METHOD ARTICLE

Internet-of-Things object model [version 1; peer review: awaiting peer review]

Rong Long¹, Xiaohui Fan¹, Kai Wei², Junxuan Bai¹, Shanpeng Xiao¹

¹China Mobile Research Institute, Beijing, 100053, China
²China Mobile IOT Company Limited, Chongqing, 401121, China

Abstract

Background: With the advancement of communication technology and advanced sensors, there are massive demands for Internet-of-Things (IoT) applications in buildings, communities, factories, parks, etc. Accessing IoT devices provides convenience for scene management and monitoring, ameliorating production and life intelligently. However, due to the lack of a unified model for IoT devices, data is often skipped over IoT platforms and transmitted to applications directly. This leads to the fact that each manufacturer needs to produce its devices and develop its customized software, which hugely increases the development cycle. On the other hand, it is difficult to convey information between different systems, limiting cross-system control. Moreover, digital twin relies on large amounts of heterogeneous data, and it is impracticable to provide enough data without a unified model for device description.

Methods: First, we illustrate the motivation, design goals, and design principles for creating the Internet-of-Things Object Model (IoT-OM). Then we propose a unified description to define IoT devices. The proposed concept has been accepted by several companies, and we analyse one platform that adopts the model. To demonstrate the effectiveness of the model, we introduce two projects based on the platform. One project is an intelligent fire protection system, and another project is an intelligent air quality monitoring system.

Results: We measured the time taken by five companies when developing IoT devices and their applications, including the development cycle duration without utilizing the proposed model and the duration using the model at China Mobile's OneNET platform. The results prove that the proposed model can significantly shorten the development cycle.

Conclusions: This paper proposes a model for IoT devices, which helps to unify heterogeneous data among different manufacturers and helps to shorten the development cycles for developers.

Keywords
Object model, Internet of Things, development cycle, digital twin
Corresponding authors: Rong Long (longrong@chinamobile.com), Junxuan Bai (baijx6@163.com), Shanpeng Xiao (xiaoshanpeng@chinamobile.com)

Author roles: Long R: Conceptualization, Data Curation, Software, Writing – Review & Editing; Fan X: Funding Acquisition, Project Administration, Supervision, Writing – Review & Editing; Wei K: Resources, Software, Validation; Bai J: Investigation, Writing – Original Draft Preparation, Writing – Review & Editing; Xiao S: Funding Acquisition, Supervision, Writing – Review & Editing

Competing interests: Long Rong, Xiaohui Fan, Junxuan Bai, and Shanpeng Xiao are affiliated with (and receive salary from) China Mobile Research Institute, and they are responsible for the demand research and design of IoT-OM. Kai Wei is affiliated with (and receives salary from) China Mobile IOT Company Limited, and is responsible for collecting the properties and functions of IoT devices.

Grant information: This work was supported by the National Natural Science Foundation of China [U21B2029]. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2022 Long R et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Long R, Fan X, Wei K et al. Internet-of-Things object model [version 1; peer review: awaiting peer review] Digital Twin 2022, 2:5 https://doi.org/10.12688/digitaltwin.17562.1

First published: 12 Apr 2022, 2:5 https://doi.org/10.12688/digitaltwin.17562.1
Introduction

With the development of communication technology and new sensors, there are massive demands for Internet-of-Things (IoT) applications in buildings, communities, factories, parks, etc. For example, fire detection of a surveillance system can be notably enhanced by connecting monitoring IoT devices in buildings or outdoor scenes\(^1\), and construction sites tend to apply IoT devices and digital twin techniques for new projects\(^2\). Also, remote production management can be achieved in some factories by accessing IoT manipulators\(^3\)–\(^6\). In large parks, IoT devices are set up in public areas to scan the passenger volume or traffic load\(^7\)–\(^10\), thereby avoiding security incidents. It can be observed that the accessing of IoT devices provides convenience for production and human life\(^11\),\(^12\).

However, since IoT devices are produced by different manufacturers, a unified model for describing devices is missing. After being encapsulated on IoT devices, data is often transmitted to applications directly through the gateway, whereas IoT platforms are skipped. As the data path becomes longer, the timeliness of the system will be weakened. In addition, it is laborious to expand the scale of IoT devices under these circumstances. Digital twin relies on large amounts of heterogeneous data. As a result, the perception and simulation abilities of digital twin cannot be fully developed.

