existing clusters from other profiles, especially:

- Home Automation
- Smart Energy
- Smart Energy 2.0 (draft)

The SmartCoDe profile introduces new types of devices based on services identified in deliverable “D1.1.1 Model of local energy resource cluster”. The whole profile is service oriented. Each service is described as a set of communication clusters required to achieve its functionality. Clusters contain a set of attributes that represent device state together with commands that enable communication between application objects. Each cluster is identified with a unique ID.

[2] http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx#7
[3] http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx#9

For example, the Variable Service (VARSVC) contains a cluster which allows configuring demand for energy consumption with appropriate criticality levels. Required commands and attributes are described in low level form of data packets and structures as described below.

The application protocol is designed to support more fine grained approach than ZigBee SE (1.0) and new types of services allow implementation of new methods for saving energy not directly supported by existing ZigBee profiles. For example EuPs of the Virtually Storable Service class allows to accumulate energy in thermal form for later use. The application protocol and the SmartCoDe application responsible for saving energy are considered to be as much independent as possible to allow for further optimization without the need to change the protocol stack.

Interoperability with existing ZigBee profiles is possible to the level compliant with high grade security requirements defined in the DoW. If functionality of any existing profile is required (e.g. regulation of lightening from ZigBee HA), it is possible to reuse the whole protocol stack except security services, which are provided by a separate SmartCoDe sub-layer.

Interoperability with already existing devices is also possible without their customization. In this case, a SmartCoDe device should implement both profiles (SmartCoDe and standard ZigBee) and operate as a bridge between them. Bridging process can include:

a) functional translation, e.g. commands from the first profile are translated into commands of the second profile (complex energy management commands can be potentially translated into simple switch on/off commands)

b) non-functional translation, e.g. two different parts of network can have different security requirements (existing ZigBee HA lightening systems does not support strong authentication mechanisms based on public key cryptography)

Of course, communication in the non-SmartCoDe part of network is secured in accordance with the respective ZigBee standard.

Implementation is straightforward and based on D1.1.1 “Model of local energy resource cluster“. Each service is implemented by exactly one C++ class. Class hierarchy and naming convention strictly follows terminology defined in D1.1.1. For example, there are 2 types of services - for energy using products and local energy providers. Particular services are then derived from these base classes.

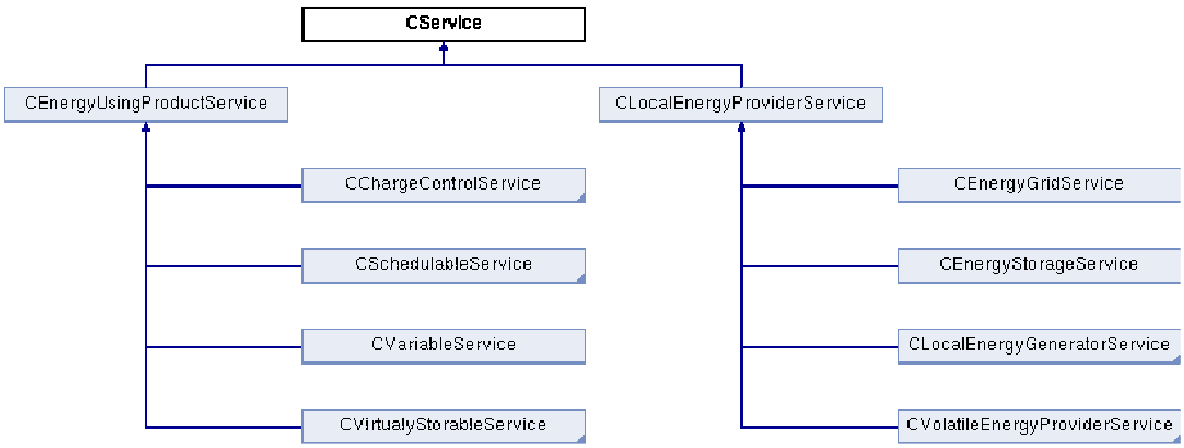


Figure 2: Inheritance diagram for CService

Each `CService` class representing network service delegates communication tasks to cluster objects (`CCluster`) implementing ZigBee clusters described above.

