# CONTENTS

Introduction ........................................................................................................... 1  
   Why a recommendation and not standard? .................................................. 2  
Heucod Architecture .......................................................................................... 6  
   Logical architecture overview....................................................................... 7  
Heucod Event Ontology ....................................................................................... 9  
   Events of daily living (EDL) ....................................................................... 10  
   Activities of Daily Living (ADL)................................................................. 11  
   Instrumental Activities of Daily Living (ADL)........................................... 12  
   Adverse Events (AE).................................................................................. 13  
   Basic Event Properties.............................................................................. 14  
   Basic Event JSON Format......................................................................... 15  
Heucod Reference Implementation ................................................................ 16  
   HEUCOD RECOMMENDATION ONTOLOGY EVENTS ......................... 16  
      .NET (C#) mapping of BasicEvent......................................................... 17  
      Basic heucod persistence service mongo db ...................................... 24  
      heucod WEB TOOL ........................................................................... 25  
      HEUCOD EVENTS OF DAILY LIVING SIMULATOR ....................... 28  
      Abnormal activity sensor service......................................................... 29  

---
INTRODUCTION

This section introduces the HEUCOD RECOMMENDATION starting with a brief background and motivation section, followed by the intended use of HEUCOD.

HEUCOD is an acronym for Healthcare Equipment Usage and Context Data and targets a larger ecosystem of assisted living, telecare, telehealth, and telemedicine standards and products, including internet of things (IoT), connected care sensors, devices, and software. HEUCOD is an industry-driven open recommendation which fits into this overarching ecosystem making it easy for industry to adhere to common standards for data collection e.g., supplying sensors, services, and user interfaces for citizens and caregivers where adherence to the recommendation implies that it is easy to combine products from different vendors, and for end-user organizations to setup systematic tracking of the products they have procured, including data on frequency and type of use. For instance, a public home care service organization would want to know how often their rehabilitation equipment is being used, but they could also be interested in understanding whether a new circadian lighting system has an effect on sleep quality and general activity level of their residents. At the same time, the product vendors could be interested in tracking how much their products are being used, and even whether their products need maintenance. All of this is supported by the HEUCOD recommendation. Furthermore, HEUCOD is an open and free-to-use recommendation, allowing for any embedded sensor gateway extension that communicates with a range of home, care, wellness, and medical devices, and which supplements personal computer applications and web servers and web services. In addition to the recommendation, HEUCOD supplies an open-source reference implementation, which makes it easy for commercial vendors to implement the recommendation.
WHY A RECOMMENDATION AND NOT STANDARD?

HEUCOD is a recommendation of “standardized use”, rather than a standard. A standard usually implies something that is legally enforceable and/or standardized and maintained by a recognized standardization body. Instead, HEUCOD uses the approach of the world wide web community’s (W3C) definitions, which are called “recommendations”, which is used for web-technologies HTML, HTTP, XML etc. These are all recommendations rather than standards, as they do not dictate how they should be used or implemented.

BACKGROUND AND MOTIVATION

Today, a gap exists in the care technology sector, where private sector vendors, including manufacturers and resellers of care products such as the three Danish care sector companies PRESSALIT, LIGHTCARE, and MEDICA, and their customers, including care organizations such as AARHS KOMMUNE and other public and private care organizations, are not able to reliably identify which of their care and healthcare technology products are being used, where, how often, and to which effect. Thus, important data on device usage and the context-of-use is either not being recorded or are only registered by the vendors themselves for internal use in closed systems, in which case the data is most often inaccessible for the care institutions, while data collected by the care organizations are in turn often not available to private sector vendors due to problems with the new general data protection regulations (GDPR) and other data-ownership issues.

This implies that valuable usage and context information is lost to both the private sector care product vendors and the public care institutions. Data which could be used to improve product innovation and development for the vendors through smart industry methods, including data mining, statistical modelling, and machine learning, as well as data which could be used to measure and optimize the deployment of care devices in the care organizations.

While the vendors would like more knowledge and understanding on how their devices are being used and in which context-of-use, the care organizations would like be able to analyze the feasibility of their care product investments and patterns of use, and maybe even identify new usages for the products they have invested in. Also, the care institutions currently have no way of identifying products that are not being used anymore and which could have been put to good use elsewhere.

Furthermore, both vendors and care organizations are jointly missing out on the opportunity to do research into new use cases of how usage data can be used for increased care and safety, e.g. for the early detection of the onset of illness, which international research have found to be a highly promising field.

Due to these challenges and opportunities, public care organizations have raised the possibility of requiring all care product vendors to be able to deliver data to a joint usage
data repository for future tenders and product procurement processes. This move is welcomed by both industry and research organizations represented by AARHUS UNIVERSITY in HEUCOD, as this can be seen as a great business opportunity, as well as an opportunity to support more open data and open science, which has proven valuable in order to create new services and end-user decision support applications.

