Huawei IoT Certification Training

# HCIA-IoT

# Trainee Guide for IoT Engineers

ISSUE: 2.5

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

**Huawei Technologies Co., Ltd.**

| | |
|---|---|
| Address: | Huawei Industrial Base Bantian, Longgang Shenzhen 518129<br>People's Republic of China |
| Website: | http://e.huawei.com |

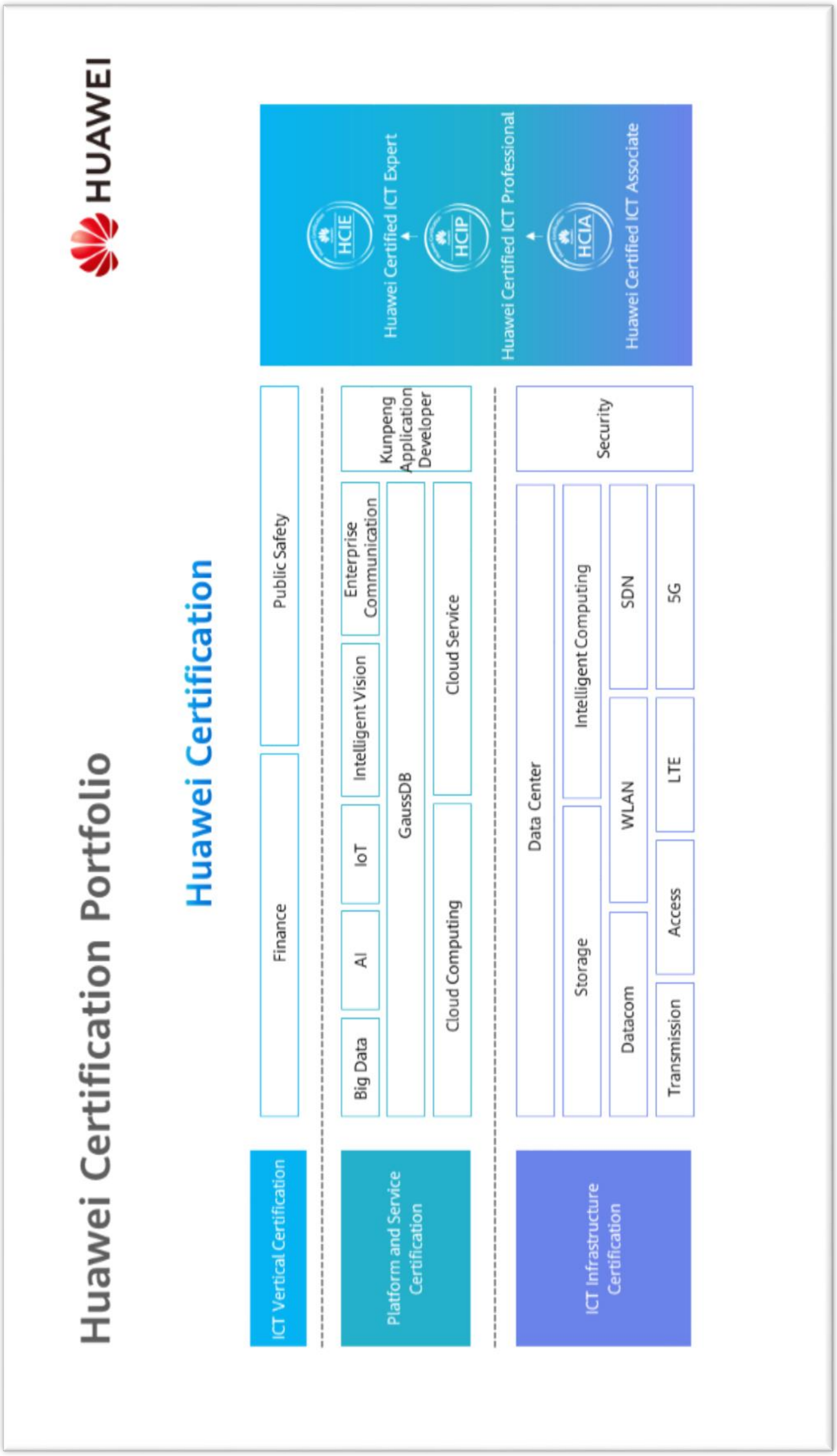
## Huawei Certification System

Huawei Certification follows the "platform + ecosystem" development strategy, which is a new collaborative architecture of ICT infrastructure based on "Cloud-Pipe-Terminal". Huawei has set up a complete certification system consisting of three categories: ICT infrastructure certification, platform and service certification, and ICT vertical certification. Huawei provides the only all-range technical certification in the industry.

Huawei Certification offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE). Huawei Certification covers all ICT fields and adapts to the industry trend of ICT convergence. With its leading talent development system and certification standards, it is committed to fostering new ICT talent in the digital era, and building a sound ICT talent ecosystem.

Huawei Certified ICT Associate-Internet of Things (HCIA-IoT) is designed for Huawei IoT product technical personnel (including university students or IoT practitioners) and frontline engineers in Huawei offices and representative offices. HCIA-IoT covers the HUAWEI CLOUD IoT platform, Huawei LiteOS, and IoT communications technologies (wireless communications technologies and IoT gateway technologies).

The HCIA-IoT certificate system introduces you to the industry and market, helps you innovate, and puts you at the forefront of the IoT.



## Huawei Certification Portfolio

HUAWEI

**Huawei Certification**

ICT Vertical Certification
- Finance
- Public Safety

Platform and Service Certification
- Big Data
- AI
- IoT
- Intelligent Vision
- Enterprise Communication
- Kunpeng Application Developer
- GaussDB
- Cloud Computing
- Cloud Service

ICT Infrastructure Certification
- Data Center
- Security
- Storage
- Intelligent Computing
- Datacom
- WLAN
- SDN
- Transmission
- Access
- LTE
- 5G

HCIE Huawei Certified ICT Expert

HCIP Huawei Certified ICT Professional

HCIA Huawei Certified ICT Associate

# About This Document

## Overview

Driven by the global informatization tide, China's computer industry has developed rapidly in recent years. Various emerging technologies have become increasingly mature. The Internet of Things (IoT) has become an integral part of our lives. Things like smart watches, smart bands, vending machines, and smart vehicles are all powered by IoT technology.

Many traditional industries, such as electric power and transportation, are also leveraging IoT technology. In addition, IoT is being used together with emerging technologies such as artificial intelligence (AI) to form AIoT, which has become popular. Whether you are from an ICT industry or a traditional one, learning and understanding IoT technology is an important means to innovate solutions.

HCIA-IoT V2.5 trains and certifies engineers to develop end-to-end IoT services based on the Huawei IoT solution architecture.

## Description

This guide is used together with the HCIA-IoT V2.5 courses. It supplements course slides and adds related knowledge points.

This guide is divided into 13 chapters corresponding to the 13 courses slides that are based on the IoT architecture. The following is a summary of each chapter:

- Chapter 1 describes the concept and development history of the IoT, layers of the IoT architecture, and Huawei's IoT solution.
- Chapter 2 describes IoT industry applications and solutions, covering smart city, smart campus, smart grid, IoV, and industrial IoT scenarios. It also describes the difficulties facing each industry and the transformation and benefits brought by IoT technology. In addition, it explores the combination of the IoT with six emerging technologies and describes the development trends of the IoT industry.
- Chapter 3 describes the security challenges faced by the IoT industry, and introduces Huawei's IoT security solutions and related applications.
- Chapter 4 describes common network communications technologies of the IoT based on the solutions described in chapter 2.
- Chapter 5 describes key technologies and corresponding solutions of the widely used NB-IoT communications technology in the IoT field.
- Chapter 6 describes the popular communication technology 5G, including its development process, core technologies, application scenarios, and commercial solutions.
- Chapter 7 describes difficulties and challenges faced by IoT gateways in industrial scenarios, edge computing, and mesh networking.

- Chapter 8 describes capabilities of home gateways, as well as the Huawei HiLink platform and corresponding solutions.
- Chapter 9 describes the positioning of the IoT platform in the IoT solution, capabilities required by the IoT industry, and the services and capabilities of the HUAWEI CLOUD IoT platform.
- Chapter 10 describes the basic knowledge required for IoT platform development and the secondary development process based on the HUAWEI CLOUD IoT platform.
- Chapter 11 describes the basic knowledge and development history of operating systems, the role of the IoT operating system in the IoT industry, problems faced by the IoT operating system and IoT devices, and Huawei LiteOS.
- Chapter 12 describes experimental theories. It describes basic knowledge about the single-chip microcomputer and sensor technologies required for device development, and knowledge about the kernel, framework, and APIs of Huawei LiteOS.
- Chapter 13 describes AT commands used in communications module interconnection testing and the network interconnection process using communications modules.

This guide also provides links to related knowledge for HCIA-IoT learners. You do not need to master all of the knowledge related to this course in this guide because of the difficulties, but we encourage you to engage in further learning on your own.

Now, let's begin to learn about the IoT.

# Contents

**About This Document .......... 3**
Overview .......... 3
Description .......... 3
**1 What Is Internet of Things? .......... 12**
1.1 Origin of the IoT .......... 12
1.2 IoT Architecture .......... 12
1.3 Huawei IoT Solution .......... 13
1.3.1 Platform Layer .......... 14
1.3.2 Network Layer .......... 14
1.3.3 Sensing Layer .......... 14
**2 IoT Industry Applications and Solutions .......... 16**
2.1 Smart City Solution .......... 16
2.1.1 Common Problems .......... 16
2.1.2 Solutions .......... 18
2.2 Smart Campus Solution .......... 19
2.2.1 Common Problems .......... 19
2.2.2 Solutions .......... 20
2.3 Smart Grid Solution .......... 21
2.3.1 Common Problems .......... 21
2.3.2 Solutions .......... 22
2.4 IoV Solution (IoV and DRIS) .......... 22
2.4.1 Common Problems .......... 22
2.4.2 Solutions .......... 23
2.5 Industrial IoT Solution .......... 24
2.5.1 Development Bottlenecks .......... 24
2.5.2 Solutions .......... 25
2.6 Development Trend of IoT Applications .......... 26
**3 IoT Security Technologies .......... 28**
3.1 Typical IoT Security Cases .......... 28
3.2 Huawei IoT Security Architecture .......... 29
3.3 Typical Huawei IoT Security Cases .......... 31
**4 Common IoT Communications Technologies .......... 32**
4.1 Wired Communications Technologies .......... 32
4.1.1 Ethernet .......... 32

4.1.2 RS-232 and RS-485 ....33
4.1.3 Communication Serial Port Bus ....34
4.1.4 M-Bus ....34
4.1.5 PLC ....34
4.1.6 Comparison of Wired Communications Technologies ....35
4.2 Wireless Communications Technologies ....35
4.2.1 Short-Range Wireless Communications Technologies ....35
4.2.1.1 Bluetooth ....35
4.2.1.2 Wi-Fi ....36
4.2.1.3 ZigBee ....37
4.2.1.4 Z-Wave ....37
4.2.1.5 Comparison of Short-Range Wireless Communications Technologies ....38
4.2.2 Cellular Mobile Communications Technology ....39
4.2.2.1 2G ....39
4.2.2.2 3G ....39
4.2.2.3 4G ....39
4.2.2.4 5G ....40
4.2.2.5 Comparison of Cellular Mobile Communications Technologies ....40
4.2.3 LPWA Communications Technologies ....41
4.2.3.1 Sigfox ....41
4.2.3.2 LoRa ....41
4.2.3.3 NB-IoT ....41
4.2.3.4 eMTC ....41
4.2.3.5 Comparison of LPWA Technologies ....42
4.2.4 Wireless Communications Technologies ....43
**5 NB-IoT Communications Technologies and Solutions ....45**
5.1 NB-IoT Standards Evolution and Industry Development ....45
5.1.1 Evolution of NB-IoT Standards ....45
5.1.2 LPWA Technologies Adopted by Global Operators ....46
5.1.3 NB-IoT Spectrum Selections of Global Operators ....46
5.1.4 NB-IoT Industry Development ....47
5.2 Key NB-IoT Technologies ....48
5.2.1 Ultra-Low Cost ....49
5.2.1.1 NB-IoT Deployment Modes ....49
5.2.1.2 Chip Design ....50