The complete description of all methods and attributes of implemented in class hierarchy is out of scope of this document and can be found in SmartCoDe Programming Reference Manual referenced in the section 5. Instead of exhausting low level description, we will describe implementation of one chosen service.

Example: Implementation of VirtuallyStorableService (VSTS)

Virtually Storable Service is implemented by `CVirtuallyStorableService` class. It supports two communication clusters:

- Message cluster (optional)
- Virtual storage management cluster (mandatory)

Message cluster is standard ZigBee cluster which can be used for delivering textual notification about device status and can be shown e.g. on LCD display of local management unit.

Virtual storage management cluster is a custom cluster which hides implementation details required for storing different forms of energy like heat or water mass. The primary goal of cluster is to provide simplified interface (façade) allowing to setup expected parameters of virtual storage and SmartCoDe will optimize energy consumption to meet defined criteria. It is assumed that device implementing *storage management cluster* reuses existing ZigBee metering and controlling clusters. Cluster is implemented by `CVirtualStorageManagementCluster` as described bellow.

SmartCoDe application profile references this custom cluster and defines new cluster ID in reserved ZigBee area:

Functional Domain	Cluster Name	Cluster ID
General	Basic	0x0000
General	Identify	0x0003
General	Time	0x000A
Smart Energy	Simple Metering	0x0702
Smart Energy	***Message***	***0x0703***
EuP	Schedule	0xFC00
EuP	Energy Profile	0xFC01
EuP	***Virtual Storage Management***	***0xFC02***
EuP	Load Control Vector	0xFC03
LEP	Power Generator	0xFC04
LEP	Battery	0xFC05

The assignment of custom IDs is based on the following table (ZigBee Cluster List):

Cluster Identifier	Description
0x0000 – 0x7fff	Standard ZigBee cluster.
0x8000 – 0xfbff	Reserved.
0xfc00 – 0xffff	***Manufacturer specific cluster within a standard ZigBee***

Virtual Storage Management cluster defines new ZigBee attributes:

Attribute Set Identifier	Description
0x000	Storage Type
0x001	Storage State
0x002	Expected State

with the following new types:

Storage Type Attribute Value	Description
0x000	Water Tank
0x001	Thermal Storage

and extended attribute sets.

Storage State Attribute Set:

Identifier	Name	Type	Access	Default	Mandatory/Optional
0x0010	IsRemovable	Bool	Read	FALSE	M
0x0011	StateOfCharge	Int8	Read	0	O
0x0012	UnderlyingMeter	DeviceId	Read	-	M

IsRemovable**:** Storage can be stable part of infrastructure (e..g. built-in water tank) or can be removed in any time (e.g. replaceable battery).

StateOfCharge**:** Charging **s**tate in percent. Charging status is available for water tank or battery, but not for thermal storage like room.

UnderlyingMeter**:** Metering device which can provide more detailed information about storage. Target metering device should support following clusters:

- Water Tank – Simple Metering Cluster (m3) (ZigBee SE)
- Thermal Storage – Temperature Measurement Cluster (ZigBee HA)

Expected State Attribute Set:

Identifier	Name	Type	Access	Default	Mandatory/Optional
0x0020	IsFullChargingExpected	Bool	Read/Write	FALSE	M
0x0021	IsUnoccupiedSetpointSupported	Bool	Read/Write	FALSE	O
0x0023	MinValue	Int16	Read/Write	-	O
0x0024	MaxValue	Int16	Read/Write	-	O
0x0025	PreferedValue	Int16	Read/Write	-	O
0x0026	MinValueUnoccupied	Int16	Read/Write	-	O
0x0027	MaxValueUnoccupied	Int16	Read/Write	-	O
0x0028	PreferedValueUnoccupied	Int16	Read/Write	-	O

IsFullChargingExpected**:** If set to true SmartCoDe will charge storage to its max. capacity and prevent loses of energy from storage. If *IsFullChargingExpected* is set to TRUE other attributes should not be filled

IsUnoccupiedSetpointSupported: Storage supports different behaviour based on building/room occupancy.

MinValue**:** Lower boundary which is acceptable for storage management. Interpretation is based on storage type (e.g. see *Thermostat Settings Attribute Set* from [ZCL] how to calculate value).

MaxValue**:** Upper boundary which is acceptable for storage management.