Currently, there are no existing products, technologies, or relevant recommendations that allows for gathering care equipment usage data from many different vendors and combine these data with the specific context-of-use which is often highly relevant in the care and healthcare sectors. Thus, the HEUCOD group of industry, research, and end-user organizations, all of whom are working in the care and healthcare sectors, have found that there is a clear need for creating a common and open industry standard with a supporting open source reference implementation, in order to secure support for the standard from their customers, e.g. public care organizations. This work requires a bottom-up analytic and experimental approach where each industrial partner investigates which usage and context data would be relevant to share for their particular products, and how to best share the data while also protecting their business interests. At the same time, industry partners will gain access to increased usage data of their own products, procured via the public care organizations, and properly anonymized in order to avoid data protection and privacy issues emerging.

Thus, the overall aim of the HEUCOD recommendation is to enable private care product vendors and public and private care organizations to better monitor the actual usage of the deployed healthcare technologies using a shared open standard and data collection platform, combing it with added knowledge of the context-of-use. The aim of the HEUCOD project is thus to pursue the following activities:

1. Analyzing the need for, and designing and developing an open industry standard for capturing relevant usage and context data from medico, telehealth and assisted living equipment, and providing easy and secure data access for private vendors, care managers, caregivers, and decision and policy makers from public and private end-user organizations in the healthcare sector.

2. Developing specific proof-of-concept prototypes for the participating partners, in order to demonstrate the usage of the open source reference implementation platform.

3. Documenting the HEUCOD recommendation with working code samples and a reference implementation

With the HEUCOD open industry recommendation it will be possible to:

1. Analyze the potential of combining relevant data on equipment use and context, to allow for better evaluation of the relevance and cost-effectiveness of the investments made using machine learning and data mining methods and technologies,
2. Provide insights into the potential for detecting missing, improper, or erroneous use of care equipment,

3. Investigate the potential for achieving tertiary effects that have not yet been identified which may potentially be achieved in the future based on HEUCOD collected internet of things (IoT) data, e.g. by looking for patterns of equipment usage correlated to electronic patient journal records using machine learning and statistical modelling.

HEUCOD INTENDED USE

The HEUCOD recommendation is a communication and interface recommendation. Its main contributions are provided in the shape of an ontology of events to be exchanged between sensors, services (including decision support algorithms), and user interfaces for citizens and their families, caregivers, and for care givers and decision makers.

HEUCOD is concerned only with data distribution and server-side components. Thus, HEUCOD does not support any specific sensors, gateways or devices on the client side. Instead, HEUCOD uses commercially available sensors and gateways to supply the data. As the recommendation is relying on established IoT protocols and frameworks, it is easy to adopt third-party sensors and equipment to deliver the data.

For showcasing the reference implementation, HEUCOD is using the CARIOT gateway made by companies MEDICA and ALIVIATE to supply bed occupancy, room movement, and toothbrushing information. A license can be acquired for the CARIOT gateway, including full source code and hardware schematics. The CARIOT gateway is easily replaceable with third party gateways and sensors, and the HEUCOD project aims to present several types of sensor platforms that will support data collection. Also, HEUCOD uses a collection of three sensors (toilet, soap-dispenser, and wash-basin sensors) from company PRESSALIT to showcase toilet and hygiene sensing, data collection, and decision support. Finally, it is using circadian lighting information supplied by company LIGHTCARE.

HEUCOD is designed to help avoid vendor lock-in issues for end-user organizations, allowing end-user organizations to require all products to adhere to the HEUCOD recommendation during a procurement process. Likewise, lack of support for the HEUCOD recommendation is not a problem for product vendors, as the open recommendation allows for easy and low-cost integration, as it is based on open standards and freely available state-of-the-art software components and contemporary communication standards and protocols.
The HEUCOD recommendation and reference implementation is not intended as:
- A diagnostic and decision support tool
- An active patient monitoring device
- An emergency room monitoring device
HEUCOD ARCHITECTURE

In this section, the architecture of the HEUCOD recommendation and reference implementation is described. This include an overview of the ontologies and services provided by HEUCOD, and the communication broker system used.

HEUCOD is intended to be used by industry vendors for their existing and new devices. They can either use HEUCOD directly in the individual sensor or gateway devices in the home setting, or retrofit and adapt to HEUCOD services from their own backend systems, if a data collection system is already in place. This means, that no products need to be modified in order to suit HEUCOD. Instead, those features which are requested by an end-user organization, can be adapted to fit existing data.

HEUCOD can be configured to run in a variety of settings depending on local requirements. Thus, HEUCOD can both run as a collection of distributed cloud services (e.g. on AWS, Azure and other services), on a private self-hosted server system (requires a Windows Server for the reference implementation) e.g. hosted on a municipal private secured network, hosted on a private server in a single care facility for localized monitoring and data collection e.g. in a nursing home, or even in the home of the individual citizen running on a local device for maximum privacy and data protection. In the latter case, warnings caused by discovered adverse events can be allowed to be send, while all activity data of the citizen is kept private and local, or alerts can even be kept local in the home, only available to visiting caregivers.

Also, a HEUCOD server can be created by a public municipal care service, while a private care company can likewise establish a HEUCOD platform, e.g. in case they are providing additional self-paid care and monitoring services to the citizens. Finally, industry vendors can use HEUCOD to monitor the usage and health of their individual products. All of this can be configured after mutual agreement with the end-users and end-user organizations.