5.2.2 Ultra-Low Power Consumption ....50
5.2.3 Ultra-Wide Coverage ....51
5.2.4 Massive Connections ....52
5.3 NB-IoT Solutions ....52
**6 5G Communications Technologies and Solutions ....55**
6.1 5G Standards Evolution and Industry Development ....55
6.2 Key 5G Technologies ....57
6.2.1 New Architecture ....58
6.2.1.1 Network Slicing ....58
6.2.1.2 NSA&SA Networking ....59
6.2.1.3 5G Core Network (NGC) ....59
6.2.2 New Radio ....60
6.2.3 Full Spectrum ....60
6.3 Three 5G Application Dimensions ....61
6.3.1 eMBB ....61
6.3.2 mMTC ....61
6.3.3 URLLC ....61
6.4 5G Commercial Solutions ....62
6.4.1 B2C Solutions ....62
6.4.2 B2B Solutions ....63
6.4.3 B2H Solutions ....64
**7 Huawei Industrial IoT Gateways ....66**
7.1 **Overview of Industrial IoT Gateways** ....66
7.1.1 **Challenges Faced by IoT in the Industrial Field** ....66
7.1.2 **Why Industrial IoT Gateways?** ....67
7.2 **Edge Computing** ....67
7.3 **Network Topology** ....69
7.3.1 **Star Topology** ....70
7.3.2 **Ring Topology** ....71
7.3.3 **Bus Topology** ....71
7.3.4 **Mesh Topology** ....72
**8 Home IoT Gateways ....74**
8.1 Home IoT Gateway Overview ....74
8.1.1 Home Network Development ....74
8.1.2 Positioning of Smart Home Gateways ....75
8.1.3 Key Services of Smart Home Gateways ....76
8.2 Huawei Smart Home Solution ....77
8.3 HiLink Platform ....78

**9 IoT Platform........................................................................80**
9.1 Overview of Ecosystem and Platform........................................80
9.1.1 Platform Model........................................................................80
9.1.2 Ecosystem...............................................................................80
9.2 Origin of the IoT Platform..........................................................81
9.2.1 IoT Is Ushering In Industry Innovation and Transformation...........81
9.2.2 Major Challenges Facing the IoT Industry..................................82
9.2.3 Requirements for the IoT Platform............................................83
9.3 Introduction to the HUAWEI CLOUD IoT Platform.......................84
9.3.1 IoT Platform Overview..............................................................84
9.3.2 IoT Device Access....................................................................88
9.3.2.1 Overview...............................................................................88
9.3.2.2 Typical IoT Protocols..............................................................89
9.3.3 Device Management.................................................................93
9.3.3.1 Device Registration and Access Authentication........................93
9.3.3.2 Command Delivery.................................................................93
9.3.3.3 Device Linkage Rules.............................................................96
9.3.3.4 Firmware and Software Upgrades...........................................97
9.3.3.5 Batch Operations...................................................................98
9.3.4 Data Forwarding......................................................................98
9.3.5 Application Integration.............................................................99
9.3.5.1 Application-side RESTful APIs................................................99
9.3.5.2 Application Registration Authentication..................................100
9.3.5.3 Subscribe/Push....................................................................100
9.3.6 IoT Analytics..........................................................................101
9.3.7 IoT Studio..............................................................................107
9.3.8 Other Related Services...........................................................108
9.3.8.1 Edge Computing..................................................................108
9.3.8.2 Global SIM Link...................................................................110
9.3.9 Characteristics of the HUAWEI CLOUD IoT Platform..................111
9.3.9.1 Access Agnostic...................................................................111
9.3.9.2 Reliable..............................................................................111
9.3.9.3 Secure................................................................................113
9.3.9.4 Scalable.............................................................................113
9.3.9.5 Open..................................................................................114

**10 IoT Platform Secondary Development ... 115**
10.1 Introduction to Platform Secondary Development ... 115
10.2 Product Development ... 115
10.2.1 Product Model ... 117
10.2.1.2 Product Model Example: Smart Water Meter ... 118
10.2.2 Codec ... 119
10.2.2.1 Overview ... 119
10.2.2.2 Data Reporting ... 120
10.2.2.3 Command Delivery ... 122
10.3 Development on the Application Side ... 124
10.3.1 Calling APIs ... 125
10.3.1.1 Making an API Request ... 125
10.3.1.2 Authentication ... 128
10.3.1.3 Returning a Response ... 130
10.3.2 API ... 131
10.3.2.1 Creating a Device ... 131
10.3.2.2 Querying a Device ... 132
10.3.2.3 Modifying Device Information ... 132
10.3.2.4 Querying Device Messages ... 133
10.3.2.5 Deleting a Device ... 133
10.4 Development on the Device Side ... 134
**11 IoT OS ... 140**
11.1 Preliminary Study of the OS ... 140
11.1.1 Concept ... 140
11.1.2 History ... 141
11.1.3 Development ... 143
11.2 Challenges to the IoT OS ... 143
11.2.1 IoT Terminal Development ... 144
11.2.2 IoT OS Development ... 144
11.3 Huawei LiteOS ... 144
11.3.1 Introduction to Huawei LiteOS ... 144
11.3.2 Commercial Use of Huawei LiteOS ... 145
11.3.3 Huawei LiteOS Industry Applications ... 145
**12 Sensing Layer Development ... 147**
12.2 Hardware Development at the Sensing Layer ... 148
12.2.1 Single-Chip Microcomputer Basics ... 148

12.2.1.1 Microcomputer Working Principles ........ 148
12.2.1.2 Instruction Set ........ 149
12.2.1.3 Processing Units ........ 151
12.2.1.4 Architecture of the Single-Chip Microcomputer ........ 151
12.2.1.5 ADC and DAC ........ 152
12.2.2 Sensing Technology Basics ........ 152
12.2.2.1 Photoelectric Sensor ........ 153
12.2.2.2 Temperature Sensor ........ 153
12.2.2.3 Acceleration Sensor ........ 154
12.2.2.4 Heart Rate Sensor ........ 155
12.3 Software Development at the Sensing Layer ........ 155
12.3.2 Huawei LiteOS Kernel ........ 156
12.3.2.2 Task ........ 156
12.3.2.3 Memory ........ 161
12.3.2.4 Interrupt ........ 164
12.3.2.5 Queue ........ 164
12.3.2.6 Event ........ 165
12.3.2.7 Semaphore ........ 166
12.3.2.8 Mutex ........ 166
12.3.2.9 Time Management ........ 167
12.3.2.10 Timer ........ 167
12.3.3 Huawei LiteOS Framework ........ 168
12.3.4 Huawei LiteOS APIs ........ 169
**13 AT Commands of the Communication Module ........ 170**
13.2 Introduction to AT Commands ........ 171
13.2.1 AT Command Types ........ 171
13.2.2 NB-IoT AT Commands ........ 171
13.2.3 Wi-Fi AT Commands ........ 172
13.2.4 AT Commands for Huawei-Certified Modules ........ 173
13.3 Terminal-side Development for Communication Modules ........ 174
13.3.1 NB-IoT Terminal Interconnection Process ........ 174
13.3.2 Wi-Fi Terminal Interconnection Process ........ 175
**14 Conclusions ........ 176**
**15 References ........ 178**

**16 Appendix ........................................ 179**

# 1 What Is Internet of Things?

## 1.1 Origin of the IoT

When we hear about Troy, we may think of the Trojan horse. In the IoT industry, there is also a well-known event related to Troy, the Trojan room coffee pot event. This event, which took place at the Trojan Room of the Computer Laboratory in Cambridge University in 1991, can be said to be the origin of the IoT.

The Auto-ID Center in America first proposed the concept of the IoT in 1999 based on radio-frequency identification (RFID) and Electronic Product Code (EPC). At the 2005 World Summit on the Information Society, the International Telecommunication Union (ITU) discussed the concept of the IoT and revised its definition. In 2009, then Chinese Premier Wen Jiabao proposed the idea of "Sensing China" in Wuxi. This could be regarded as the beginning of the IoT enlightenment in China. In 2015, Chinese Premier Li Keqiang signed and released *Made in China 2025*, which can be seen as representative of China's IoT policies.

Today, the IoT has been widely used in our daily lives. IoT applications can be classified into three types: consumption-driven, policy-driven, and industry-driven.

Consumption-driven applications align with our daily lives. They are used in products such as smart bands, VR devices, smart speakers, and shared bicycles.

Policy-driven applications are a series of IoT applications promoted by government agencies, helping them better manage cities and provide better public services. These applications include smart security, smart city management, and smart street lamp management applications.

Industry-driven applications are mainly oriented to customers for commercial use. Customers directly use IoT technology in business, such as smart factories. They can collect production data from robotic arms or machine tools and analyze it to maximize its use. This type of applications is also used in IoV, smart logistics, and smart agriculture.

## 1.2 IoT Architecture

The IoT is an Internet where all things are interconnected, which contains two meanings. One is that the essence of the IoT is an Internet. The IoT is a technology designed based on Internet technologies. The other meaning is that things connected to the Internet are not just our computers and mobile phones. They can be a variety of things around us, such as air conditioners and refrigerators. IoT technology aims to help us connect all things to the Internet.

Since the advent of the Internet, people have become closer to each other. The IoT connects everything. In the era of information explosion, what we see on the Internet can be described as data. We can say that the IoT is actually a data service technology. To build such an IoT world, the connected devices need to be smart enough to understand our needs and provide us with more intelligent services.

The IoT architecture consists of four layers: application, platform, network, and sensing.

The sensing layer is the layer of things. It contains various types of sensors, chipsets, and modules. This layer is used to collect data, process signals, and communicate with and send data to the platform layer using a communications module.

The network layer is equivalent to a communications medium between the sensing layer and the platform layer. We can use a wireless network, such as GPRS, NB-IoT, or 4G, or use wired access, such as gateways, to transmit data to the platform layer.

The platform layer is responsible for storing, using, and maintaining the data. We can use the obtained data for data processing and other operations.

The application layer is oriented to customers. It is responsible for presenting the collected data specifically.

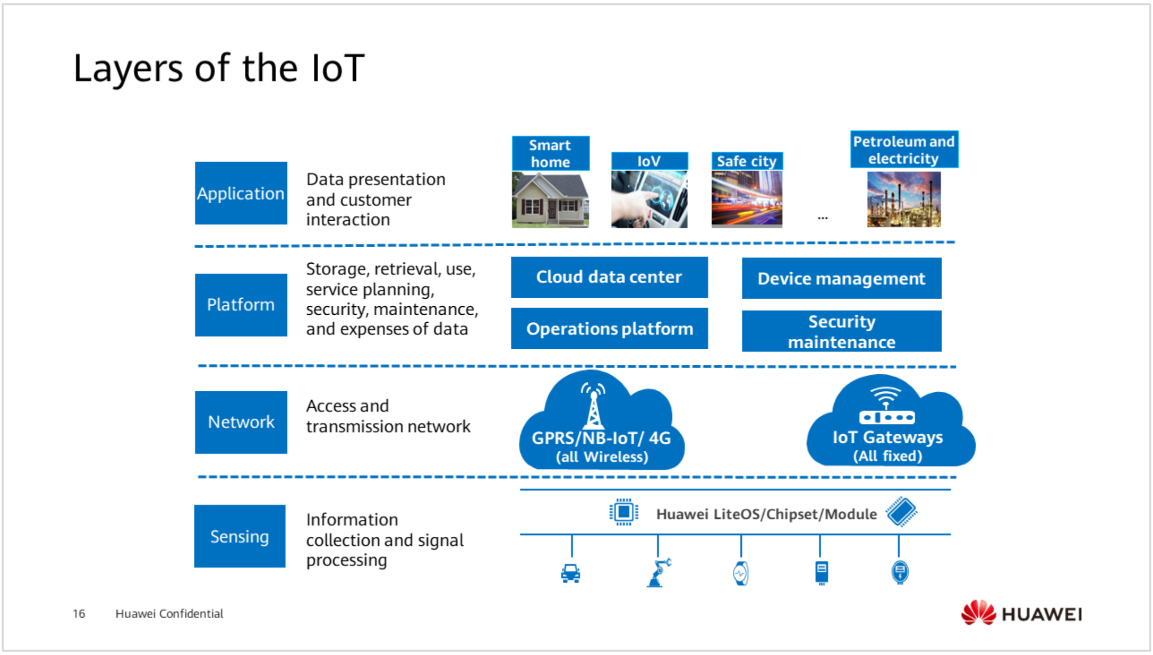

Layers of the IoT
Application
Data presentation and customer interaction
Smart home
IoV
Safe city
...
Petroleum and electricity
Platform
Storage, retrieval, use, service planning, security, maintenance, and expenses of data
Cloud data center
Device management
Operations platform
Security maintenance
Network
Access and transmission network
GPRS/NB-IoT/ 4G (all Wireless)
IoT Gateways (All fixed)
Sensing
Information collection and signal processing
Huawei LiteOS/Chipset/Module
16
Huawei Confidential
HUAWEI

**Figure 1-1 Layers of the IoT**

## 1.3 Huawei IoT Solution

This section describes Huawei's full-stack IoT solution to meet IoT requirements. Huawei provides full-stack products from the sensing layer to the platform layer to help developers quickly develop and update products. The solution mainly includes Huawei LiteOS, Huawei network solutions, and HUAWEI CLOUD IoT platform.