PreferedValue: Prefered value (prefered value is not always average between lower and upper boundary)/

MinValueUnocuppied, MaxValueUnocuppied, PreferedValueOccupied: The system allows to define specific values for unoccupied rooms/building. Example: unoccupied house has lower default temperature and potential water tank for showering does not have to hold higher reserve.

Implementation in C++ then directly implements programming primitives like structures, classes and enumerations to transform specification into executable code. In simulation environment, each ZigBee commands is represented by simple method and packet payload is represented by method parameters, for example:

```
enum SVirtualStorageType
{
      WaterTank,
      ThermalStorage
};
struct SVirtualStorageExpectedState  {
      bool            IsFullChargingExpected;
      bool            IsUnoccupiedSetpointSupported;
      uint16_t        MinValue;
      uint16_t        MaxValue;
      uint16_t        PreferedValue;
      uint16_t        MinValueUnoccupied;
      uint16_t        MaxValueUnoccupied;
      uint16_t        PreferedValueUnoccupied;
};

class CVirtualStorageManagementCluster: public CCluster
{
  private:
      ...
      SVirtualStorageExpectedState    expectedState;

  public:
      SVirtualStorageExpectedState GetExpectedState();
      void SetExpectedState(SVirtualStorageExpectedState expectedState);
};
```

Definition and implementation of all other services and communication clusters is analogical.

3.4 Security

The security layer is based on proposed standard „A Cryptographic Suite for Embedded Systems (SuiteE)" developed by The European Telecommunications Standards Institute (ETSI) [SUITE1] where integrity, confidentiality and authentication are addressed.

The Cryptographic Suite for Embedded Systems (Suite E) aims to optimally meet the wide variety of cryptographic requirements, by providing a compact and complete collection of cryptographic algorithms having minimal code space, computational requirements and bandwidth usage. Additionally, the selection of these algorithms are tuned to minimize overall system costs in mass production by selecting easily embeddable algorithms which will further reduce code space, energy usage and increase computational performance.[4]

The main building blocks for security layer are:

- Encryption – AES-128 (CTR, CBC-MAC and AES-CCM modes are supported)
- Deterministic random number generator - reduced set of options to the CTR_DRBG definition using AES as defined in [NIST-800-90].
- Hash: AES-MMO [MOV96].
- Elliptic Curve Signatures with Partial Message Recovery – Elliptic Curve Pintsov-Vanstone Signatures (ECPVS) defined in [X9.92.1] and [IEEE1363-A]

[4] http://tools.ietf.org/html/draft-campagna-suitee-01#ref-ISO-10118-2

- Elliptic Curve Implicit Certificates (ECQV) defined in [SEC4].
- Elliptic Curve Key Agreement Scheme – ECMQV [SEC1]

Changes and improvements compared to DoW

Proposed ECC based algorithms and schemas are more feasible for small wireless devices and guarantee higher security level at higher speed and lower energy consumption than RSA based authentication mentioned in DoW (p. 25).

The savings in communication costs are caused mainly by significantly shorter key sizes. The comparison of the energy consumption and speed of handshake protocol based on RSA and ECC can be found for example in [ENES]. Recommended key sizes are defined in [NIST800-57].

Algorithm security lifetimes	Symmetric key algorithms (Encryption & MAC)	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
Through 2010 (min. of 80 bits of strength)	2TDEA[23] 3TDEA AES-128 AES-192 AES-256	Min.: L = 1024; N =160	Min.: k=1024	Min.: f=160
Through 2030 (min. of 112 bits of strength)	3TDEA AES-128 AES-192 AES-256	Min.: L = 2048 N = 224	Min.: k=2048	Min.: f=224
Beyond 2030 (min. of 128 bits of strength)	AES-128 AES-192 AES-256	Min.: L = 3072 N = 256	Min.: k=3072	Min.: f=256

Table 1: Recommended algorithms and minimum key sizes

Impact on other tasks and work packages

The protocol design and selection of algorithms is closely related to WP3 (ultra low cost implementation) where implementation of required ECC smartcard modules is evaluated by Infineon and Ardaco.