In this way, HEUCOD provides maximum flexibility in order to adapt to localized requirements to security and ethics, especially in support of the European Union’s common General Data Protection Regulation (GDPR).
LOGICAL ARCHITECTURE OVERVIEW

Figure 1 presents the logical architecture overview of the HEUCOD project based on the OPEN CARE / EVODAY code base and the HEUCOD recommendation. From the bottom of Figure 1 (BLUE) the Sensor layer consists of a long range of sensor components which can send data to the Decision support layer (GREEN) using either REST-based HTTP calls and MQTT calls.

For sensors who do not have IP-communication capability, data can be sent via the Gateway layer (BLACK), which is usually physically manifested as an embedded gateway computer, which can translate BLE and Zigbee (and other formats) into IP or Lora based communication using the HEUCOD IoT industry recommendation.

In the Decision support layer (GREEN), the sensor data are processed using a variety of statistical learning and artificial intelligence algorithms. For instance, a toilet visit, without soap dispensing and wash basin usage will constitute a “non-optimal hygiene” adverse event, which will be available for all users who are monitoring the individual user. Likewise, a missed oral-care event will be reflected in the digital model the user being monitored in the Model layer (GREY). Depending on needs, either the Current Activity of Daily Living is modeled, while also historic ADL can be shown, e.g. for the last 24 hours. Also, the Adverse events service documents all undesired events and happenings. Also, in the Model layer, it is possible to activate persistence services, e.g. for storing events, activities and adverse events in an SQL relation database, or using the HL7 FHIR format. Persistence services are not mandatory however, due to potential privacy issues. However, if an end-user organization wants to document its work in detecting and preventing adverse events, such services should be activated.

Finally, in the Interface layer (ORANGE), third-party systems and users can obtain access to the Model layer. Basically, this works by subscribing to a specific “monitor ID”, which is completely anonymous. After this, MMQT events (or REST HTTP-requests) are provided for the caller of these integration services.
Figure 1. Provides an overview of a range of recommended HEUCOD components. At the bottom layer we find a range of typical HEUCOD sensors. These sensors range from highly advanced devices with artificial intelligence, to basic sensors that can simply deliver state information to the gateway or server layers. Some will be able to feature a decision support layer themselves, while others need a gateway and a server-based service to work. The gateway level is relevant for those scenarios where several sensors are needed in the home setting. Rather than using expensive infrastructure, this can be shared using a gateway device. Next, the decision support layer, which can either run directly on the gateway in the home of the user, or on a private server or in a cloud configuration. These services combine knowledge from one or more sensors, and even from other services, in order to provide an advanced decision support model, informing the model layer of relevant new events, or the interface layer directly.
HEUCOD EVENT ONTOLOGY

In the HEUCOD recommendation events are the main concept and target of standardization. HEUCOD recommends a range of event types, each of which can be used to describe an important event or activity of daily living, including events such as “user is home” – seeing a user arrive home, bed occupancy if a user is in bed, room movement if the user is moving. HEUCOD does not dictate that these events should be used, but recommends them to describe common human activity in meaningful and standardized manner. HEUCOD events also include adverse events, e.g. if a fall has been detected, missed hand-washing after a toilet visit, or missed toothbrushing.

In HEUCOD, a broker pattern is assumed, meaning that all events are delivered to one or more brokers. On these brokers, one or several services are listening, and each of them are capable of combining events with previous knowledge and possible machine learning. These are the services introduced in Figure 1, and thus the logical architecture overview. In turn, these services can themselves contribute with events passed to the broker, which other services may then use. In the end, events are picked up by persistency services, and user interface services. This could be localized clients in the home of the user, providing nudging, e.g., “remember to wash your hands” or “remember to brush your teeth”, but it could also be warnings to caregivers, e.g., “a citizen has been in bed for more than 9 hours”, or “a citizen has not been to the toilet for more than 4 hours”, or “a citizen has not brushed her teeth for the last three days”.

The ontology is arranged as an inheritance hierarchy, which can be expanded indefinitely. At the top level, we have the BasicEvent type. Also, an AdvancedEvent type is work in progress. The BasicEvent can be used to describe any event, and the HEUCOD brokers will accept any BasicEvent, or one it’s descendants.

Below a range of UML diagrams are presented showcasing the various event types, arranged in a hierarchy of events, as shown in Figure 2.

![UML Diagram](image-url)
**Events of daily living (EDL)**

Just below the Basic Event we find the EDL event family (Events of Daily Living), which are the events which are naturally occurring in the home setting of users, as shown in Figure 3. EDL's are typically very basic in nature, and are usually tightly interwoven with activities of daily living ADL's (see below) and instrumented ADL's (IADL). From a collection of EDL’s - a classification algorithm can typically infer relevant ADL and IADL activities, as well as adverse events (AE - see below) - for example a range of EDL events in a given timeframe - e.g., during a 10-minute window - can be used to infer whether a fall has occurred or whether the person was on the toilet and would perform correct handwashing operations. If not, an adverse event can be signaled. In turn, an AE event listener service will pick this up, and decide whether to raise an alert to the designated contact person. In a nursing home scenario, this could be for the caregiver office, and for home care, it could be to the smartphone of a relative.