### 1.3.1 Platform Layer

The top layer of the Huawei IoT solution is the HUAWEI CLOUD IoT platform. The platform layer is located in the middle of the IoT architecture. It has an application layer above it, and the network and sensing layers below it. It serves as a bridge between the upper and lower layers. The IoT platform is divided into two layers. The upper layer is the service enablement layer, which provides services for a wealth of applications. The lower layer is the connectivity management layer. It is used to connect IoT devices to the platform layer and manage them, and is used for authentication.

### 1.3.2 Network Layer

Under the platform, there are two network access modes. The first mode is wireless access. NB-IoT is an end-to-end solution. It is mainly proposed by Huawei and Vodafone, and Huawei is actively engaging in NB-IoT, so NB-IoT can be regarded as an IoT solution of Huawei. Various devices are connected to base stations through a series of protocols, transfer data to the IoT platform using core networks, and provide data to vertical industry applications. In this process, NB-IoT is mainly used to transfer data.

5G supports a large number of industries. It divides the entire network into different slices based on users' network requirements. These slices have different attributes. For example, UHD slices have the attributes of high bandwidth and high transmission rate, and real-time service slices feature low latency. Based on this, different network requirements of different industries can be implemented on the same 5G network.

In addition to the preceding wireless access technologies, Huawei provides a series of access solutions for the fixed access mode, such as industrial gateways used in harsh industrial environments. These industrial gateways have the edge computing capability to meet industrial requirements for low-latency near-end processing.

For homes, Huawei also provides smart gateways. With them, customers can remotely control and access the cloud platform through a mobile phone to deliver commands. Then, the smart devices connected to smart gateways are controlled based on the commands to meet customer requirements. In addition, smart gateways can monitor smart home devices in real time. If a fault occurs, they report it to customers and handle it by themselves.

### 1.3.3 Sensing Layer

In addition to its own platform and gateways, Huawei also has its own operating system, Huawei LiteOS, which is located at the sensing layer. Huawei LiteOS is a lightweight, intelligent operating system that features low power consumption and fast response. The architecture of the operating system is described as a "1+N" architecture. "1" means that the operating system has only one kernel, and "N" means there are N pieces of middleware. The following takes Iron Man/Tony Stark as an example to better understand this architecture. The kernel can be considered as Tony Stark. Then, different middleware can be added for different scenarios. The middleware is similar to different components of Iron Man suits used to handle different situations. Huawei LiteOS is designed to shorten the time to market (TTM) of smart devices. In addition to being applicable to multiple Huawei products, Huawei LiteOS supports connection to products and platforms of partners in the industry, greatly shortening the development time of the IoT industry.

# 2 IoT Industry Applications and Solutions

This chapter describes problems encountered in the domains of smart city, smart campus, smart grid, IoV, and industrial IoT, and benefits of using IoT technology. It also introduces the development trends of the IoT based on other popular technologies.

## 2.1 Smart City Solution

Smart city is a big topic because a city contains many elements, which can create a plethora of problems. This section describes a few common problems encountered in smart city and the solutions.

### 2.1.1 Common Problems

The first problem is traffic management, which is critical to urban development. A highly developed city must have a correspondingly developed transportation system. To make a city better, transportation is one of the first issues to address. In cities, traffic accidents and congestion occur often as a result of drivers disobeying traffic rules. This forces other drivers to waste more time on the road. In addition, drivers brake frequently due to traffic jams, reducing the energy efficiency of gas-powered vehicles and increasing pollution. City managers need to find an appropriate solution to traffic congestion and alleviate the pollution.

Parking problems can be described from two aspects: parking management and experience. First, in terms of management, most parking space details are not clear enough. Navigation apps prompt you about parking lots near your destination, but they cannot tell you whether there are available parking spaces. When you arrive, you may find that the parking lot is full.

The tidal effect in parking lots is also a problem. The parking lot is congested once in the morning and once at night. For example, when you go to work in the morning, you drive cars from parking lots near home and park them in your companies' parking lots. In the morning, parking spaces near home are unused. When you go home in the evening, parking spaces in companies become unused. So there's a time-based imbalance for these parking spaces.

In addition, for parking space managers, charging and inspection problems also exist. They can centrally manage parking spaces at the entrances and exits of shopping malls. But they may have trouble charging roadside parking spaces. Another problem is illegal parking. Traffic police give parking tickets to vehicles that violate parking rules, but this is a waste of manpower and resources.

In terms of parking experience, drivers cannot find idle parking spaces because it is difficult to obtain parking space details. It is also difficult to find cars in parking lots. For example, you park your car in the large parking lot of a shopping mall, but when you finish shopping and head back to your car, you cannot find it. Congestion at the entrances and exits of parking lots during peak hours is also a problem. These are the parking problems we may encounter.

Street lamp issues also exist in cities. Currently, most street lamps in cities are controlled by time-based rules. When the sky is almost dark, street lamps are turned on. When the sun comes out in the morning, street lamps are turned off. However, the time when the sky turns dark varies seasonally. Lights come on too early in the summer and too late in the winter. In these cases, street lamps controlled by time waste electricity.

There are also special cases, for example, solar eclipses. Although a solar eclipse occurs in the daytime, the sky is as dark as at night. Street lamps controlled by time cannot be turned on for emergencies. Problems also occur when street lamps are faulty and need to be repaired. Maintenance is often performed in the morning. When street lamps are not turned on in the morning, maintenance personnel need to check them one by one to locate faulty ones, which is time- and labor-consuming. If maintenance personnel do not perform maintenance in time, faults may not be detected. If a technology can be used to monitor the running status of each street lamp in real time and report an error once a street lamp becomes faulty, maintenance will be more convenient.

Energy conservation is a problem, too. For example, in the second half of the night, there are fewer vehicles on the road. Leaving street lamps on is a waste of energy. A relatively intelligent system is required to automatically detect pedestrians or vehicles on the road and adjust the brightness of street lamps.

The next problem is the revenue of lamp pole resources. There used to be many utility poles on the road, but now they can hardly be seen. This forces us to seek more revenue from rare pole position resources. For example, we can install surveillance cameras and 5G base stations on street lamp poles. We can also install charging piles for new energy vehicles under street lamp poles. In this way, we can maximize the utilization of pole position resources. These are the street lamp problems that we may encounter.

Firefighting management also has problems. In our cities, firefighting is still an issue that we need to pay attention to, because fires can cause great damage to our personal and public property. Small public areas in cities are the most prone to fires. Why? First, these areas are small and things are placed haphazardly. Flammable materials are likely to be dangerously placed. Second, in older places, electric cables are aged. Cable aging is a more serious threat in old restaurants, because they use more electricity. In addition, electricity piracy is severe in these places, which increases fire risks. To sum up, fire hazards are prominent, fire facilities are weak, and fire warnings are not responsive.

Manhole cover management is also an obvious problem in cities. There are a large number of manhole covers, which belong to different bureaus, such as water, communications, and gas bureaus. Therefore, quantity and ownership issues make managing these manhole covers difficult. Which bureau is responsible for managing these manhole covers? How can they manage a large quantity of manhole covers? Another problem is theft and loss. Manholes with missing covers pose a serious safety risk to drivers and pedestrians. In addition, the sewage pipes under manhole covers may contain toxic gas, which can float upwards from manholes with missing covers, posing additional public health and safety risks. Manhole cover management is a significant issue that requires the coordination of multiple bureaus.

Similarly, there are problems with sanitation management in cities. We can roughly divide the problems into four directions: outdated, low, disordered, and slow. First, the sanitation bureau does not collect information about public garbage bins on the road. The management mode of regular clearing is outdated. Second, sanitation is mainly managed by people. Human factors seriously impede sanitation management, causing problems such as command and dispatch difficulties. Third, sanitation bureaus are often averse to using new technologies. They are still using conventional methods to manage urban sanitation. The sanitation industry lacks innovation and develops slowly.

## 2.1.2 Solutions

Various problems in different aspects of city management hinder the development of cities. City managers need to find better ways to solve these problems. Huawei has designed a comprehensive smart city solution to solve problems related to transportation, parking, street lamps, firefighting, manhole covers, and sanitation.

The smart city solution corresponds to the architecture of the IoT. At the sensing layer, there are a large number of IoT devices, such as surveillance cameras on roads, gateways that control various sensors, and low-power consumption devices such as electricity meters and water meters. We can select the most appropriate network technologies for different IoT devices. For example, we can use the wireless 2G, 3G, 4G or NB-IoT or wired network to connect devices to the IoT platform. After the devices are managed on the IoT platform, data can be uploaded to the platform database and analyzed to solve problems. At the top are the solutions for different smart city applications.

The advantages of the smart city solution lie in three aspects: lowering the development threshold of IoT applications, reducing the development difficulty for developers, and shortening the incubation time of the entire IoT city ecosystem. In this way, a smart city ecosystem can be quickly implemented. The platform can aggregate information from all aspects of cities and provide more convenient services for comprehensive city management. For devices and networks, this solution adapts devices and standards used and provides a unified management model for city managers, avoiding fragmented access of IoT applications to the platform.

Detailed solutions to some of the problems described earlier are provided. The first is the intelligent transportation solution. This solution uses application systems such as e-Police and signal control to control traffic in real time. It uses sensors such as cameras on roads to remotely manage city roads and relieve traffic pressure. In addition, it uses electronic and automatic patrolling methods to handle vehicles violating traffic laws and regulations, thereby reducing congestion caused by the handling of traffic violations, reducing the time spent on roads, and reducing pollution.

The smart parking solution is similar to the smart city solution. The IoT devices used in the smart parking solution are geomagnetic vehicle detectors. A geomagnetic vehicle detector can be placed or buried under each parking space. The detector can detect the change in the geomagnetic field to determine whether a vehicle is parked. If these devices are connected to the Internet, they can also tell drivers where there are idle parking spaces. In addition to detection, they can be used as collectors of parking information. The wireless technology NB-IoT is used to connect devices to the platform because NB-IoT applies to low power wide coverage scenarios. The smart parking solution not only obtains information of many parking lots, but also enables drivers to use self-service charging, reducing parking management costs.

In the smart street lamp solution, smart street lamp control devices, including various sensing components such as photosensitive sensors, are deployed at the sensing layer. At the network layer, a wide range of street lamps are connected to the Internet through a carrier's wireless WANs or a self-established mesh network such as ZigBee. Then, the information is collected to the platform, and the platform interconnects with the public cloud to manage and centrally monitor resources. As mentioned above, street lamps occupy rare pole position resources in cities. In addition to being used for lighting, street lamp poles can also be used for other purposes.

In the smart firefighting solution, intelligent smoke sensors are used. Smoke sensors can automatically report alarms when a fire occurs, helping us control fires on time. This solution uses networks such as NB-IoT to transmit data, so the data can be uploaded to the platform for processing. This solution also uses applications such as alarm handling, remote muting, device self-check, and fire prevention linkage for fire management.

In the smart manhole cover solution, sensors such as accelerometers are installed under manhole covers to monitor manhole data. These sensors are triggered when the manhole cover is moved. When manhole covers are taken away, toxic gases in the sewers may float up to the street. Therefore, gas detection sensors need to be placed near manhole covers. The smart manhole cover solution is similar to the smart street lamp solution. It can monitor a large number of manhole covers and implement intelligent O&M.

The last solution is the smart sanitation solution. This solution manages sanitation personnel, vehicles, and garbage collection in real time. It supervises the working time, routes, fuel consumption, and water spraying of sanitation vehicles, the status of garbage bins, and the tracking of garbage collection in real time. In addition, real-time visualized supervision refines sanitation management, standardizes operations, and makes appraisal intelligent, making sanitation work orderly, effective, and controllable.

## 2.2 Smart Campus Solution

### 2.2.1 Common Problems

Smart campus is similar to smart city, because the IoT devices used by them are almost the same. However, they are still two different scenarios.

Campuses are common in our daily lives. For example, Disneyland parks, zoos, industrial parks of large enterprises, and technology parks. These campuses face a large number of challenges.