Even if SmartCoDe protocol refers to particular cryptographic primitives they will not be part of software implementation because hardware implementation is more effective in all terms mentioned above. SmartCoDe stack currently defines only required interfaces and final protocol software will use security features of target platform. Preliminary design assumes usage of 16-bit security controller from Infineon SLE 77 family.

It is important to mention that usage of smart cards or trusted platform modules does not strengthen network security, but physical security and efficiency of overall system is increased. The smart card is a potentially replaceable part of SmartCoDe node, but it is assumed that replacement of already deployed smart cards will be considered only in sectors with specific requirements, procedures and regulations related to IT security (e.g. banking). Because of this, the overall security model is designed to be sufficient for the coming decades.[5]

The development of beyond-state-of art commissioning methods happens independently and is not affecting the implementation of energy saving algorithms.

[5] It is practically impossible to predict safety of algorithms in more than 20years.

4 Development tools

The initial protocol software is implemented in C++ and SystemC library. This allows integration with energy management algorithms developed in the same environment and common evaluation in the existing simulation framework before porting code into target hardware platform.

Type	Tool
Programing language	C++
Simulation library	SystemC
IDE	Eclipse
Documentation	Doxygen

Table 2: Development tools

5 Documentation

The results of all work related to protocol software implementation is documented in:

- API of protocol software is documented in SmartCoDe Programming Reference Manual

Low level specification of messaging schema (packet formats, commands, messages) is documented in separate document „T2.2 - Optimization and adaptation of the ZigBee standard.doc". This document was input for development of API mentioned above.