![Figure 3. The EDL family of events. Events of daily living (EDL) are the naturally occurring events related to human activity in the home setting and beyond. The standard does not limit further definitions of EDL events but recommends using the once provided in the standard.](image-url)
Activities of Daily Living (ADL)

Activities of daily living (ADLs) are routine activities people do every day without assistance. ADL is thus the basic activities of daily life that most people are used to be doing - usually without assistance.

The concept of ADLs was originally proposed in the 1950s by Sidney Katz at the Benjamin Rose Hospital in Cleveland, Ohio and has been added to and refined by a variety of researchers since that time.

Assisted living facilities, in-home care providers, and nursing homes specialize in providing care and services to those who cannot perform ADLs for themselves. An overlap between EDL and ADL can be expected, as ADL holds added clinical meaning.

There are six basic ADLs: eating (self-feeding), bathing, personal hygiene (washing hands - brushing teeth) and grooming (including brushing/combing/styling hair), getting dressed, toileting, transferring (functional mobility).

The performance of these ADLs is important in determining what type of long-term care and health coverage, such as Medicare, Medicaid in the USA, public homecare or long-term care support in parts of Europe.

There is a hierarchy to where the early-loss function is hygiene, while the mid-loss functions are toilet use and transferring, and the late loss function is eating. When there is only one remaining area in which the person is independent, there is around 63% chance that it is eating and only a 4% chance that it is hygiene. Thus, tracking a lack of hygiene can be a powerful indicator for an emerging lack of cognitive or physical function.

The ADL definition in HEUCOD is expanded to provide Sleeping and Resting, Brushing, Socializing, Watching TV, and Exercising, as shown in Figure 4.

Figure 4. Activates of daily living (ADL events.) is a separate class of events used to describe the function of the user.
**Instrumental Activities of Daily Living (ADL)**

Instrumental activities of daily living (IADL) are not necessary for fundamental functioning, but they let an individual live independently in a community, and can thus be important to monitor as well.

Along with ADL, IADL is often used to assess the level of independence of the individual citizen.

Multiple scales or instruments for the evaluation of IADLs exist. These include, but are not limited to, Lawton and Brody IADL, Health and Retirement Study Care Questionnaire, and Pfeffer Functional Activities Questionnaire. Each instrument includes its definition of IADL disability. Therefore, in the use of analyzing for cognitive impairment, the results differ depending on which instrument the evaluator used.

One of the significant limitations that all these instruments have in common is that they are often self-reported. A person is biased toward his or her abilities to perform specific tasks. Thus, they may either overestimate or underestimate their abilities.

Another major issue arises when examiners use IADL instruments to detect the probability of dementia in mildly cognitive impaired (MCI) individuals. Studies had shown that there is no difference in the ability to perform certain IADLs between MCI individuals with dementia and MCI individuals without dementia, which may be because current instruments are not sensitive to subtle changes. For example, an individual may have no issues with driving but have trouble adhering to traffic laws. On the Lawton and Brody IADL questionnaire, this individual would receive full points in the evaluation of the mode of transportation. See Figure 5.

---

**Figure 5. Instrumented Activities of Daily Living (IADL)**
**Adverse Events (AE)**

An adverse event is an incident that results in harm to the patient or citizen. Adverse events commonly experienced include falls, medication errors, malnutrition, incontinence, and hospital-acquired pressure injuries and infections. It can also include missed personal hygiene and brushing actions.

Physical and cognitive functional decline can have a significant impact on a person’s ability to perform activities of daily living. Older people are particularly vulnerable to experiencing adverse events due to inherent complexity in managing their care and a decline in physiological reserves.

Adverse events in patients in home healthcare include (but not limited to) the following list (source Schildmeijer, K., Unbeck, M., Ekstedt, M., Lindblad, M., & Nilsson, L. (2018)). Adverse events in patients in home healthcare: a retrospective record review using trigger tool methodology. BMJ open, 8(1), e019267. 
https://doi.org/10.1136/bmjopen-2017-019267):

1. Pressure ulcer
2. Fall
3. Undernutrition
4. Insufficient oral health
5. Moderate/severe gastrointestinal problem
6. Distended urinary bladder
7. Escape from home/special accommodation

Figure 6. Adverse events (AE)
Basic Event Properties

In the UML class diagram shown as Figure 7, the properties of the Basic Event are shown. The detailed type information can be found in the source code of the reference implementation. Also, a JSON representation is available.

It is recommended to fill out “PatientID”, “SensorID”, and “SensorType” and “Timestamp” as a minimum level of information. However, this is not required, but useful if you wish to work with the reference implementation services. Also, a value and a unit is recommended. It is also possible to use the “Advanced” field to add additional dynamic types and extensions.

Most descendants of BasicEvent does not include added properties according to the recommendation, but each manufacturer is free to implement additional fields as required. It is not recommended to change any of the existing fields. Also, in order to keep services compatible, it is recommended to not require special fields and data structures to be present, as this would likely break the open nature of the recommendation.

All fields are “nullable” and then corresponding JSON format accepts completely empty fields.