In terms of management and operations, most traditional campuses are managed manually. However, manual management is ineffective, crude, and prone to error. Some unrelated personnel may be able to enter a campus due to the carelessness of the security personnel. A large number of human resources are required to handle complex tasks, which increases operational costs. These problems require campuses to become more intelligent.

The first common problem that campuses meet is visitor management problem. Let's take Huawei as an example. After arriving at a Huawei campus, visitors need to register and obtain visitor cards at a guardhouse and wait for employees to accompany them. This is time-consuming. 6,300 working days are estimated to be wasted in a year. Similarly, employees need to swipe their cards at entrances and register when they forget their cards. This wastes 7,800 working days in a year. Intelligent management and control methods are urgently needed to help Huawei save labor costs.

Power is wasted in traditional campuses because lighting, fresh air, and air conditioning systems are managed and controlled by facilities and property management personnel. Electricity is wasted whenever someone forgets to turn off the lights or the air-conditioner, or when someone turns on a light during the daytime instead of opening curtains to let in sunlight. Power consumption accounts for about 60% of the total operational cost of Huawei's campuses.

The preceding problems can be categorized into three major types. The first is high operational costs of traditional campuses. The second is traditional campuses are mainly managed by people. The last is poor employee satisfaction with campus services due to the lack of a well-developed management system to coordinate various parts. Due to lack of intelligent management, problems can only be resolved reactively after they occur.

### 2.2.2 Solutions

To solve these problems, Huawei has designed an overall solution for campuses. The overall architecture of the solution is divided into three parts. At the sensing layer are various devices in smart campuses. The devices transmit data to the management platform. The platform processes and analyzes the data and uploads it to the cloud to provide services for users.

For visitor management, a visitor can make a reservation in advance. The visitor will then receive an SMS containing a verification code. The visitor can show the verification code to a security guard and obtain a temporary pass to visit the campus. In this way, the visitor can save a lot of time.

As we mentioned, energy costs account for nearly 60% of the total cost. Therefore, energy control is an important part of campus management. In this regard, Huawei's solution is divided into four parts: power generation, distribution, utilization, and management. In a smart campus, in addition to the mains, the solution uses PV to generate more power. This method not only saves costs but also is environmentally friendly. In terms of power distribution, the solution uses uninterruptible power supply (UPS) to achieve a balanced power supply during peak and off-peak hours, thereby reducing power costs.

For power utilization, the solution uses sensors to monitor environmental data. For example, sensors can first check whether there are people in the room. If there are, the lights and air conditioner are turned on. Then, sensors can check the temperature and humidity and adjust the air conditioner to control the temperature and humidity. At the same time, the solution can use smart meters to check overall energy usage and control energy use. Last, in terms of power management, the solution can analyze the uploaded data and design an appropriate solution to manage and control energy. This improves energy utilization and reduces operational costs.

## 2.3 Smart Grid Solution

The whole process of electric power supply consists of six steps, from power generation in a power plant to power consumption by users. The first step is power generation, which is the production of electric power. Currently, the main power stations are thermal power stations, nuclear power stations, and hydropower stations. The subsequent step, power transformation, is a preparation for transmission by increasing or decreasing the voltage. The higher the voltage, the smaller the transmission loss. Therefore, the voltage is increased before transmission to reduce transmission loss. Later, before the electric power is transmitted to each home, another power transformation is used to reduce the voltage. The standard AC voltage in China is 220 V, and in some countries or regions, it is 110 V, but the voltage for power transformation and transmission is at the kilovolt level. Therefore, the voltage is decreased before the electric power is distributed to every home or factory through the distribution system. After that, people can use electricity. To sum up, the entire process of the electric power supply is power generation, transformation, transmission, transformation, distribution, and consumption.

### 2.3.1 Common Problems

The difficulties and pain points facing electric power companies may be rare in China. They may occur in regions outside China, such as Africa and Latin America. First, the line loss is high, because they do not have advanced technologies to control the line loss rate. In countries such as Japan and South Korea, their line loss rate was decreased to 4% to 5% as early as 12 years ago. In countries with large areas such as China and the United States, the line loss rate has been decreased to 7% to 8%. But in some countries and regions, the line loss rate is as high as 18% or 20%. It means that electric power companies in regions with high line loss rate make 10% or more fewer profits than those in regions with low line loss rate.

In addition, the cost and cycle of meter reading are problems. In Africa and Latin America, electric power companies mainly use manual meter reading, which is costly and error-prone.

Load balancing is also a problem, since there is no way to store electricity on a large scale in the world. If electricity is produced but not used, it is wasted. Therefore, if the power consumption and generation capacity of the area cannot be accurately estimated, it is easy to cause a waste of power resources.

In addition to unbalanced power grid development in different regions, electric power companies in different countries and regions raise different requirements. For example, countries and regions in Africa and Latin America do not have the same comprehensive infrastructure as China, North America, and Europe. Therefore, they may want to build infrastructure first. They also want to shorten the electricity fee collection period and reduce the line loss rate. Reducing line losses can bring them considerable direct income. In China, Europe, and America, infrastructure is already comprehensive, so their goals are to make grids safer, reduce power failures, and recover power supply quickly after a power failure. These regions also have the problem of load balancing. They hope to accurately and reasonably estimate power consumption, so they can guide power plants to generate power. In addition, electric power companies in all regions want to improve customer satisfaction, provide power saving suggestions, and provide power consumption details for customers.

The major difference between a smart grid and a traditional grid is that a smart grid establishes bidirectional information flows between power generation and consumption. In a traditional power grid, only electric power is transmitted between power distribution centers and users, and no information is exchanged. In a smart grid, this situation is transformed. In addition to power transmission between power distribution centers and users, information and data is also transmitted. Scheduling centers support intelligent, real-time scheduling. They can schedule resources in each step of the power supply process in real time, building an intelligent grid. In addition, new energy supplies and services can be accessed to build green grids. For example, clean energy such as solar and wind energy is used at the power generation side and new services such as charging piles and electric vehicles are provided at the power consumption side. With these new energy supplies and services, smart grids have evolved into a global energy Internet. Non-renewable power generation, such as thermal power generation, will be gradually replaced by renewable power generation. Global energy data will be exchanged on this energy Internet to maximize resource utilization.

### 2.3.2 Solutions

We are gradually moving towards smart grids. Huawei's smart grid solution is called AMI. AMI is short for Advanced Metering Infrastructure. The AMI solution can solve the pain points of electric power companies. Smart metering replaces manual metering, improving metering efficiency and accuracy and reducing metering costs. In addition, the accurate data sent back by smart metering can be used to estimate the power consumption status of each area or even each home. The subsequent power generation can be arranged purposefully, the power consumption of users can be analyzed, and energy-saving solutions can be provided for users.

To build fully connected grids, in addition to transmitting electric power, this solution also needs to transmit data flows. Fully connected grids mean that all devices used in the power transportation process are centrally managed and data generated by these devices is uploaded to the cloud. The cloud centrally schedules the operation of the entire power system. In addition, different new technologies are used in production, management, and marketing of power grids, thereby building fully connected power grids that connect numerous devices and feature high security, low latency, wide coverage, and low power consumption.

## 2.4 IoV Solution (IoV and DRIS)

This section describes the IoV scenario that is closely related to our lives and introduces the IoV and DRIS solutions.

### 2.4.1 Common Problems

The first problem is about safety. These are very common. For example, if lights are flashing on dashboards but some drivers do not understand the fault details, they may ignore the lights, disregarding the risks. Similarly, vehicles are mainly driven manually. In foggy weather, low visibility makes driving dangerous. If a technology can help the drivers identify road situations and drive cars automatically, driving experiences will be greatly improved.

There are also some value-added services, such as insurance. The traditional vehicle insurance pricing model is based on vehicles, that is, calculating premiums based on the number of claims customers have made. But this mode is unreasonable. There is usually only a small difference between the premiums of people with good driving habits and those with bad driving habits. Vehicle insurance premiums should be calculated based on drivers instead of vehicles. The premiums should be determined based on people's driving habits.

### 2.4.2 Solutions

The technology that can solve the preceding problems is the IoV. IoV uses in-vehicle devices and wireless communications technologies to utilize all vehicle information on the information network platform and implement different functions. IoV is part of the IoT. First, it has in-vehicle devices at the sensing layer. In addition, it uses wireless communications technologies to transmit vehicle data to the platform for analysis. The platform provides the data to applications for analysis and function implementation.

In addition, the IoV in the future will have edge computing features. Vehicles in the IoV will be intelligent and can control the distances between them and other vehicles to reduce accidents, regardless of whether other vehicles are intelligent or not. Vehicles in the IoV can communicate with each other to improve traffic efficiency.

When discussing the IoV, we have to mention V2X, which is also called the vehicle-road cooperation solution. It integrates digital technologies such as cloud, IoT, AI, and 5G to implement efficient coordination between pedestrians, vehicles, roads, and clouds.

A vehicle is regarded as a device like a mobile phone or a computer. It senses all things on the road, including other vehicles, pedestrians, roadside infrastructure, and networks for information exchange, that is, V2V, V2P, V2I, and V2N.

In addition to the V2X, another concept to share with you is the Digital Road Infrastructure Service (DRIS). The DRIS does not differ greatly from the V2X. The slight difference between the two is that V2X focuses more on vehicles and the intelligence that can be achieved by them, whereas DRIS focuses on a variety of roadside devices. DRIS includes services on the cloud side and the edge side. DRIS on the cloud side provides data analysis and collaboration with the edge side on the cloud platform. DRIS on the edge side provides real-time service processing capabilities related to edge computing, such as sensor data access and road event identification. The goal of DRIS is to connect multiple roadside sensors to implement digital sensing of roads and provide useful traffic information to traffic participants.

The development of the IoV is divided into three phases. The first phase is to meet the requirements of entertainment and navigation. In this phase, vehicles are connected to the Internet so drivers can listen to music online and obtain road information. The navigation system can update road information in real time. In the second phase, IoV pays more attention to vehicle status and collect information such as fault detection, fuel consumption, and mileage. In the third phase, IoV can analyze surrounding environment information through sensors installed on vehicles to achieve automated driving. Currently, the development of the IoV industry is in the second phase and is moving towards the third phase.

Intelligent vehicle devices can be installed by original equipment manufacturers (OEMs) and aftermarket service providers. OEMs provide safety facilities and entertainment devices before delivery. Aftermarket service providers provide auxiliary measures and devices such as insurance, smart rearview mirrors, and in-vehicle Wi-Fi.

UBI is a new type of vehicle insurance in the IoV. UBI is short for usage-based insurance. UBI determines premiums based on usage and driving habits of drivers. First, it analyzes drivers' driving habits based on the data uploaded by the IoV and classifies them into four categories: high-risk, poor-behavior, neutral-behavior, and high-value. Then, it sets specific premiums for drivers based on how they score in these categories. UBI is fairer than traditional insurance services. UBI can also encourage drivers to improve their driving behavior through scoring and other methods.

In addition, there are a series of automatic toll collection services. These services are more efficient than the electronic toll collection (ETC) that we are currently using. ETC requires drivers to wait at toll gates for a short period of time, whereas automatic toll collection is automatically performed based on vehicle driving tracks. There may be no toll stations after automatic toll collection becomes commonplace. GPS positioning and GIS systems are used to determine driving tracks of vehicles and determine vehicle tolls, which makes our travel more convenient.

## 2.5 Industrial IoT Solution

This section focuses on the industrial IoT. The industrial scenario is always important in the IoT, because a large number of products need to be manufactured. Improving the efficiency of industrial manufacturing becomes particularly important. Therefore, it is an inevitability that IoT technology will be applied in manufacturing to improve efficiency and accelerate the development of the IoT.

### 2.5.1 Development Bottlenecks

In the process of pursuing more efficient industrial production, we have encountered some bottlenecks. Our goal is to be fully connected, that is, to connect people, data, and machines. When we shift our focus from production automation to full connectivity, what we need to do is to make manufacturing more efficient, refined, and intelligent. Therefore, we need to collect all vertical production information, from devices at the sensing layer such as machine tools and mechanical arms to communications devices. All the device data can be used so all the devices can be centrally managed.

In addition, we need to integrate different industries. Although there are many different applications in the upper-layer industries, the devices used at the sensing layer in the industrial field may be similar. Therefore, we can integrate these devices to implement comprehensive connections between people, data, and machines. We will begin with the collection of data used by the devices to construct the production IoT. We will use the cloud platform and big data analytics to build an industrial IoT featuring device-cloud synergy. After that, we will make O&M data on the cloud platform movable so that O&M personnel can know the production status of devices in the factory anytime and anywhere, thereby connecting all things in the industrial IoT ecosystem.