From an industrial point of view, the current development process will ultimately lead to three difficulties: (1) due to the low-level data sharing, it is hard to provide a large amount of data for potential digital twin applications; (2) the cost for business deployment is high; (3) there are numerous obstacles for industrial cooperation. Specifically, since different companies have defined their own model for device description, no connection is constructed between massive data, which leads to low data utilization. In addition, due to different standards of devices, new applications need to be customized and developed multiple times to obey the standards, which increases the cost of service deployment. Last but not least, due to different descriptions, each company has its own production chain, making it difficult for collaboration, and increasing the difficulty for maintenance.

In this paper, we propose a unified IoT object model, called IoT-OM, to describe the devices, which is capable of dealing with the problems mentioned above. The model describes the attributes and capabilities of the device over existing network protocols, forming the twin entity in virtual space. In order to explain the proposed unified object model, we introduce two projects to verify our model. In addition, a statistical result proves that the proposed model can significantly shorten the development cycle.

Motivation, objectives, principles, and definition of IoT-OM

Nowadays, most IoT devices transmit data based on the classic network model, i.e., Open System Interconnection Reference Model (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP). This paradigm requires manufacturers not only to master product-related techniques but also to acquire network communication knowledge. Therefore, this will increase the workload for manufacturers and restrict the deployment of IoT devices.

To handle the problem, we propose a unified model for describing the device in the classic network model, called IoT Object Model (IoT-OM). As illustrated in Figure 1, the data for IoT-OM is transmitted in Application Layer in TCP/IP network model\(^11\).

From a business perspective, IoT-OM facilitates device management for IoT platforms. As shown in Figure 2, IoT-OM locates between the terminal devices and applications. After employing IoT-OM, the IoT platform can rapidly obtain device information, which is supplied by different manufacturers. Moreover, it is effortless to provide plenty of services, such as data storage, online debugging, key management.

The design objectives of IoT-OM is listed as follows:

- There is a need to design a unified description method to realize data sharing for devices from different manufacturers.
• It is necessary to decouple the data for device access and application, providing developers with concise data without considering the hardware.

• It should not be complicated to access terminal devices, which can reduce the setup difficulties for manufacturers.

Therefore, when designing IoT-OM, there are several principles to follow:

• Simplicity. The model needs to be independent of network and protocols and pay more attention to device description. It should provide manufacturers with a simple and intuitive understanding.

• Compatibility. In order to be compatible with devices produced from different manufacturers, the model should be divided into common attributes and specific attributes. Different devices contain the same common attributes and include specific attributes according to functions and purposes. Moreover, templates should be provided for different industries.

• Scalability. Due to the rapid advancement of smart devices, IoT-OM should support future services.

To meet the requirements mentioned above, we define the hierarchical architecture of the unified object model in Figure 3. As demonstrated in Figure 3, IoT-OM consists of three levels: Devices, Components, and Functions.

• Devices level is the top level of the model, which defines different IoT devices, but is not responsible for specific capabilities. Taking a house as an example, it includes lights, displays, curtains, door locks, etc.

• Components are divided into Public Components and Industrial Components.
  – Public Components record shared information and capabilities, for example, a switching component. Each device has a switching component to start and stop the device.
  – Industrial Components record specific information and capabilities. For instance, only washing machines can wash clothes, so washing-related components are regarded as industrial components.

• Functions are partitioned into Properties, Actions, and Events.
  – Properties define what the device is, and the device states when it is running.
  – Actions define what the device can do and describe the capabilities that the platform can employ.
  – Events define periodic messages or alarm messages.

Specifically, the composition of an IoT device is defined in Figure 4.

• omThing defines the entire IoT device, but it does not contain specific content. It corresponds to the Devices level.

• omInfo describes what the device is, its manufacturer, brand, date of production, etc.

• omComponent corresponds to the Components level, and each component describes a capability of the device. Also, a device may consist of multiple components.

• omProperty corresponds to the Properties module. It contains readable variables, parameters, and states of the device.

• omAction corresponds to the Actions module. It describes active functions of the platform, such as obtaining device information, changing device states, etc.

• omEvent corresponds to the Events module. It describes passive process of the device, such as periodic message reporting, abnormal alarm, etc.

• omSchema describes the formats and the ranges of data, such as type, default value, maximum and minimum, etc.

Figure 3. The hierarchical architecture of IoT-OM (Internet-of-Things Object Model).