The SensorType is used to identify the type of event which is used. Please be aware that this field will change name to EventType for the next release version.
Basic Event JSON Format

A typical event could follow the below JSON format

```json
{"timestamp": 1595598365, "value": 1, "room": "", "eventTypeEnum": 82098, 
"startTime": 1595634723, "directEvent": false, "patientId": "3", "monitorId": "3", 
"sensorId": "FF:FF:30:00:1C:AA", "sensorType": 
"OpenCare.EVODAY.EDL.UserIsHomeEvent", "gatewayId": "25002"}
```

This particular even is broadcasting a UserIsHome event, meaning that the user has been sensed being at home.
HEUCOD REFERENCE IMPLEMENTATION

In this section, we provide an introduction to the HEUCOD reference implementation. An overview is provided including screenshots of demonstration services and clients. These services and clients can be found in an open source repository. PLEASE NOTE THIS IS A DRAFT DOCUMENT- and thus all source code is currently still in closed repository until the codebase has been tested to be running stable. Also, we demonstrate the HEUCOD server components in use with an actual commercial gateway device, using a range of sensors, the CARIOT FLEX gateway. In this section, we also present a “Smart Mirror Client” prototype, which showcases how HEUCOD can be used to easily build innovative advanced ICT and IoT products.

The HEUCOD reference implementation showcases the use of the HEUCOD recommendation. It is meant as inspiration for other developers to help them understand the principles and basic operations of HEUCOD seen from a working codebase with support hardware. It can indeed be used as a full server stack for further industrial and/or academic development, experimentation, and– as it is implemented using .NET code with FreeBSD open source license. In addition, most of the basic services can run on both Windows and LINUX, while some demonstration clients are made specifically for Windows and Android.

HEUCOD RECOMMENDATION ONTOLOGY EVENTS

The HEUCOD recommendation mainly builds on the BasicEvent and its descendents. In the HEUCOD reference implementation, both JSON, .NET (C#), Python and C implementations of these mappings are supported.
Below the C# mapping is shown:

```csharp
namespace OpenCare.EVODAY
{
    using System;
    using System.Collections.Generic;
    using System.Globalization;
    using Newtonsoft.Json;
    using Newtonsoft.Json.Converters;

    /// <summary>
    /// BasicEvent is the basic model and DTO (data transfer object) for most direct sensor events and interpreted event
    /// as well as for technical events occurring.
    /// Basic events represent the basic generic event type used by the HEUCOD standard and the OpenCare
    /// architecture.
    /// BasicEvent contains most of the basic relevant (but optional) fields that are supported in HEUCOD.
    /// For simplicity, we recommend not introducing too many fields in the subclasses of BasicEvent,
    /// in order to support a simple
    /// code base in the HEUCOD standard. However, it is naturally allowed to add additional attributes to the subclasses
    /// which should preferably contain very vendor specific code, which one would not expect
    /// third party service providers
    /// to use or support.
    /// </summary>
    public partial class BasicEvent
    {
        /// <summary>
        /// The unique ID of the event. Usually a GUID or UUID but one is free to choose
        /// This is decided by the Sensor RTC clock parameter. Default is true.
        /// Also, please check for SendingDelay parameter, e.g. if the sensor has delayed communications,
        /// it can inform the server of this
        /// </summary>
        [JsonProperty("id", NullValueHandling = NullValueHandling.Ignore)]
        public string Id { get; set; }

        /// <summary>
        /// The timestamp of the event being created in the UNIX Epoch time format.
        /// For units WITH real time clock - the event is set on the device
        /// For units WITHOUT a real time clock - the event is set on the receiving server
        /// This is decided by the Sensor RTC clock parameter. Default is true.
        /// Also, please check for SendingDelay parameter, e.g. if the sensor has delayed communications,
        /// it can inform the server of this
        /// </summary>
        [JsonProperty("timestamp", NullValueHandling = NullValueHandling.Ignore)]
        public long? Timestamp { get; set; }

        /// <summary>
        /// Does the sensor/gateway forwarding this message have an RTC clock.
        /// In other words, can we trust the time to be true, or will the server has to fill this
        /// </summary>
        [JsonProperty("sensorRtcClock", NullValueHandling = NullValueHandling.Ignore)]
        public bool? SensorRTCClock { get; set; }

        /// <summary>
        /// If the sensor/gateway forwarding this message had to wait before sending, e.g. due to buffering
        /// what was the delay in milliseconds. The server may use this information to subtract the delay
        /// </summary>
        [JsonProperty("sendingDelay", NullValueHandling = NullValueHandling.Ignore)]
        public long? SendingDelay { get; set; }

        /// <summary>
        /// ID of the user or patient to whom this event belongs
        /// </summary>
    }
}
```
[JsonProperty("patientId", NullValueHandling = NullValueHandling.Ignore)]
public string PatientId { get; set; }

/// <summary>
/// ID of the caregiver - e.g. one helping with a rehab or care task that is reported
/// </summary>
[JsonProperty("caregiverId", NullValueHandling = NullValueHandling.Ignore)]
public string CaregiverId { get; set; }