## 2.5.2 Solutions

How can we achieve full connectivity? We need to use various emerging ICT technologies. First, we need to make the devices at the sensing layer intelligent. Devices using multiple different communications protocols can be connected and communicate with each other, thereby achieving collaborative operation between them.

In addition, networks need to meet the requirements of connections in large quantities and in different scenarios. In the same industrial scenario, devices in different environments or in different applications have different requirements for networks. For example, when we control precision mechanical devices, we need networks with low latency and high reliability. For video surveillance applications, networks must provide high bandwidth. For metering applications, the network rate can be low and the network cost must be low. All these devices are used in the same industrial scenario. Therefore, we hope that networks can connect all these devices.

After the requirements of devices and networks are met, we need to process the collected data to maximize data value. In this way, we can achieve intelligent coordination and linkage between devices and full connectivity.

Based on the preceding ICT technologies, Huawei designs an overall industrial IoT solution. Like other IoT solutions in the industry, the Industrial IoT solution is also divided into four layers. At the sensing layer, various devices are used in the industrial field. At the network layer, wired and wireless access modes are used, for example, Ethernet or industrial IoT gateway. Most devices in the industrial field are huge and cannot be moved. Therefore, wired connections are used to ensure the stability of data transmission. Wireless connections will be gradually used to make the industrial IoT more flexible.

The platform layer provides capabilities such as device access, device management, and device data processing. It analyzes the data collected by devices, and transfers the processing results to various applications at the application layer for display.

The industrial IoT solution has typical applications. The first is production visualization, which is important in the industrial IoT. Production visualization means that data collected by the platform is displayed in graphics or other easy-to-understand modes. It can convert the data into a production management execution process and decision-making basis that are easy to understand.

For the industry, production visualization can improve the production reliability of factories and help enterprises better control equipment in the production process, thereby allowing factories and enterprises to control costs more easily. In addition, it provides valuable experience capital for product update, iteration, and development. For plant managers and decision makers, production visualization is a comprehensive decision-making tool that helps them better understand where problems occur, increase production output, and reduce costs.

Different parts in industrial production applications, for example, process flow, process operation efficiency, and production data sorting, can be presented in a visualized manner.

The second application is asset locating. Asset locating is a common application in campus and industrial scenarios, because warehouses are used to store assets in campus and industrial scenarios. It is important to make the best use of every item in a warehouse. We need to be able to locate the items easily and know their validity periods. That's where asset locating and tracking comes in. Asset locating can be used in campuses and industrial scenarios to help managers locate the assets that need to be used.

In industrial scenarios, all transport vehicles or machine tools can also be located. The location of a transport vehicle can be displayed on the platform and the shortest transport path can be planned for the transport vehicle. Some industrial valuables also need to be located. When these valuables are taken out of certain areas, the platform can report alarms to supervisors.

How are locating and tracking performed? First, tags are attached to all objects to be located. These tags contain wireless communications chips. Positioning base stations send signals to the tags to determine the locations of the objects. In addition, different devices are classified into the high-precision positioning scenario and low-precision positioning scenario according to their application scenarios. For example, devices that do not need to be moved frequently adopt low-precision positioning, and devices that often need to be moved adopt high-precision positioning. The location information is reported to the IoT platform and used in various applications, such as optimization of production resource usage, and foolproof management and authentication of personnel.

## 2.6 Development Trend of IoT Applications

This section describes the development trends of IoT applications based on some emerging technologies. IoT technology covers a lot of content because different technologies are used at different layers of the IoT. For example, at the network layer, we use communications technologies such as 5G. At the platform layer, we use technologies such as AI for data processing.

There are seven emerging technologies, which can be called 5IABCDE for short. "5" stands for 5G; "I" for IoT, "A" for AI, "B" for blockchain, "C" for cloud computing, "D" for big data, and "E" for edge computing. These seven technologies are popular. All of them are used in IoT application scenarios. This section uses several popular technologies as examples to explain the development trends of IoT applications.

5G and edge computing can be applied to the IoT. In addition to providing ultra-high transmission rates, 5G can provide ultra-low latency for things in the IoT. Edge computing can also greatly reduce network latency. How do these two technologies provide services for the IoT?

In the IoV scenario, these two technologies can provide low-latency services for assisted driving or automated driving. In the IoV, low network latency is important. If the network latency is too high, when there is a risk of collision, vehicles may not brake in time and accidents may occur.

The same requirement needs to be met in remote surgery and industrial manufacturing scenarios. If the latency is too high during remote surgery, doctors' operations cannot be displayed quickly enough, which poses risks to patients. Furthermore, in the high-precision industrial manufacturing field, a difference in one millimeter can have a great impact. Therefore, we need to use lower-latency networks or technologies for remote control. We use a combination of 5G or edge computing technology and IoT.

In addition, we use technologies such as big data, cloud computing, and AI. These technologies and the IoT are related to each other. Take the human body as an example. The IoT is mainly responsible for data collection. It functions like sensory organs of the human body, such as the eyes, ears, and nose. We obtain data using big data technology. We store the data in the cloud, which uses cloud technology. At the same time, AI technology, which similar to the brain, is responsible for decision-making and control. The relationship between big data, cloud computing, IoT, and AI is as inseparable as different parts of the human body.

Among the four technologies above, AI and IoT are closely related. The combination of AI and IoT is called AIoT. The data collected by the IoT can be used by AI, and a massive volume of data is what AI needs most. The two complement each other and can provide us with more intelligent services. Currently, the implementation process of most IoT projects integrates AI. We are in increasing need of AI-based decision-making. AIoT technology is no longer optional. In the future, IoT solutions that do not integrate AI capabilities will become increasingly less competitive.

In addition to the new technologies mentioned above, other new technologies, such as quantum computing, can be combined with the IoT. Although quantum computing cannot replace traditional computing in the short term, its prospects are bright. It is the future of computing, solving many problems that traditional computing cannot, such as database search, large sample simulation, and machine learning. As an analogy, traditional computing is a car, and quantum computing is a rocket. Although the rocket is very fast, it cannot be widely used in our daily lives. Only when our living standards reach a certain level can rockets be used as a means of transportation. The same is true with quantum computing. It can only become widely used when we have developed other technologies to a level similar to that of quantum computing.

There is also digital twin technology, which will be described in subsequent chapters. Simply put, it is a simulation process of obtaining data using sensors and physical models. It completes mapping in a virtual space to reflect the full lifecycle management of the corresponding entity. We can use data to present our products in the virtual world. For example, currently, O&M personnel cannot monitor the status of a high-speed train after it leaves the station. Digital twin technology can construct a model on a computer based on the train data, so that O&M personnel can view the running status of the train.

Digital twin technology can also be used in applications such as industrial IoT and smart city to build models for better O&M management.

# 3 IoT Security Technologies

Security has always been the basis of technological development. Only by ensuring the security of a technology can we develop it well. This is especially true for the IoT field, because all things are connected in the IoT world. IoT development will be restricted if we cannot resolve security issues.

This chapter describes problems faced by the IoT, Huawei's solutions to IoT security problems, and typical applications of these solutions.

## 3.1 Typical IoT Security Cases

This section describes some problems encountered by the IoT in the security field.

The first is the Tesla incident. Tesla is the most successful company in the electric vehicle market. Although Tesla is unrivaled in new-energy vehicle manufacturing, it faces substantial security challenges. Electric vehicles will be used in the Internet of Vehicles (IoV), which is one of the IoT applications. Therefore, security issues of the IoV are also related to the IoT. In this incident, when the in-vehicle system was intruded into, the attacker could remotely control the starting and stopping of the vehicle through the network. This is extremely dangerous if the vehicle is driving.

The cause of the Tesla incident is that no protection was provided for local storage of key information. Attackers could easily obtain the background remote communication certificate and password, so they could obtain the in-vehicle system login password to control the vehicle.

Some IoT-related security incidents have also occurred in China. One of them was the Nanjing Environmental Protection Bureau incident. In this incident, someone intercepted and tampered with environmental protection data at the near end of the network, causing the uploaded data to be displayed as qualified. All the indexes displayed on the monitoring platform of the Environmental Protection Bureau were normal, while the cement plant's pollution levels significantly exceeded the allowed limit. The cement plant evaded sewage charges worth millions of dollars through forged monitoring data, causing serious environmental damage.

The cause of this incident is that encryption and integrity protection were not carried out during data transmission from the gas detector of the cement plant to the RTU remote terminal, which allowed the violation to be possible. This incident occurred at the network layer of the IoT, in the form of tampering of environmental protection data through network intrusion. There have been other security incidents at the network layer, for example, the DDoS attack that occurred in the US.

DDoS, short for Distributed Denial of Service, is a common type of network attack. Hackers control a large number of devices to attack a server, causing it to break down. As a result, users cannot use the server. The DDoS attack in the US was launched using a botnet composed of IoT devices.

In this DDoS attack incident, hackers controlled some IoT devices like network cameras, video recorders, and routers. This incident paralyzed the Internet and caused network outages in several cities in the US.

In the IoT field, security is a common issue that deserves our attention. Examining these three events, we can see that IoT security issues often occur on the device and network sides. Actually, they also occur on the platform and application sides. IoT security issues are classified into four types based on the IoT architecture. In terms of industry categories, security threats mainly include LPWA security threats and IoV security threats. IoT security threats form a negative triangle model, and most of threats are on the device side, where we lack effective measures to defend against security issues. IoT devices are vulnerable to physical damage, for example, being forcibly dismantled. Because of power supply restrictions, it is not possible to run an advanced encryption system on IoT devices. Due to these two reasons, IoT security threats often target devices.

Moreover, as network technologies become increasingly advanced, more network attack technologies are emerging, and the requirements for the capabilities of network attackers are becoming lower. In the past, attackers needed to master a lot of knowledge to launch network attacks. Nowadays, attackers can launch attacks with limited communications technologies.

IoT security events increase alongside the development of the IoT. This requires us to resolve IoT security issues while developing the IoT.

## 3.2 Huawei IoT Security Architecture

How can we solve the problems mentioned above? This section describes the solution that Huawei provides.

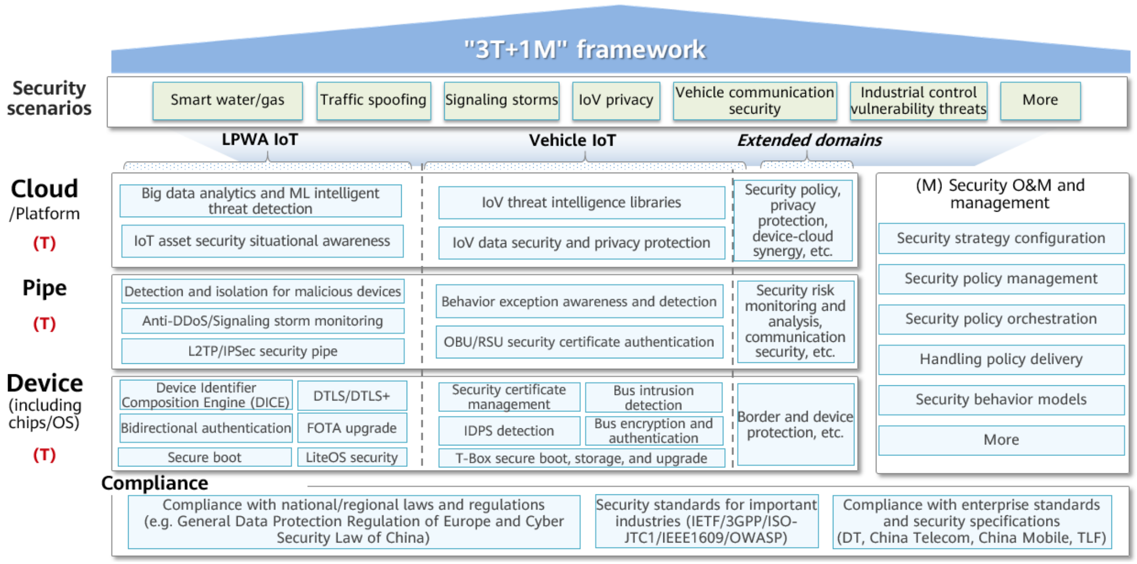

"3T+1M" framework
Security scenarios
Smart water/gas
Traffic spoofing
Signaling storms
IoV privacy
Vehicle communication security
Industrial control vulnerability threats
More
LPWA IoT
Vehicle IoT
Extended domains
Cloud /Platform (T)
Big data analytics and ML intelligent threat detection
IoT asset security situational awareness
IoV threat intelligence libraries
IoV data security and privacy protection
Security policy, privacy protection, device-cloud synergy, etc.
Pipe (T)
Detection and isolation for malicious devices
Anti-DDoS/Signaling storm monitoring
L2TP/IPSec security pipe
Behavior exception awareness and detection
OBU/RSU security certificate authentication
Security risk monitoring and analysis, communication security, etc.
Device (including chips/OS) (T)
Device Identifier Composition Engine (DICE)
DTLS/DTLS+
Bidirectional authentication
FOTA upgrade
Secure boot
LiteOS security
Security certificate management
Bus intrusion detection
IDPS detection
Bus encryption and authentication
T-Box secure boot, storage, and upgrade
Border and device protection, etc.
(M) Security O&M and management
Security strategy configuration
Security policy management
Security policy orchestration
Handling policy delivery
Security behavior models
More
Compliance
Compliance with national/regional laws and regulations (e.g. General Data Protection Regulation of Europe and Cyber Security Law of China)
Security standards for important industries (IETF/3GPP/ISO-JTC1/IEEE1609/OWASP)
Compliance with enterprise standards and security specifications (DT, China Telecom, China Mobile, TLF)

**Figure 3-1 IoT security solution architecture (see the Appendix for a reference image)**

Figure 3-1 illustrates the framework of the Huawei IoT security solution, which can be summarized as 3T+1M. In the 3T+1M framework, T stands for technology, and M stands for maintenance. The 3T indicates the different security technologies at the sensing layer, network layer, and platform layer in the IoT architecture, and 1M means unified security O&M at each layer. Through this framework, we can summarize Huawei's solutions in the IoT security field.