Figure 4. The composition of an IoT device using IoT-OM (Internet-of-Things Object Model).
To illustrate IoT-OM, we take intelligent lights as an example (see Figure 5 and Extended data14). The light has a timing switch, consisting of a timing component and a switching component. The functions are implemented by different properties, actions, and events. The timing component contains start time, current time, and duration. The current time of the device can be modified by setting time. Also, the light needs to report the current time periodically and send it to the platform. By combining these two components, a timing switch is defined. Moreover, other devices may have the same functions, which can be implemented rapidly by reusing these two components.

Before transmitting data, it is necessary to describe the device and specify the device parameters on the platform, and the defined model has to be installed on the device. After that, there is no need to transmit vast and complicated headers on the internet. Only the model values (IoT-OM (Payload) in Figure 1) need to be transmitted between devices and platforms.

Use cases
China Mobile OneNET Studio
Currently, the proposed IoT-OM has been adopted by several companies and implemented in their platforms, such as China Mobile OneNET Studio15. Similar concepts are also utilized by other companies, such as AWS IoT16, Microsoft Azure IoT17, Alibaba Cloud18, and Tencent Cloud19.

![Figure 5. The illustration of an intelligent light.](image)

The process of creating an IoT device is illustrated in Figure 6. There are two types of devices: new devices that will be added to the scenario and existing ones already planted. For both devices, the first two steps are the same: creating the product on the platform and defining the instance for the device based on IoT-OM. The third step is different. Generic SDK (Software Development Kit) will be generated for new devices based on the designated functions. For existing devices, protocol-related SDK will be created. Then these SDKs need to be installed on IoT devices to connect to IoT platforms. After the installation, these devices need to be debugged on the platforms.

In Figure 7, the data flow of OneNET platform is presented. IoT devices transmit data to the platform through CoAP (Constrained Application Protocol) or MQTT (Message Queuing Telemetry Transport) protocols, and the platform uses commands to control the devices. Also, when the application tries to control the devices or retrieve some data, it will send commands to the platform. The data gathered at the platform is used to create twin devices. After creating a twin device, it can simulate the operation and fault information so that manufacturers can debug the device on the twin device. Also, software engineers can develop software on the twin device. See Software availability for further information on creating object models using OneNET studio. This parallel collaborative development can shorten the development cycle (Table 1).

![Figure 6. The pipeline for creating a device on OneNET platform.](image)
Intelligent fire protection system

At present, China Mobile OneNET Studio has been utilized in several applications. The first application is an intelligent fire protection system. Fire protection systems have become highly complex as technology advances, including many subsystems and hundreds of devices. The monitoring and alarm system involves gas detection equipment, electrical fire monitoring devices, manual alarm equipment, sound and light alarms, etc. The fire protection system in buildings involves fire hydrants, foam extinguishers, dry powder extinguishers, smoke exhaust equipment, fire doors, fire telephones, fire power supplies, fire water level controllers, etc. For evacuation systems, it involves emergency broadcasting, emergency lighting, evacuation indicators, etc. The fire protection system outside buildings involves fire hydrants, fire manhole covers, fire pumps, fire trucks, etc. For these complex subsystems and hardware, if they can cooperate, the efficiency of the entire system will be hugely enhanced.

Traditional fire protection system has the following shortcomings. (1) The management and maintenance of fire protection facilities are not timely, and equipment may not be available when a fire occurs. (2) It is slow to detect small fires, and fires can not be noticed when unattended, which will result in damaging fires. (3) Fire fighting system is not associated with cross-platform data, i.e., the spatial structure of a building, which hinders the fire fighting operations. OneNET Studio adopts IoT-OM and realizes a fire monitoring system in a densely populated neighborhood in Guiyang, China. Figure 8 demonstrates the architecture of a smoke and temperature monitoring application. IoT devices transmit data to OneNET Studio using NB-IoT (Narrow Band Internet of Things) techniques. Then, the device information will be displayed on the PC platform instantly, and an alarm message will be promptly sent to the mobile application. Figure 9 illustrates the architecture of a fire water monitoring application. Compared with traditional fire-fighting facilities, the intelligent system can obtain equipment information accurately and instantaneously by appending IoT devices. Figure 10 shows an application of electricity safety monitoring. The device data will be sent to the platform and can be utilized for monitoring. Figure 11 presents the visual interface, and it shows the fire-monitoring devices in this area, the number of alarming devices, etc.
Figure 8. Demonstration of smoke and temperature monitoring. NB-IoT: Narrow Band Internet-of-Things.