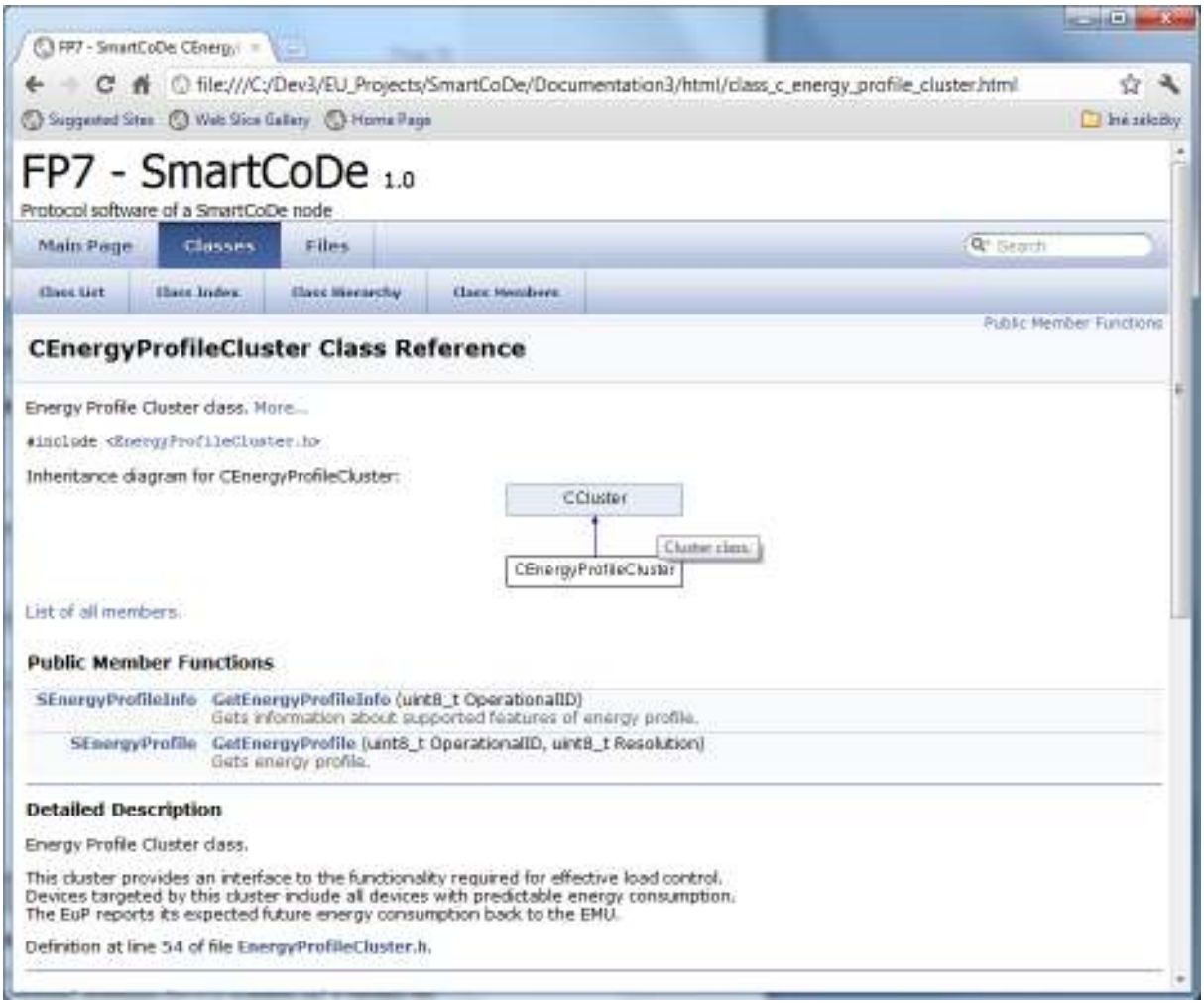


Figure 3: Example of SmartCoDe protocol reference manual

6 Conclusion and future work

The first version of protocol level software is developed on the functional and architecture level model that acts as virtual prototype, as long as no hardware prototype (WP3) is available yet. It handles the communication layers above PHY, MAC and NWK layer.

Created C++ model of protocol stack is functional and allows further experiments with different approaches for energy management. Object oriented approach keeps code consistent with mental model and allows rapid prototyping and expression of the new ideas directly in the form of executable specification

Next step in development is integration with energy management algorithms and validation in simulation environment. Prototyping of energy management algorithms can potentially raise additional requirements for format of exchanged messages without significant impact on whole design. After integration and validation, all software modules they will be ported to target platform. For final version protocol software will be rewritten to low level C language and optimized.

7 Abbreviations

AES	Advanced Encryption Standard
CA	Certification Authority
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECMQV	Elliptic Curve Menezes-Qu-Vanstone
NIST	National Institute of Standards and Technology
PKI	Public Key Infrastructure
ZCL	ZigBee Cluster Library
ZDP	ZigBee Device Profile

8 Bibliography

[IEEE1363] Institute of Electrical and Electronics Engineers, "Standard Specifications for Public Key Cryptography", IEEE 1363, 2000.

[ENES] Sun Microsystems Laboratories, "Analysis of Public-Key Cryptography on Small Wireless Devices", http://users.soe.ucsc.edu/~awander/stuff/EnergyPaper.pdf .

[SEC1] Standards for Efficient Cryptography Group, "SEC1: Elliptic Curve Cryptography", SEC 1, May 2009, http://www.secg.org/download/aid-780/sec1-v2.pdf

[SEC2] Standards for Efficient Cryptography Group, "SEC2: Recommended Elliptic Curve Domain Parameters", SEC 2, September 2010, http://www.secg.org/download/aid-784/sec2-v2.pdf.

[SEC4] Standards for Efficient Cryptography Group, "Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), v0.97", SEC 4, March 2011, http://www.secg.org/download/aid-785/sec4-0.97.pdf.

[SUITE1] A Cryptographic Suite for Embedded Systems (SuiteE), ETSI, October 2011, http://tools.ietf.org/html/draft-campagna-suitee-01.

[NITS800-57] Recommendation for Key Management, Special Publication 800-57 Part 1, NIST, 03/2007.

[NIST-800-90] National Institute of Standards and Technology, "NIST SP 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators(Revised)", NIST Special Publication 800-90, March 2007, http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf.

[MOV96] Menezes, A., van Oorschot, P., and S. Vanstone, "Handbook of Applied Cryptography", 1996, http://www.cacr.math.uwaterloo.ca/hac.

[ZigBee] ZigBee Standards Organization, "ZigBee Specification, revision 17", October 2007, http://www.zigbee.org/ZigBeeSpecificationDownloadRequest/tabid/311/Default.aspx/.

[ZigBeeSE] ZigBee Standards Organization, "ZigBee Smart Energy Profile Specification, revision 15", December 2008, http://www.zigbee.org/ZigBeeSmartEnergyPublicApplicationProfile/tabid/312/Default.aspx.