The Huawei IoT security solution is focused on LPWA and IoV, since most IoT security issues arise from these two fields. Of course, Huawei also supports other fields.

Huawei provides security applications for the three layers of the IoT architecture, from the sensing layer and network layer to the platform layer. Huawei also provides compliance at the bottom. Huawei has different security measures at different layers. It adheres to the concepts of device authentication, pipe monitoring, and cloud analysis. It uses device trustworthiness and identity authentication, pipe monitoring awareness, and security policy orchestration and collaboration in the cloud to build a service-oriented IoT security solution for customers. Huawei also uses unified security O&M and management across all layers to ensure normal collaboration between layers.

The following describes the security measures used at each layer. FOTA, short for firmware over-the-air, allows remote device upgrades. At the sensing layer, FOTA digital signatures ensure the validity and integrity of firmware upgrades. The integrity of upgrade packages is checked to ensure that the firmware version in use is secure and reliable.

At the network layer, the transmission network may be intruded and data may be tampered with. To protect the network and data, network transmission channels such as L2PT and IPsec are used. In addition, the platform detects upstream data for abnormal behavior and events in a timely manner.

At the platform layer, authentication is used to ensure that each device connected to the platform and all uploaded data are secure. E2E authentication and transmission security are carried out on the platform side. As shown in Figure 3-1, different devices or personnel, such as gateways or devices at the sensing layer, development or O&M personnel, and applications, must all be authenticated and authorized through different networks and portals when they access the cloud platform. This eventually ensures secure E2E data transmission of the IoT. The IoT platform may suffer from DDoS attacks. Therefore, methods must be used to defend against DDoS attacks. The method is to classify messages from devices by different priorities. If a DDoS attack occurs, the server can then determine how to process messages from different devices according to the congestion level. For example, when a server is in a low congestion state, it allows low-priority devices to access the network later or discards low-priority packets. This ensures that high-priority data can be sent to the server for normal communication.

## 3.3 Typical Huawei IoT Security Cases

This section describes typical applications of these solutions.

Here is a case related to bicycle sharing. The lock of a shared bicycle is an IoT device and it has many security risks. First, most shared bicycles do not use secure encryption methods, which allows for password forging and cracking. Some shared bicycles may use encryption methods but consume high power, failing to meet battery requirements. To address these issues, Huawei recommends DTLS+ with dual authentication to improve confidentiality and reduce power consumption.

Second, some bicycle platforms and devices may not be collaboratively connected. As a result, many malicious operations on bicycles are invisible to the platform, and the platform cannot identify damaged bicycles. The HUAWEI CLOUD IoT platform implements device-cloud synergy so that all operations on bicycles are visible on the platform, helping reduce malicious behavior.

The last problem is malicious upgrades of lock firmware. Hackers can use forged servers to upgrade lock firmware and change passwords to gain complete control of bicycles. To prevent firmware and passwords from being maliciously upgraded or changed, the HUAWEI CLOUD IoT platform uses lightweight trusted DICE to secure boot and trusted measurement of the running state.

# 4 Common IoT Communications Technologies

This chapter introduces knowledge at the IoT network layer. Communications technologies at the network layer function as a medium for connecting the sensing layer and platform layer. Communications technologies are the foundation of the IoT. If we treat the IoT as a logistics system, then communications technologies are different modes of transportation, for example, air, water, or land travel. Common communications technologies can be divided into two categories: wireless and wired. Wireless technologies can be further divided. For example, there are cellular networks used by operators, and a series of short-range communications technologies like Bluetooth.

## 4.1 Wired Communications Technologies

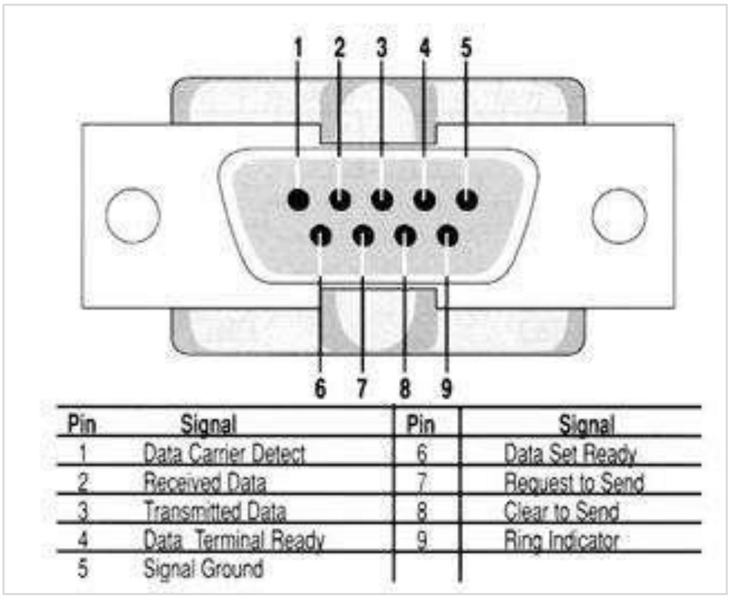
### 4.1.1 Ethernet

Basically, Ethernet is a type of network that we plug into our computers with standard network cables. Ethernet is the main local area network (LAN) technology of TCP/IP, and is the most common communications standards used in the current LAN. In the IoT field, Ethernet is widely used in industrial sectors and to provide wired access in a typical office environment. Thanks to its low cost, Ethernet has been used to create industrial Ethernet.

Ethernet uses carrier sense multiple access with collision detection (CSMA/CD). CSMA requires that a station listen on a channel before sending data and send the data only when the channel is idle. However, if both stations detect that a channel is idle and start to transmit data at the same time, a collision may occur. In addition, the channel status may be incorrectly determined if data from another station is being transmitted and not arrived at the station that performs the listening.

To address these issues, CSMA is further improved to enable a source station to continue listening on the channel during the data transmission. A collision is determined if an electromagnetic wave that exceeds an amplitude of a carrier signal sent by the source station is detected on the channel. Once a collision is detected, the source station immediately stops sending data and sends a series of blocking signals to the peer station on the bus. In this way, the damaged frames can be terminated quickly, which reduces the time and bandwidth. In other words, a station is required to detect collisions during data transmission and once a collision is detected, stop data transmission immediately. Such requirement is stipulated in CSMA/CD.

## 4.1.2 RS-232 and RS-485



| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 1 | Data Carrier Detect | 6 | Data Set Ready |
| 2 | Received Data | 7 | Request to Send |
| 3 | Transmitted Data | 8 | Clear to Send |
| 4 | Data Terminal Ready | 9 | Ring Indicator |
| 5 | Signal Ground | | |

**Figure 4-1 RS-232 port diagram (see the Appendix for a reference image)**

Anyone who has done embedded development before is likely familiar with RS-232. This port is found on the rear of legacy desktop computers. RS-232 supports one-to-one communications over short distances (less than 20 meters). RS-485 is an upgraded version. RS-485 supports one-to-many transmission. Such a bus supports a maximum of 128 transceivers. The transmission rate and communication distance are also greatly improved.

**Table 4-1 Comparison between RS-232 and RS-485**

| Item | RS-232 | RS485 |
|---|---|---|
| Communication distance | Less than 20 m | 1200 m theoretically; 300–500 m in reality |
| Transmission mode | Unbalanced transmission: single-end communications | Balanced transmission: differential transmission |
| Number of transceivers | One-to-one communications | A maximum of 128 transceivers on the bus |
| Transmission rate | 38.4 kbit/s | 10 Mbit/s |

Table 4-1 lists the differences between RS-232 and RS-485. The first difference lies in the transmission mode. RS-232 uses unbalanced transmission, which is single-ended communications. RS-485 uses balanced transmission, which is differential transmission. The second difference lies in the transmission distance. RS-232 is suitable for communications between local devices, and the transmission distance is less than 20 m in general. The transmission distance of RS-485 ranges from tens of meters to thousands of meters. Third, RS-232 allows only one-to-one communication, whereas RS-485 allows up to 128 transceivers on the bus.

### 4.1.3 Communication Serial Port Bus

In addition to RS-232 and RS-485, there is USB, which is used in serial port communications. USB, short for universal serial bus, is used to connect external devices to computers. Before the emergence of USB, there were lots of serial ports and parallel ports. For example, keyboards, mouse devices, modems, printers, and scanners are all connected to different ports. One port can be used to connect to only one device. Computers are unlikely to support so many ports, resulting in insufficient scalability and low speed. USB is designed for high speed, scalability, and ease-of-use.

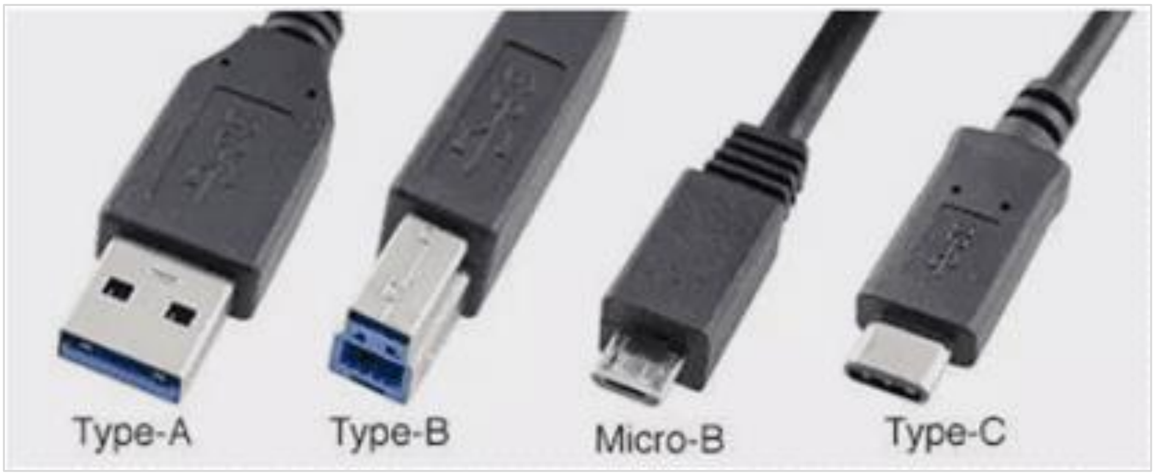


**Figure 4-2 Common USB types**

USB is also widely used in the IoT. There are four common types of USB ports: Type-A, Type-B, Micro-B, and Type-C, as shown in Figure 4-2.

### 4.1.4 M-Bus

M-Bus is short for Meter Bus. It is designed for remote metering and often used by electricity, water, and gas meters. This technology is not common in China, but it is widely used in Europe. M-Bus can power devices remotely without installing power cables. Meters using M-Bus will not be affected in the case of a power outage at home.

### 4.1.5 PLC

PLC is short for power line communication. PLC transmits data on an electrical power cable. How does it work? First, high-frequency signals that carry data are loaded onto the current and transmitted through the wire. At the receiver end, an adapter separates the high-frequency signals from the current and sends the signals to a computer.

