FLEXIBLEPOWER
ALLIANCE NETWORK

High level Functional Specification of the FlexiblePower Application Infrastructure

Version 0.4
January 2013

Summary

This document provides a high level functional description of the FlexiblePower Application Infrastructure. This application infrastructure creates interoperability between various SDM approaches on the one hand and different appliances and their protocols on the other hand.

Contents

1 Introduction

Smart grids are used to better incorporate renewable energy sources. These renewables often have an intermittent character and cannot be planned in advance. Therefore a smart grid has some way of dynamically matching supply and demand (Supply and Demand Matching; SDM).

Over the years a lot of different SDM approaches have been developed. Unfortunately these SDM approaches are not interoperable. A similar issue can be identified on the appliance level. Appliances provide the flexibility that is being exploited by SDM. To begin with there a lot of different appliances (washing machines, Combined Heat Power Systems, PV panels, fridges, etc.). They also use different protocols for communication (Zigbee, Z-wave, WiFi, PLC, etc.).

All this variety on both the SDM and appliance level presents Energy Management Systems (EMS) with a big challenge. This challenge is depicted in Figure 1.

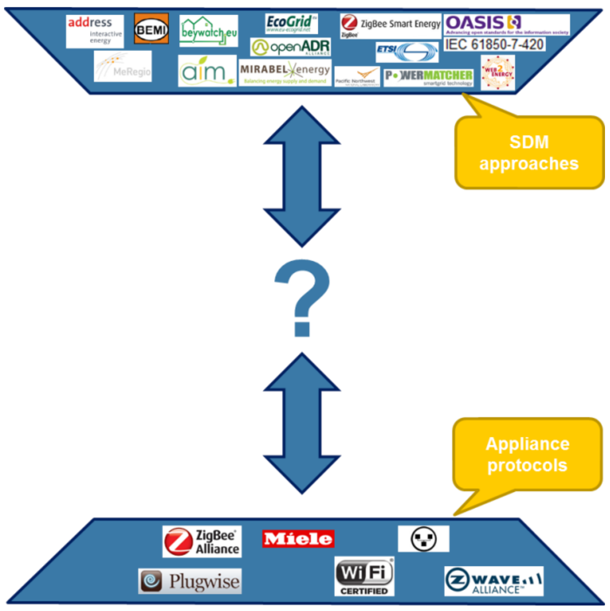


Figure 1: Interoperabilty challenge for energy management

Nowadays most EMS'es are tightly coupled to a particalar SDM approach. This results in a lock-in for consumers. A switch to another SDM approach/service almost always requires the installation of another EMS (hardware box).

The FlexiblePower Application Infrastructure (FPAI), which is the subject of this document, aims to create an interoperable platform that is able to connect to a variety of appliances and support a host of SDM approaches. This way the EMS hardware does not need to be changed when a consumers switches from one service to another. At the same time the FlexiblePower Application Infrastructure makes it easier for service providers to introduce new services, since they do not have to provide the EMS hardware to their consumers to go with it.

This document provides a high level description of the FlexiblePower Application Infrastructure. A detailed functional design document is also available.

2 Scope

The FlexiblePower Application Infrastructure can be used to flexibly support different Supply and Demand Management approaches towards end-customers.

With the FlexiblePower Application Infrastructure framework the household is managed via an FP Home Box. This box will be responsible for the negotiation between the energy service providers and the household and the coordination and management of energy resources located within the household.