/// <summary>
/// The ID that can be used to monitor events of this person. This is mainly
/// used in databaseless applications, but can have many shapes. In some cases
/// the monitor ID can be the same as the patient ID - but this is a string.
/// </summary>
[JsonProperty("monitorId", NullValueHandling = NullValueHandling.Ignore)]
public string MonitorId { get; set; }

/// <summary>
/// Location can be an address or apartment ID. Care should be taken to avoid
/// GDPR complications by only storing mapping to actual locations on a secure
/// device - e.g. on the CITIZEN MONITOR in the nursing office in a nursing
/// setting - where the doors are locked - and data are only available to staff members.
/// </summary>
[JsonProperty("location", NullValueHandling = NullValueHandling.Ignore)]
public string Location { get; set; }

/// <summary>
/// Could be the name or identifier of the care facility or care organization
/// responsible for the event.
/// </summary>
[JsonProperty("site", NullValueHandling = NullValueHandling.Ignore)]
public string Site { get; set; }

/// <summary>
/// Name of the room where the event occurred (if any).
/// </summary>
[JsonProperty("room", NullValueHandling = NullValueHandling.Ignore)]
public string Room { get; set; }

/// <summary>
/// Optional field for street address and number
/// GDPR complications by only storing mapping to actual street addresses on a secure
/// device - e.g. on the CITIZEN MONITOR in the nursing office in a nursing home
/// setting - where the doors are locked - and data are only available to staff members.
/// </summary>
[JsonProperty("streetAddress", NullValueHandling = NullValueHandling.Ignore)]
public string StreetAddress { get; set; }

/// <summary>
/// Optional field for city
/// GDPR complications by only storing mapping to actual locations on a secure
/// device - e.g. on the CITIZEN MONITOR in the nursing office in a nursing home
/// setting - where the doors are locked - and data are only available to staff members.
/// </summary>
[JsonProperty("city", NullValueHandling = NullValueHandling.Ignore)]
public string City { get; set; }
/// <summary>
/// Optimal field for post code
/// GDPR complications by only storing mapping to actual locations on a secure
/// device - e.g. on the CITIZEN MONITOR in the nursing office in a nursing home
/// setting - where the doors are locked - and data are only available to staff members.
/// </summary>
public string postalcode { get; set; }

/// <summary>
/// All sensors should have a unique ID which they continue to use to identify themselves.
/// However, the sensor ID is actually not mandatory to use.
/// </summary>
public string SensorId { get; set; }

private string sensorType;

/// <summary>
/// The type of sensor used. This should preferably match the name of
/// the "class" of the device following the HEUCOD ontology
/// </summary>
public string SensorType
{
    get
    {
        if (sensorType == null)
        {
            sensorType = this.GetType().ToString();
            eventTypeEnum = Utility.HashFunction(sensorType);
        }
        return sensorType;
    }
    set
    {
        sensorType = value;
        eventTypeEnum = Utility.HashFunction(sensorType);
    }
}

/// <summary>
/// The model of the device. While the type should be a standard type if possible
/// the model name or number can be used to differentiate
/// </summary>
public string DeviceModel { get; set; }

/// <summary>
/// The vendor of the device - e.g. who built the
/// device. While the type should be a standard type if possible
/// the vendor name can be used to differentiate between different types
/// </summary>
public string DeviceVendor { get; set; }

private long? eventTypeEnum;
/// The type of event used. This should preferably match the name of
/// the "class" of the device following the HEUCOD ontology naming.
///
/// [JsonProperty("evenTypeEnum", NullValueHandling = NullValueHandling.Ignore)]
public long? EventTypeEnum
{
    get; set;
}

/// The ID of a gateway who is either relaying the event from a sensor
/// or if the event is generated by the gateway itself. Please remember,
/// that some sensors have their own end-to-end communication capabilities,
/// and are thus not tied to a gateway.
/// This value should be grabbed from the hardware platform used (i.e. CPU ID, unique MAC address, etc).
/// No IP address or similar.
///
/// [JsonProperty("gatewayId", NullValueHandling = NullValueHandling.Ignore)]
public string GatewayId { get; set; }

/// The ID of the service generating the event. A service can be a sensor monitoring service,
/// or it could be a higher level service - interpreting data from one or more sensors
/// and even from several sensors, and maybe historical data
///
/// [JsonProperty("serviceId", NullValueHandling = NullValueHandling.Ignore)]
public string ServiceId { get; set; }

/// Is this a direct event? This mean there is no gateway involved.
/// The field is non mandatory
///
/// [JsonProperty("directEvent", NullValueHandling = NullValueHandling.Ignore)]
public bool? DirectEvent { get; set; }

/// The primary value (as a long). If the value is not a number, use the
description field instead.
///
/// [JsonProperty("value", NullValueHandling = NullValueHandling.Ignore)]
public double? Value { get; set; }

/// The secondary value. If there are more than 3 values, use the
advanced field to add data.
///
/// [JsonProperty("value2", NullValueHandling = NullValueHandling.Ignore)]
public double? Value2 { get; set; }

/// The tertiary value. If there are more than 3 values, use the
advanced field to add data.
///
/// [JsonProperty("value3", NullValueHandling = NullValueHandling.Ignore)]
public long? Value3 { get; set; }
/// The unit of the device. The unit can be a simple unit of the SI system, e.g. meters, seconds, grams, or it could be based on other systems. There are many representations for units of measure and in many contexts, particular representations are fixed and required, e.g. in medical contexts the unit of mass is typically grams, in chemistry it might be moles, in the field of electronics it could be amperes. In the context of the device, the unit "mcg" is specified for micrograms.