Figure 9. Demonstration of fire water monitoring. NB-IoT: Narrow Band Internet-of-Things.

Figure 10. Demonstration of electricity safety monitoring. NB-IoT: Narrow Band Internet-of-Things.
In addition to displaying the status of a device, IoT-OM can realize control between different systems. In traditional systems, the data of a system cannot be transmitted to other systems. For example, when a fire occurs, the information of the fire monitoring system cannot be conveyed to the elevator control system. However, the IoT platform can couple different systems and provide cross-system control.

Intelligent air quality monitoring system
Air quality is increasingly important to the health of citizens, and it is of great significance to monitor the air quality of buildings. Traditional monitoring systems are implemented using wired networks, which is expensive and complicated in construction and maintenance. Since there is no need to set up a wired network, IoT techniques can reduce the difficulty of building an intelligent monitoring system. In addition, considering the large variety of equipment and manufacturers, it was laborious to connect the massive number of devices onto a platform.

By utilizing IoT-OM, OneNET Studio can connect an amount of IoT devices produced by other companies and monitor the air quality. Figure 12 illustrates the structure of the air quality monitoring system, and Figure 13 shows the devices and the visual interface of OneNET Studio.

Comparison for development cycle
Table 1 lists the production information from a selection of companies that utilize IoT-OM and who have accesses to OneNET Studio. Also, we count the number of accessed devices and the comparison of development cycles. As illustrated in the table, the development cycles for the product are hugely reduced after utilizing IoT-OM. The data is provided with the permission of these companies.

Conclusions
In this paper, we propose an object model to provide a unified description for IoT devices, which can simplify the development processes for manufacturers and software engineers. Especially, the descriptions and capabilities of devices are defined on the communication protocols. A twin device can be created using IoT-OM to debug the virtual device. Moreover, it is possible to develop applications based on the twin device. At present, China Mobile OneNET Studio has adopted the proposed IoT-OM and provided SDK to manufacturers. We counted the development cycles of five companies and uploaded the result (see DevCycles.xlsx in Extended data*). According to these statistics, manufacturers can significantly reduce development cycles. In addition, utilizing a unified description for devices makes it easier to establish the connection between different systems and improves the intelligence for cities and buildings.

The realization of digital twin relies on a large amount of real-world data, and IoT-OM provides a practical way to obtain heterogeneous data. Although virtual debugging and development have been implemented on some platforms, twin devices...
are not intelligent enough. In the future, we plan to improve the proposed model in two aspects. The first aspect is to utilize IoT-OM in more scenarios to verify the effectiveness of the model. The second aspect is to investigate how to improve the intelligence of twin devices through collected data.

**Data availability**
*Underlying data*
No underlying data are associated with this article.

**Extended data**
B2SHARE: Internet-of-Things (IoT) Object Model. [http://doi.org/10.23728/b2share.25290da7604a4211a45a8b6f3f9b98ff](http://doi.org/10.23728/b2share.25290da7604a4211a45a8b6f3f9b98ff)

This project contains the following extended data:
- night_light.json (a basic object model for IoT night light. [Visual Studio Code](https://code.visualstudio.com/) is required to review the code).
- Night Light.png (the structure of the IoT night light).

---


**Figure 13. Internet-of-Things devices for air quality monitoring from the manufacturer and the visual interface of OneNET Studio.**
- street_light.json (a basic object model for IoT street light).

Visual Studio Code is required to review the code.

- Street Light.png (the structure of the IoT street light).

- DevCycles.xlsx (the information of IoT devices, number of accessed devices, and development cycles listed in Table 1).

Data are available under the terms of the Creative Commons Zero “No rights reserved” data waiver (CC0 1.0 Public domain dedication).

Software availability

The source code for creating the object model on the OneNET website is not publicly available due to the nature of the proprietary software license, and the copyright is owned by China Mobile Communications Corporation. For readers who wish to use IoT-OM and define IoT devices using OneNET platform, please send an access request email to the corresponding author (longrong@chinamobile.com) including a detailed description of the purpose of usage and scenario. If readers agree to only use the account for research purposes and without redistributing the associated software/code, We will send readers a trial account of OneNET platform.

The OneNET platform provides an open-source Software Development Kit (SDK) here: https://open.iot.10086.cn/doc/iot_platform/book/device-connect&manager/sdk.htm. The SDK can be used for software development of IoT applications and is under BSD license. The SDK is installed on the devices that handles communications, and describes the devices.

References