PLC can be used only in near-end scenarios where the voltage does not change, since high-frequency signals loaded onto the wire will disappear when the voltage on the wire changes. In metering scenarios, data is loaded on the cable and uploaded to the management terminals, which connect to base stations and upload the data to the database through switches and servers. This technology is applied to data transmission from metering terminals to management terminals. In the subsequent process, power transformation and transmission are involved, where PLC is not applicable.

### 4.1.6 Comparison of Wired Communications Technologies

**Table 4-2 Comparison of wired communications technologies**

| Communication Mode | Characteristics | Application Scenario |
|---|---|---|
| Ethernet | Comprehensive protocols, universal, cost-effective | Intelligent terminals |
| RS-232 | One-to-one communications, cost-effective, short transmission distance | A few instruments, industrial control |
| RS-485 | Bus topology, cost-effective, strong anti-interference capability | Industrial instruments, meter reading |
| USB | One-to-one communications, universal, fast transmission | Smart home, office, mobile devices |
| M-Bus | Designed for meter reading, common twisted-pair cables, strong anti-interference capability | Industrial energy consumption data collection |
| PLC | For power line communication, wide coverage, easy installation | Power grid transmission, electricity meters |

Table 4-2 lists the comparison of wired communications technologies. Most of the wired communications technologies are used in industrial sectors and public utilities. In the IoT field, moveable devices account for a large proportion. Therefore, there are relatively few application scenarios for wired communications. More data is transmitted wirelessly.

## 4.2 Wireless Communications Technologies

### 4.2.1 Short-Range Wireless Communications Technologies

#### 4.2.1.1 Bluetooth

Bluetooth is everywhere in our daily lives. It is used by mobile phones, computers, tablets, and other devices. The Bluetooth technology was developed by the telecom giant Ericsson in 1994 as an alternative to RS-232 data lines. It can connect multiple devices, without data synchronization issues. In the IoT, devices like smart watches and smart scales all use Bluetooth. Bluetooth used for transmission ranges from about 10 cm to 10 m, which is relatively short compared with other wireless communications technologies. It provides a high transmission rate up to 1 Mbit/s.

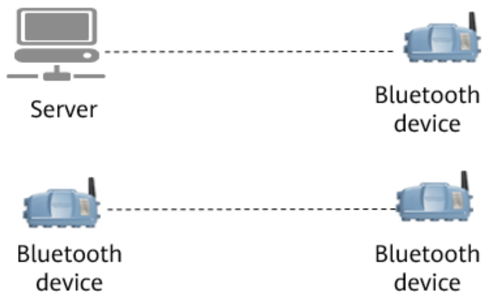


**Figure 4-3 Bluetooth working mode**

The latest Bluetooth, 5.0, supports a transmission rate of up to 3 Mbit/s and can transmit over about 300 meters. This technology can be divided into Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) and Bluetooth low energy (BLE). The BR/EDR type supports only point-to-point communications, whereas BLE supports point-to-point, broadcast, mesh, and other modes of communications. The BLE type is mainly used in IoT scenarios for data transmission among multiple devices.

#### 4.2.1.2 Wi-Fi

Wi-Fi is widely used for daily Internet access at home or in the office. Wi-Fi usually uses the 2.4 GHz and 5 GHz frequency bands to provide different services for different devices.

Wi-Fi has a longer range than earlier versions of Bluetooth. It also supports one-to-many connections and fast transmission. However, it has poor security and stability. You likely have experience with this in your daily life, for example, video freezing when watching a video, and long delay when playing games. The delay may be 20 ms to 30 ms, or even 100 ms to 200 ms. Wi-Fi also uses a lot of power.

Similar to Bluetooth, there is a next generation Wi-Fi standard, Wi-Fi 6. This version supports a transmission rate of 9.6 Gbit/s and the latency is kept down to 20 ms.

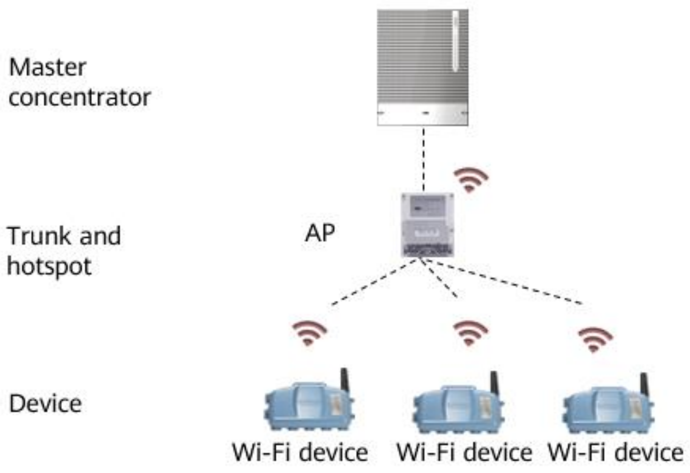


**Figure 4-4 Wi-Fi working mode**

### 4.2.1.3 ZigBee

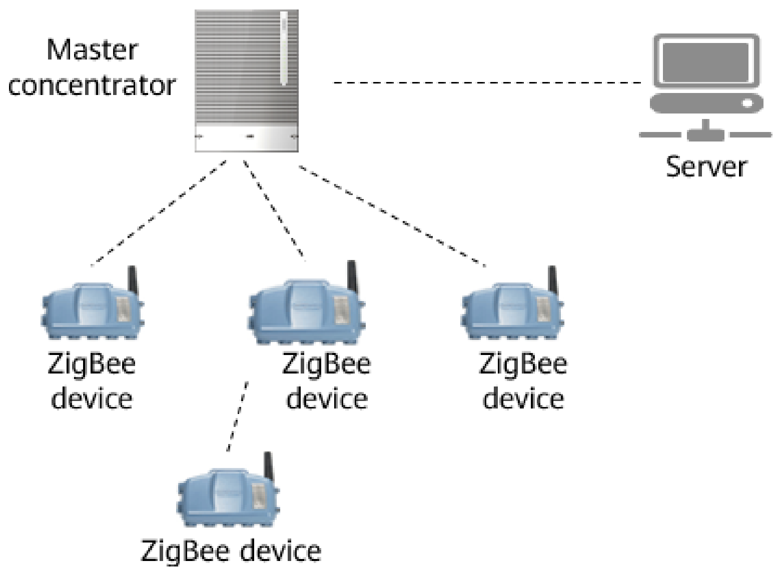


**Figure 4-5 ZigBee working mode**

ZigBee features low consumption. Figure 4-5 shows the ZigBee working mode. As shown in Figure 4-4, Wi-Fi devices can only be connected to access points (APs) or master concentrators. However, ZigBee is different. ZigBee allows for direct data transmission between devices.

ZigBee networking is easy. If the AP in a Wi-Fi network is faulty, the entire network breaks down. This will not happen in a ZigBee network, because each device on a ZigBee network functions as a relay. If a device fails, the network adjusts on the fly to find another device that can take its place. The name ZigBee is derived from the waggle dance that honey bees perform through movements such as zigzagging to communicate the location of pollen. In other words, the waggle dance can be thought of as their communications network. This is similar to easy networking of ZigBee.

The cost of a ZigBee module is as low as US$2. The transmission rate is 20 to 250 kbit/s, which is very low when compared with Wi-Fi. However, ZigBee lacks robust compatibility and is difficult to maintain.

### 4.2.1.4 Z-Wave

Z-Wave is similar to ZigBee, but Z-Wave is more reliable. In addition, Z-Wave is not open-source. Z-Wave chips can be obtained only from Sigma Designs. In the initial design, the Z-Wave technology was intended for wireless control of smart home appliances. Data was transmitted in a small data format and the transmission rate of 40 kbit/s was adequate. When the technology was first utilized, a rate of 9.6 kbit/s was used. Compared with other radio technologies of the same type, Z-Wave uses a lower transmission frequency, can transmit over longer distances, and is priced competitively.

### 4.2.1.5 Comparison of Short-Range Wireless Communications Technologies

**Table 4-3 Comparison of common short-distance wireless communication technologies**

| | Bluetooth | Wi-Fi | ZigBee | Z-Wave |
|---|---|---|---|---|
| Frequency band | 2.4 GHz | 2.4 GHz<br>5 GHz | 868 MHz/915 MHz, 2.4 GHz | 868.42 MHz (Europe)<br>908.42 MHz (USA) |
| Transmission rate | 1–3 Mbit/s (24 Mbit/s over 802.11 links) | 802.11b: 11 Mbit/s<br>802.11g: 54 Mbit/s<br>802.11n: 600 Mbit/s<br>802.11ac: 1 Gbit/s<br>802.11ax: 9.6 Gbit/s | 868 MHz: 20 kbit/s<br>915 MHz: 40 kbit/s<br>2.4 GHz: 250 kbit/s | 9.6 kbit/s or 40 kbit/s |
| Typical distance | 1–300 m | 50–100 m | 2.4 GHz band: 10–100 m | 30 m (indoor) to 100 m (outdoor) |
| Typical application | Data exchange between nearby nodes such as a mouse, wireless headset, mobile device, and computer | WLAN, high-speed Internet access at home and other indoor areas | Home automation, building automation, and remote control | Smart home appliance, monitoring and control |

This section compares the four short-range wireless communications technologies. Bluetooth and Wi-Fi have higher transmission rates. Earlier versions of Bluetooth only support one-to-one connections, whereas Wi-Fi supports one-to-many. Bluetooth is mainly used on devices such as mouse devices, headsets, and mobile phones, whereas Wi-Fi is mainly used for high-speed Internet access at home or in other indoor environment. ZigBee and Z-Wave feature low transmission rates and multiple connections. They are often used in IoT fields such as home automation, smart home, and smart building. They are unlikely to be used outside of the IoT, because of their low transmission rate.

## 4.2.2 Cellular Mobile Communications Technology

### 4.2.2.1 2G

2G has been around for a long time. The emergence of 2G marks the origin of digital communications. The full name of 2G is Global System for Mobile Communications, or GSM. On a 2G network, we can make calls and send SMS messages. Later, GPRS emerged. It is also known as 2.5G. It has a higher transmission rate than GSM, and it can be regarded as an upgrade to GSM.

### 4.2.2.2 3G

A 3G network provides a higher data transmission rate, which can reach hundreds of kbit/s. Over a 3G network, users can make video calls and watch TV programs using mobile phones. In China, three major operators use different standards for 3G networks separately: CDMA2000, WCDMA, and TD-SCDMA. HSPA+, the latest WCDMA technology from China Unicom, supports a maximum transmission rate of 42 Mbit/s. So far, HSPA+ is still used in most video data transmission fields.

### 4.2.2.3 4G

HSPA+ in 3G can be used to transmit video data, but 4G is required to transmit HD video. Although 5G is now popular, most of our mobile phones still use 4G. Over a 4G network, users can use phones to stream video and listen to music.

In the 4G network, LTE-Cat is a different level of 4G. Cat stands for category. Each different category refers to a different transmission rate level supported by the device when accessing a 4G network. Each level specifies the maximum transmission rate that the device can reach. In the early days of 4G, devices were category 4, but now most of the 4G devices are category 6. Module chips for devices at different levels have different prices too.

**Table 4-4 LTE UE categories**

| Level | Downlink Rate (Mbit/s) | DL-MIMO | Uplink Rate (Mbit/s) |
|---|---|---|---|
| 1 | 10 | 1 | 5 |
| 2 | 50 | 2 | 25 |
| 3 | 100 | 2 | 50 |
| 4 | 150 | 2 | 50 |
| 5 | 300 | 4 | 75 |
| 6 | 300 | 2 or 4 | 50 |
| 7 | 300 | 2 or 4 | 150 |
| 8 | 1200 | 8 | 600 |
| 9 | 450 | 2 or 4 | 50 |

| Level | Downlink Rate (Mbit/s) | DL-MIMO | Uplink Rate (Mbit/s) |
|---|---|---|---|
| 10 | 450 | 2 or 4 | 100 |

In the IoT field, category 1 UEs are widely used because some devices demand high network rates, although most devices use LPWA. UE category 4 or later provides a higher rate, but its modules are too expensive for IoT devices. It is more cost-effective to use UE category 1.

UE category 1 supports a downlink rate of up to 10 Mbit/s, enabling IoT devices with lower power consumption and costs to connect to LTE networks. LTE operators around the world deploy LTE networks based on 3GPP Release 8 or later. As such, operators can simply reconfigure parameters to permit the access of UE category 1 without needing to upgrade the networks.