/// <summary>
/// The unit of the second value from the device. The unit can be a simple unit of the SI system, e.g. meters, seconds, grams, or it could be based on other systems. There are many representations for units of measure and in many contexts, particular representations are fixed and required, e.g. in medical contexts the unit of mass is typically grams, in chemistry it might be moles, in the field of electronics it could be amperes. In the context of the device, the unit "mcg" is specified for micrograms.

/// <summary>
/// The unit of the third value from the device. The unit can be a simple unit of the SI system, e.g. meters, seconds, grams, or it could be based on other systems. There are many representations for units of measure and in many contexts, particular representations are fixed and required, e.g. in medical contexts the unit of mass is typically grams, in chemistry it might be moles, in the field of electronics it could be amperes. In the context of the device, the unit "mcg" is specified for micrograms.

/// The length of the event period - in milliseconds. E.g. if a PIR sensor has detected movement - and it covers 90 seconds, it would be 90 000 ms.

/// For how long is the sensor blind. E.g. a PIR sensor will detect movement - and then send a signal. After this - it will be "blind" typically between 10 and 120 seconds. This is important for the classification services. The value is in seconds.

/// Start of the observed event.

/// End of the observed event.

/// The average power consumption of the device in watts - use together with the length to get the kWh.
/// <summary>
/// The battery level in percentage (0-100). A battery alert service may use this information to send alerts at 10% or 20% battery life - and critical alerts at 0%
/// <summary>

@JsonProperty("battery", NullValueHandling = NullValueHandling.Ignore])
public int? Battery { get; set; }

/// <summary>
/// The self-reported accuracy of the sensor or service event.
/// E.g. a sensor may be 99% sure that it has detected a fall
/// while a classification service may be 88% sure
/// <summary>

@JsonProperty("accuracy", NullValueHandling = NullValueHandling.Ignore])
public int? Accuracy { get; set; }

/// <summary>
/// RSSI stands for Received Signal Strength Indicator. It is often used with radio based networks, including WiFi, BLE, Zigbee, Lora
/// When e.g. used with a BLE beacon, it is the strength of the beacon's signal as seen on the receiving device, e.g. a smartphone. The signal strength depends on distance and Broadcasting Power value. At maximum Broadcasting Power r (+4 dBm) the RSSI ranges from -26 (a few inches) to -100 (40-50 m distance).
/// RSSI is used to approximate distance between the device and the beacon.
/// At maximum Broadcasting Power (+4 dBm) a BLE RSSI usually ranges from -26 (a few inches) to -100 (40-50 m distance).
/// Often it is used along with another value defined by the iBeacon standard: Measured (transmission) Power
/// Due to external factors influencing radio waves such as absorption, interference, or diffraction, RSSI tends to fluctuate.
/// The further away the device is from the beacon, the more unstable the RSSI becomes.
/// <summary>

@JsonProperty("rssi", NullValueHandling = NullValueHandling.Ignore])
public double? RSSI { get; set; }

/// <summary>
/// Measured Power indicates what's the expected RSSI at a distance of 1 meter to the beacon.
/// Combined with RSSI, it allows to estimate the distance between the device and the beacon.
/// <summary>

@JsonProperty("measuredPower", NullValueHandling = NullValueHandling.Ignore])
public double? MeasuredPower { get; set; }

/// <summary>
/// Signal-to-noise ratio (abbreviated SNR or S/N) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to the noise power, often expressed in decibels.
/// <summary>

@JsonProperty("signalToNoiseRatio", NullValueHandling = NullValueHandling.Ignore])
public double? SignalToNoiseRatio { get; set; }

/// <summary>
/// Link Quality (LQ) is the quality of the real data received in a signal.
/// This is a value from 0 to 255, being 255 the best quality.
/// Typically expect from 0 (bad) to 50-90 (good). It is related to RSSI and SNR values as a quality indicator.
/// <summary>

@JsonProperty("linkQuality", NullValueHandling = NullValueHandling.Ignore])
public double? LinkQuality { get; set; }
/// <summary>
/// This field can contain advanced or composite values which are well-known to the specialized vendors
/// </summary>
[JsonProperty("advanced", NullValueHandling = NullValueHandling.Ignore)]
public string Advanced { get; set; }

/// <summary>
/// This field supports adding a prose description of the event - which can e.g. by used
/// for audit and logging purposes.
/// </summary>
[JsonProperty("description", NullValueHandling = NullValueHandling.Ignore)]
public string Description { get; set; }

#region .NET Helper Properties

/// These are helper methods for .NET
public DateTime TimeStampNet
{
    get { return Utils.UnixTimeToDateTime(Timestamp); }
}

public DateTime StartTimeNet
{
    get { return Utils.UnixTimeToDateTime(StartTime); }
}

public DateTime EndTimeNet
{
    get { return Utils.UnixTimeToDateTime(EndTime); }
}

public BasicEvent()
{
    EventTypeEnum = Utils.HashFunction(this.GetType().ToString());
}
#endregion
The HEUCOD recommendation dictates the common usage model of a collection of independent services cooperating through a common HEUCOD bus based on one or more (interconnected) MQTT and/or AQMP broker services. There are many ways this can be configured, catering to differences in security models and architectures, but in the end, being able to persist events are one of the fundamental services of the HEUCOD reference implementation.