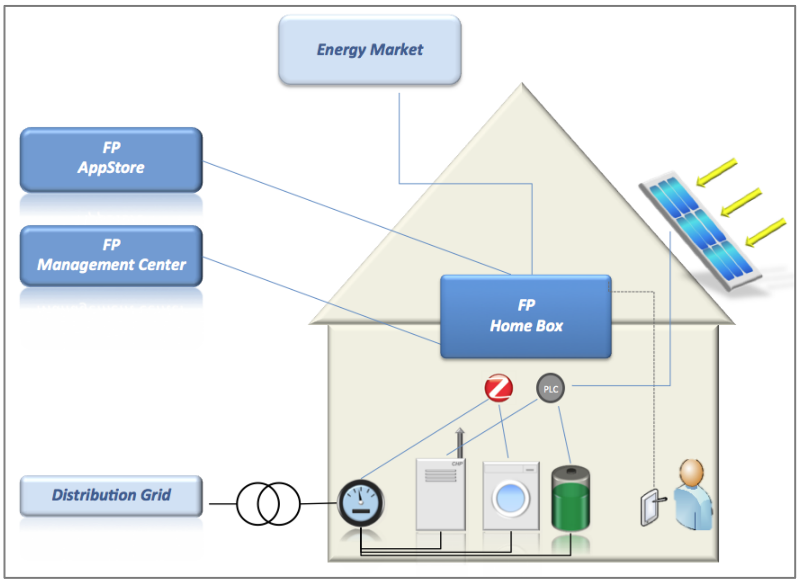


Figure 2 Scope of the FlexiblePower Application Infrastructure

In Figure 2the FlexiblePower Application Infrastructure is depicted with an example household, which also could be a more complex environment of a company with multiple sites.

The FlexiblePower Application Infrastructure context contains the following functional components:

- **Prosumer or Household.** The party who actually consumes and/or locally produces energy is called the Prosumer or Household.
- **Distribution Grid.** Via the Distribution Network the energy is provided to households (and businesses). Through the usage of electrification (EV, heat pumps) and a growth in power generation (cogeneration, solar panels) there is a higher risk of overload and an a reduced quality of energy delivery. To cope with this the DNO (the Distribution Network Operator) will need to continuously monitor the infrastructure. Based on measured load signals will be given towards households to adapt their behavior on current status of the grid (possibly but not necessarily directly via the FP Home Box). The Distribution Net will thus become an active component and FlexiblePower Application Infrastructure can play a role in this.

- **Distribution Network Operator (DNO).** The DNO is the owner of the Distribution Network, responsible for the quality and stability of the network. To manage the network the DNO will need measurements and predictions on the energy consumption and generation so it can manage the network and possibly directly control the actual generation and consumption of energy. The DNO could potentially use the FlexiblePower Application Infrastructure to monitor and manage energy (by installing a DNO App on the Home Box and interface with the App via the Market Operator).
- **Appliances/Resources.** The resources within a household that can provide flexibility with regard to: consumption, storage and production of energy. Examples of resource that can be controlled by the FlexiblePower Application Infrastructure are:
  - **Smart meter.** Households will be provided with a smart meter which can be accessed by the PowerMatcher Node.
  - **Cogeneration.** There are several types of Cogeneration devices. Examples are fuel cells and cogeneration devices with a Sterling engine such as a CHP.
  - **Household equipment.** This are devices for which start moment can possibly be shifted (e.g. washing machine or loading of an automatic vacuum cleaner).
  - **Energy storage.** This can be different kinds of storage devices like self-discharging batteries (e.g. Li-in/NiMH batteries), chemical storage batteries which conversion loss (e.g. flow batteries) and mobile storage in electrical vehicles.
  - **Solar panels.** Generation of energy via Solar Panels cannot be controlled. A household need to adapt to the moments energy is actually being produced by the panels.
  - Possibly some other energy consuming or generating household resources.
- **Home Box.** The home box is used to denote the hardware on which the FlexiblePower Runtime runs. The runtime consists of all the consumer-side software. The main functions are: the management of appliances, running energy apps and informing the end-user via a graphical user interface.
- **Energy Market.** Most energy apps will need to communicate with the outside world to perform SDM. Take the PowerMatcher as an example, all agents will run locally on a consumer level while the auctioneer is a central component. Such central components are referred to as the Energy Market. Another example would be a central server that publishes dynamic price tariffs.
- **Management Center.** To centrally manage all the home box an Management Center will be needed which will securely manage the operation and software of the connected home boxes. Possible errors in a home box will be handled remotely by the Management Centre as much as possible. The management center will have to be run by an independent party, this might be a role for The DNO.
- **AppStore.** Third parties will able to create energy apps for the Home Box and provide these modules via the FlexiblePower App Store. End users can visit the app store to see which energy apps are available and install those apps directly on their home boxes.

In this high-level specification the functionality of the FlexiblePower Runtime, the Management Center and the App Store is described.

3 FPAI Architecture

This chapter describes the high level functional architecture of the Flexible Power Application Infrastructure. It starts with the Runtime, followed by the management center and the app store.

3.1 FlexiblePower Runtime

As mentioned in the previous chapter the runtime is the software that runs on top of the home box. This section will take a closer look at how the runtime is built up. It contains the following elements:

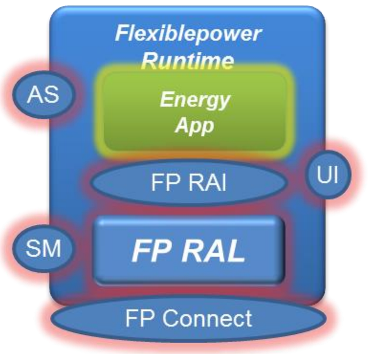


Figure 3: Overview of the FP runtime

- **Energy App.** The energy app is typically provided by a third party and provides energy management logic. An example would be the PowerMatcher app.
- **Resource Abstraction Interface (RAI).** This is an interface layer between the RAL and the energy apps.
- **Resource Abstraction Layer (RAL).** This layer monitors and controls the appliances and knows how much flexibility they can offer.
- **Connect.** This element acts as a gateway for communication with appliances.
- **SM.** This is the interface of the runtime towards the management center.
- **AS.** This is the interface of the runtime towards the app store.
- **UI.** This is the User Interface through which the end-user interacts with the runtime.

3.1.1 Energy Apps

An energy app exploits the flexibility that appliances have to offer. Often this will involve interaction with a central server such as with the SDM approaches depicted in Figure 1.

These apps are relatively easy to develop by a third party. There are two main requirements that an energy app will have to meet. Firstly, the app has to understand the Resource Abstraction Interface (RAI). This interface hides a lot of appliance details for the developers, which speeds up the development process. The RAI will be explained in more detail in the next section. The second requirement that an app has to comply with is that it should run on top of the OSGI framework (see chapter 4.2 for more information). OSGI is a framework on top of the Java programming language that enables the development of modular software and is widely known and used.

The energy apps are deployed via the app store and managed by the management center. This also takes away a lot of the operational responsibilities for a third party.

3.1.2 *Resource Abstraction Interface (RAI)*

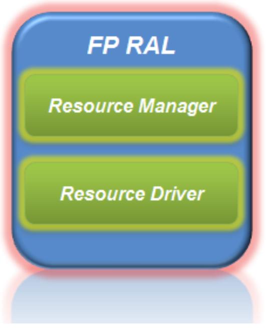
The RAI is pivotal for the FlexiblePower runtime. An energy app is only interested in exploiting energetic flexibility and not in the specifics of a washing machine for instance. The energetic flexibility is expressed in so called ‘Control Spaces’. There are four different types of ‘Control Spaces’ that cover most appliances:

- **Time shifters.** These Resources can shift the generation or usage of energy over a specific period of time. Examples are washing machines with a possibility to postpone the start time. Parameters in the ‘Control Space’ of a Time shifter are: an energy profile and a period over which the start moment can be shifted. Another example is an electric car which needs to be charged before a certain moment (before it will be used) but the actual loading can be performed and shifted within a certain time period.
- **Buffers.** This Resource can temporarily consume (or produce) more energy so they will use (or generate) less energy at a later moment in time. In most cases this are thermic buffers such as heating devices or refrigerators. Examples of parameters for the Buffer ‘Control Space’ are: total buffer capacity, filling, loading curve and discharge curve.
- **Storage.** In many aspects a Storage Resource resembles a Buffer Resource but a Storage resource can both store and return energy. Apart from parameters describing the Buffer resource, a Storage ‘Control Space’ also has parameters for storage loss.
- **Uncontrolled load/generation.** The energy behaviour of this type of Resources cannot be controlled (e.g. PV panels, TV, computers, etc.). For this resources only a prediction can be made for the expected consumption or production of energy. These predictions can be used in the rest of the framework to make decisions on energy control.

An energy app receives the control spaces and decides how to exploit the energetic flexibility. As a response to a control space an energy app will send an allocation. The allocation simply contains the energy profile that a resource will have to follow. An allocation should always respect the constraints that were expressed in the control space.

3.1.3 *Resource Abstraction Layer (RAL)*

The RAL monitors and controls the appliances in a household. This layer consists of two main components to do so: the resource driver and the resource manager. Where the RAI and the energy apps use generalized abstractions of appliances, these components are geared towards a specific appliance. Each appliance will typically have their own instance of a resource driver and manager.