Besides a basic event logging solution, logging data to a text file, there are services supporting SQL and OQL databases. The HEUCOD MongoDB service is an example of a basic OQL based persistence service. It allows capturing all events arriving to a MongoDB, from which all other services, including the HEUCOD WEB TOOL – to be presented later, can then access the data.

Most services, however, do not rely on a common data repository, thus making the services stateless, supporting a simple and highly effective approach to load balancing services across many physical or virtual servers. In the end, it will be up to clients to recognize multiple events coming from identical services, or the developers to build an effective signaling model. This also means, that basic persistence services can run on tiny embedded devices, supporting TCP/IP services and .NET or Python, and all the way to cloud services online.

Figure 8. Running the basic HEUCOD to Mongo Persistence Service.

The service can be started as a command line exe – or it can be run as a service.

Figure 9. The HEUCOD to Mongo Persistence service running
HEUCOD WEB TOOL

The HEUCOD reference implementation supports an infinite number of clients accessing the advanced or basic SQL and OQL datastores. This also include web clients and tools. One of these is the HEUCOD WEB TOOL – which is a very basic web viewer for gaining access to persisted events originating from a Mongo DB OQL data service.

The HEUCOD WEB TOOL is based on the Blazor design and C# code, and can run on both Windows and Linux hosts (either running self-hosted) or in a cloud service. Thus, for e.g. a care facility with up to a few hundred homes, it would be sufficient to run the database on a low-cost computer platform, while large scale operations, e.g. an entire municpals combined home care and care facility services, a dedicated server or cloud service would be recommended.

The HEUCOD WEB TOOL supports a range of basic operations, such as “Adverse events”, which will provide an overview of any adverse events arriving. “Events” which is an overview of all events stored in the database. Next “Patients” will provide a list of all patients (or citizens) in the datastore. The approach for basic OQL services is one of dynamic entities. This means, that there is no need to configure and create or import patients, gateways, sensors or similar in the datastore before operations are started. In fact, the HEUCOD WEB TOOL supports a model where patient, sensor, and gateway information is automatically collected from the events arriving, and then used to create a model of those entities. HEUCOD does support a model where an ID is used for patient, sensor and gateway, but this requires the use of more advanced database solutions, and will require more configuration in daily use. Like with “patients”, “sensors” provide an overview of the sensors that have delivered events at some point.

Thus, there are no “CRUD” support in the HEUCOD WEB TOOL. All CRUD is currently the responsibility of the sensors and gateways, or it could be the responsibility of any intermediate services running.
First, a list of adverse events can be found by clicking on the indicated link (see Figure 11).

Also, it is possible to retrieve a full list of events (Figure 12).

Figure 11. An overview of all adverse events received. It is possible to filter on dates, patient id, sensor id, and other filters.

Figure 12. The event overview provides insights into all events arriving from attached sensors and gateways.
Figure 13. The Patient overview presents an overview of all patients being monitored. Due to GDPR, most often there is no data related to the patient, except for the ID.

Figure 14. Presents an overview of the sensors who are sending events to the HEUCOD broker.
HEUCOD EVENTS OF DAILY LIVING SIMULATOR

The HEUCOD Events of Daily Living Simulator is a Window only tool which can be used in the development and testing of new HEUCOD components, both sensors, services, and user interfaces. See Figure 15 for the user interface.

It features a built-in MQTT connection to the HEUCOD bus and broker service. Also, it allows playing a series of events of daily living as MQTT messages, which can be used to simulate user actions. Also, it has a series of buttons (bottom), which allows triggering/simulating events of daily living messages, which is sent to the server and database.

Also, it features two sample night-time fall bathroom detection algorithms, which can detect a user falling during a night time bathroom visit. These two algorithms are based on the machine learning algorithms KNN and Decision Trees (DT). However, these algorithms could easily be placed in another service, or even on another hardware node.
ABNORMAL ACTIVITY SENSOR SERVICE

The Abnormal Activity Sensor Service listens for a range of events from the HEUCOD bus. It monitors whether the activity level of the user indicates abnormal behavior, and if this is the case, it will send either notification events or even adverse events to the HEUCOD broker. Thus, any client monitoring the HEUCOD broker will be informed, and can choose to inform any attached users.

![Image of the service]

Figure 16. The service is a headless service. It can be started as an executable or as a service.

This service was made as a show case for creating simple services reacting to a series of events and can be used along with the simulator for instance.

However, before real world usage, one needs a gateway for the home setting. Any gateway may be used, as long as it implements the HEUCOD recommendation. In the following we present a low-cost gateway system.