The function of a resource driver is to expose the functions of the appliance. It can provide information about the current state of an appliance such as the current temperature or energy consumption. It also enables remote control of the functions of an appliance, think of a washing machine for instance where a certain program can be started at a particular time.

The job of the resource manager is to determine the energetic flexibility of an appliance. In order to do so it contains a model of the appliance that takes into account the current status of the appliance and the comfort preferences of the end-user (e.g. the washing machine has to finish before 20.00h). Based on that input a control space is composed. When the resource manager receives an allocation it has to translate that into specific instructions for the appliance that it will send to the resource driver. A resource manager may be very sophisticated. Think of a resource manager for a PV panel that uses weather information to predict the amount of energy the PV panel will produce in the coming hours.

Like the energy apps the resource driver and resource manager can be installed via the app store. The resource driver and manager will in the future typically be provided by appliance vendor.

3.1.4 Connect

The connect layer has support for the different protocols that are used to communicate with appliances; Zigbee, Z-wave, WiFi, PLC, etc. The protocol implementations in the connect layer can be reused for different appliances.

3.1.5 SM

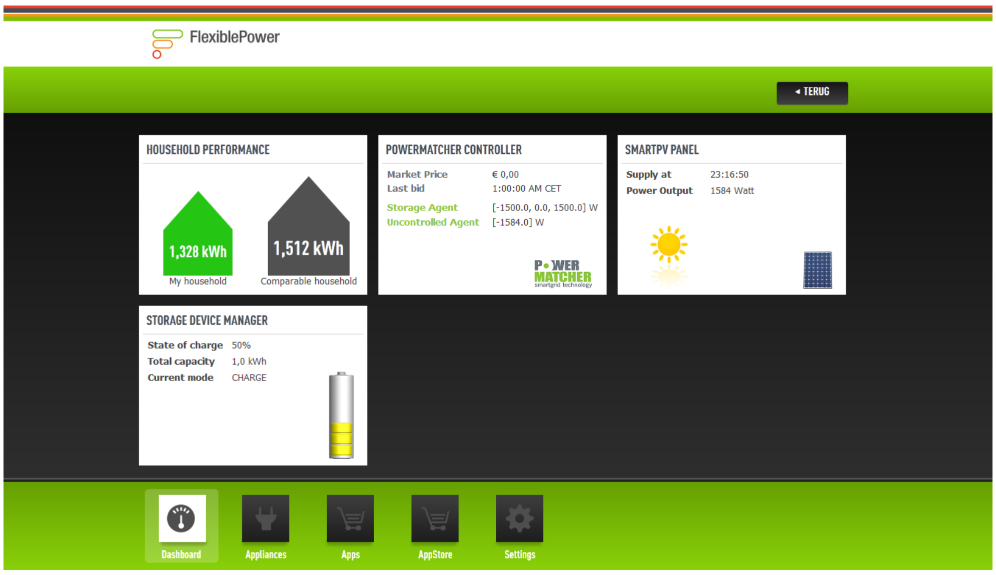
This is used to denote the Management interface through which the runtime connects with the management center. The management center is explained in more detail in section 3.2.

3.1.6 AS

This is the interface towards the app store. Through this interface the runtime can see which energy apps are available. After the selection of an app this interface is also used to perform the installation of the app.

3.1.7 UI

Through the User Interface the end-user interacts with the FP runtime. The user interface has a modular setup (like the runtime itself). Developers can develop a widget that can be deployed in the user interface. An energy app will typically come with a widget that provides feedback and control to the end-user.


FlexiblePower

◂ TERUG

HOUSEHOLD PERFORMANCE

1,328 kWh
My household

1,512 kWh
Comparable household

POWERMATCHER CONTROLLER

Market Price	€ 0,00
Last bid	1:00:00 AM CET
Storage Agent	[-1500.0, 0.0, 1500.0] W
Uncontrolled Agent	[-1584.0] W

SMARTPV PANEL

Supply at	23:16:50
Power Output	1584 Watt

STORAGE DEVICE MANAGER

State of charge	50%
Total capacity	1,0 kWh
Current mode	CHARGE

Figure 4: User interface screenshot

Figure 4 shows a few widget examples. There are two energy app widgets (a benchmark app and a PowerMatcher app) and two resource manager apps (one for a PV panel and one for a storage device).

The UI is web based and is therefore accessible for a wide variety of devices such as tablets, smart phones and PC's. Besides the web interface a set of web services will also become available to interact with the FP runtime. These can be used to develop native apps for Apple IOS and Android for instance.

3.2 Management center

The management center is used to remotely monitor, control, restore and update a large number of FP runtimes. Various protocols will be used to implement the communication between the management center and the runtime, but all will be IP-based so that a common internet connection will suffice.

The most important management protocol that will be used for the management center is TR-69 (see [TR-69, CPE WAN Management Protocol, version 1.3] at http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf). This protocol is widely used by telco's for the management of devices such as ADSL modems. The IP connection and discovery of the Node within a Household will be accomplished by the fact that the TR-96 controlled FP runtime will take initiative to request configuration. Because the connection is setup by the FP runtime an end-user does not need to change the configuration of his/her firewall. There will need to be some basic mechanism for the Management Center to recognise a runtime and link it to a specific Household (e.g. based on the Mac-address of supplied home box).

3.3 App store

The app store contains energy apps and resource drivers and managers. Before software is placed in the app store it will be checked for correct functioning and the presence of malicious code. When the software is approved it will be signed so that the end user can be sure that the software is to be trusted.

Before the app will be downloaded and installed the user is asked to grant the permissions that the app requires. The app can only operate within the permissions that were granted by the end-user. This is enforced by a security framework that is part of the FP runtime.

After the granting of the permissions the app will be downloaded, installed and automatically started. Most apps will be packaged with a widget that will be installed at the same time.

4 Implementation choices

The preceding chapters mostly focused on the concepts behind the FlexiblePower Application Infrastructure. Of course these concepts have to be implemented before they can be used in practice. It is also important for third parties to know which technical requirements they have to meet when developing energy apps and resource drivers and managers.

This chapter briefly covers a number of technical requirements for the FlexiblePower Application Infrastructure. This is followed by the explanation of the rationale behind the choice for the OSGI platform.

4.1 Implementation requirements

From the concepts of the FlexiblePower Application Infrastructure one can derive some important technical requirements:

- **Modularity.** This is one of the main starting points of the FP Application Infrastructure. It should be possible to deploy new modules (such as energy apps) and to replace old modules with new ones.
- **Hot Swappable.** The application infrastructure needs to continue functioning normally during the deployment of a module.
- **Hardware independent.** The application infrastructure will have to run on a wide variety of hardware platforms. This way the most appropriate hardware can be selected for every situation. For a typical household deployment costs are an important factor; the hardware should not be expensive (< 100 euro's), whereas an industrial application may run on a powerful server.
- **Lightweight.** The implementation of the Application Infrastructure should not result in high performance requirements to ensure that the software can run on low-cost hardware (see also the previous bullet).
- **Reliability.** Supply and Demand Management is a critical application; failing to meet agreements due to the outage of the system may result in financial consequences.

4.2 OSGI platform

The choice was made to implement the FP Application Infrastructure on top the OSGI platform. OSGI is an open standard based on Java that supports modular software.

OSGI modules can be remotely installed, started, stopped, updated and removed without the need for a system reboot. This support the *modularity* and *hot swappable* requirements.

Because of the fact that OSGI runs on top of Java it is supported by a large number of hardware platforms. This satisfies the *hardware independent* requirement.

Implementations of OSGI add little overhead to Java; it therefore meets the *lightweight* requirement.

There are various different OSGI implementations, both commercial as open source ones. The OSGI specifications are being used in a wide variety of applications such as mobile telephony, car industry, industrial automation, application servers, grid computing, etc. The broad application of OSGI indicates that is being considered as a *reliable* platform.

4.3 Reference implementation

A reference implementation of the FP Application Infrastructure has been developed. It features resource managers and drivers for each of the four appliance categories (timeshifter, buffer, storage, uncontrolled load/generation). The PowerMatcher has been implemented as an example of an energy app.

The reference implementation makes use of the Apache Felix OSGI platform. This is a widely known open source implementation of the OSGI specs.

A detailed functional design accompanies the reference implementation of the FP Application Infrastructure.