#### 4.2.2.4 5G

5G is the fifth-generation mobile communications network. 5G's theoretical maximum transmission rate can reach 10 Gbit/s. Three 5G application scenarios are defined: eMBB, which is defined by a rate of up to 10 Gbit/s; mMTC, which refers to support for up to 1 million connections per square kilometer; uRLLC, for when latency has to be kept down to 1 ms. Among the three scenarios, eMBB is closely related to users because it can provide high-bandwidth Internet access rates. The other two scenarios are closely related to the IoT. Massive connections and ultra-low latency can help improve the service capability of the IoT in many applications.

#### 4.2.2.5 Comparison of Cellular Mobile Communications Technologies

**Table 4-5 Comparison of cellular mobile communications technologies**

| | 2G | 3G | 4G | 5G |
|---|---|---|---|---|
| Frequency band | Authorized frequency band<br>(mainly 900 MHz) | Authorized frequency band<br>(mainly 900 MHz and 1800 MHz) | Authorized frequency band<br>(1800–2600 MHz) | Authorized frequency band:<br>C-band<br>mmWave |
| Transmission rate | GSM: 9.6 kbit/s<br>GPRS: 56-114 kbit/s | TD-SCDMA: 2.8 Mbit/s<br>CDMA2000: 3.1 Mbit/s<br>WCDMA: 14.4 Mbit/s | Downlink<br>Category 6/7: 300 Mbit/s<br>Category 9/10: 450 Mbit/s | 10 Gbit/s<br>(Balong 5000 chips<br>support a downlink rate of 4.6 Gbit/s and an uplink rate of 2.5 Gbit/s) |

| | 2G | 3G | 4G | 5G |
|---|---|---|---|---|
| Typical application | POS terminals and smart wearables | Vending machines, smart home appliances | Mobile terminals | AR, VR, assisted driving, automated driving, and telemedicine |

Table 4-5 lists the comparison of the cellular mobile communications technologies. Technology continues to evolve, and the evolution from 2G networks to 5G networks means higher frequency band (from 900 MHz to millimeter wave) and transmission rate. Different technologies have different applications. We should choose the most appropriate network, not necessarily the best one. For example, there is no need to use more advanced networks for POS terminals or vending machines, since 2G and 3G can fully support their functions. 5G is relatively expensive, so it is not necessarily the best match for all solutions.

## 4.2.3 LPWA Communications Technologies

### 4.2.3.1 Sigfox

Sigfox is the name of an IoT technology startup company in France, and is also the name of the LPWA communication technology developed by the company. Sigfox is designed for low-throughput projects. It can maintain stable data connections with less transmission power (50 to 100 microwatts). Sigfox exists to store the massive amount of data generated by IoT devices, but it is not sufficiently trusted for data storage and information security. All these make Sigfox not as popular as LoRa.

### 4.2.3.2 LoRa

LoRa is short for long range. Different from Sigfox, LoRa is maintained and managed by the LoRa Alliance. The LoRa technology was developed by Semtech. With joint efforts of the LoRa Alliance, Semtech developed LoRaWAN, which is applied to the LPWA field. LoRaWAN is an ultra-long-range wireless transmissions solution based on spread spectrum technology. LoRaWAN mainly runs on unlicensed frequency bands. The application of LoRa includes automatic meter reading, smart home appliance, building automation, wireless warning and security systems, industrial monitoring and control, and remote irrigation systems.

### 4.2.3.3 NB-IoT

NB-IoT was first proposed by Huawei and Vodafone. Later, companies such as Ericsson and Qualcomm joined its development. NB-IoT takes advantages of existing cellular networks. It only needs about 180 kHz of bandwidth and can be directly deployed in the existing cellular networks, greatly reducing the cost for operators.

### 4.2.3.4 eMTC

eMTC was proposed by Ericsson based on 4G networks. It is mainly used in IoT scenarios that require deep coverage and a considerable number of connections. Compared with NB-IoT, eMTC has a higher rate, but provides less coverage and requires more power. It supports voice communications, making it popular in scenarios where voice communications are required.

### 4.2.3.5 Comparison of LPWA Technologies

**Table 4-6 Comparison of LPWA technologies**

| | SigFox | LoRa | NB-IoT | eMTC |
|---|---|---|---|---|
| Frequency band | Sub-GHz unlicensed frequency band | Sub-GHz unlicensed frequency band | Mainly sub-GHz licensed frequency band | Sub-GHz licensed frequency band |
| Transmission rate | 100 bit/s | 0.3–5 kbit/s | < 250 kbit/s | < 1 Mbit/s |
| Typical distance | 1–50 km | 1–20 km | 1–20 km | 2 km |
| Typical application | Smart home appliances, smart electricity meters, mobile healthcare, remote monitoring, and retail | Smart agriculture, intelligent building, and logistics tracking | Water meters, parking, pet tracking, garbage disposal, smoke alarm, and retail devices | Shared bicycles, pet collars, POS terminals, and smart elevators |

Table 4-6 lists the comparison of LPWA technologies. Sigfox and LoRa run on the sub-GHz unlicensed frequency band. Sub-GHz means that the frequency is below 1 GHz, which mainly refers to the frequency band from 27 MHz to 960 MHz. Sub-GHz is an ideal choice for long-range, low-power-consumption communications. Why? Given the same power, higher frequencies are less able to pass through physical obstacles. Devices in LPWA scenarios, such as electric meters or water meters, are often placed deep underground, so penetration is important. They need to run on lower frequency bands.

As listed in Table 4-6, NB-IoT mainly runs on Sub-GHz licensed frequency bands, but some NB-IoT networks run on frequency bands higher than 1 GHz. The specific reasons will be explained in the following sections.

The differences between LoRa and NB-IoT are as follows:

- Different lineups

  LoRa is developed by Semtech (an American company), and NB-IoT is a standard formulated by 3GPP. The upstream and downstream industry chains of LoRa are basically controlled by Semtech. However, the entire communications industry is engaged in the NB-IoT upstream and downstream industry chains. For example, Huawei, Qualcomm, Samsung, and MediaTek are all working on NB-IoT chip modules.

- Different frequency bands

NB-IoT uses licensed frequency bands, whereas LoRa uses unlicensed frequency bands.

In China, licensed frequency bands are authorized by the Ministry of Industry and Information Technology (MIIT). Only authorized operators can use these frequency bands.

Unlicensed frequency bands can be used by anybody. If individuals or enterprises use the LoRa working frequency band for other purposes, severe interference will be caused to LoRa devices in that region, even making the entire LoRa network unavailable.

NB-IoT networks can provide better quality over LoRa networks.

- Enterprises need to build their own LoRa networks, whereas NB-IoT networks are built by operators and ready for use.

  Operators do not operate LoRa networks. If enterprises want to use LoRa networks, they need to purchase a full set of network operations software and hardware, such as base stations, network devices, and servers, and build and operate networks by themselves.

  Currently, the three major operators in China have deployed base stations and NB-IoT networks. Enterprises only need to purchase NB-IoT SIM cards from operators and insert them into devices equipped with NB-IoT modules.

- What are the advantages of LoRa?

  LoRa does not show any advantage for 90% of enterprises. However, if enterprises want to use the IoT in rural areas, remote mountainous areas, and independent factories (such as smart farms and smart factories), where there are no NB-IoT networks, they need to build LoRa networks by themselves.

  In addition, large enterprises or governments who are very sensitive to data security might prefer to build their own LoRa networks and maintain them by themselves.

  These are the major differences between NB-IoT and LoRa.

## 4.2.4 Wireless Communications Technologies

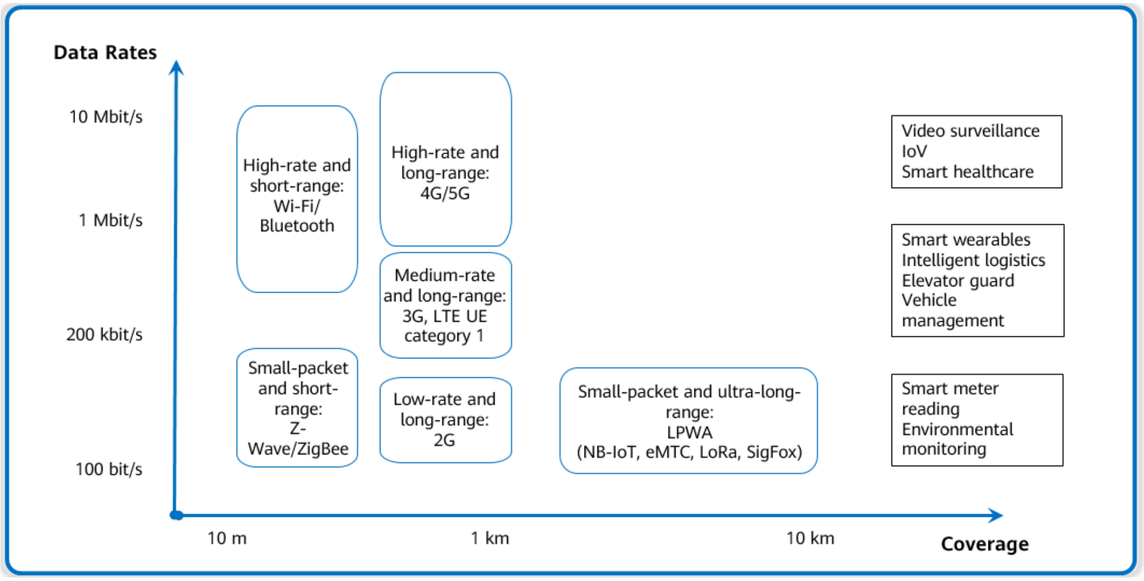

Data Rates
10 Mbit/s
1 Mbit/s
200 kbit/s
100 bit/s
High-rate and short-range: Wi-Fi/ Bluetooth
High-rate and long-range: 4G/5G
Medium-rate and long-range: 3G, LTE UE category 1
Small-packet and short-range: Z-Wave/ZigBee
Low-rate and long-range: 2G
Small-packet and ultra-long-range: LPWA (NB-IoT, eMTC, LoRa, SigFox)
Video surveillance
IoV
Smart healthcare
Smart wearables
Intelligent logistics
Elevator guard
Vehicle management
Smart meter reading
Environmental monitoring
10 m
1 km
10 km
Coverage

**Figure 4-6 Comparison of wireless communications technologies (see the Appendix for a reference image)**

All these three types of wireless communications technologies can be applied to the IoT. How do we select the right technology for a given application?

Figure 4-6 illustrates which technology works best for different scenarios.

The two axes in Figure 4-6 are the transmission rate and transmission distance, which correspond to major requirements for different IoT scenarios. There are three types of data rates: high, medium, and low. For example, Internet of Vehicles (IoV) requires a high data rate; smart logistics and vehicle management require a medium data rate; metering requires very little data transmission. These scenarios can be further divided based on the transmission distances that must be supported to get six types of networks. As mentioned above, the most appropriate technology should be selected for a specific application. In addition to the transmission distance and transmission rate, costs must be considered, in order to maximize the return on investment.

# 5 NB-IoT Communications Technologies and Solutions

This chapter describes the key technologies and features of NB-IoT, as well as some NB-IoT solutions. IoT scenarios are different from the traditional Internet used by mobile phones and computers. What are the characteristics of IoT in wireless communications scenarios? First, IoT devices at the sensing layer send and receive data packets that are small and transmitted infrequently. Some of them may send less than 10 bits of data in an entire day. Second, these devices need to conserve power, even during communications, in order to prolong their service life.

To sum up, small packets are occasionally transmitted through passive devices. The communications network needs to run on very little power and cover a wide area, which is why we call this an "LPWA" or "low power wide area" scenario.

In LPWA scenarios, the most popular technology is NB-IoT. NB-IoT is widely used in public utilities and city management. It is important to learn NB-IoT technical details and solutions.

This chapter describes the development, key technologies, and solutions of NB-IoT.

## 5.1 NB-IoT Standards Evolution and Industry Development

### 5.1.1 Evolution of NB-IoT Standards

NB-IoT was proposed by Huawei and Vodafone, and vendors such as Qualcomm and Ericsson joined later. NB-IoT evolved out of NB-M2M, and was officially named NB-IoT in 2015. In 2016, the NB-IoT standards were officially frozen, but it did not stop evolving. Many new features were introduced in R14 in 2017. NB-IoT R14 has a higher transmission rate and supports site positioning and multicast. At the latest conference on July 9, 2020, NB-IoT was incorporated into the overall 5G plan.

This event was very important for NB-IoT. It indicates that IoT devices that access the network through NB-IoT can connect to the 5G core network and use 5G services such as edge computing and network slicing. However, the current NB-IoT does not support access to 5G networks. It still needs evolution.

## 5.1.2 LPWA Technologies Adopted by Global Operators

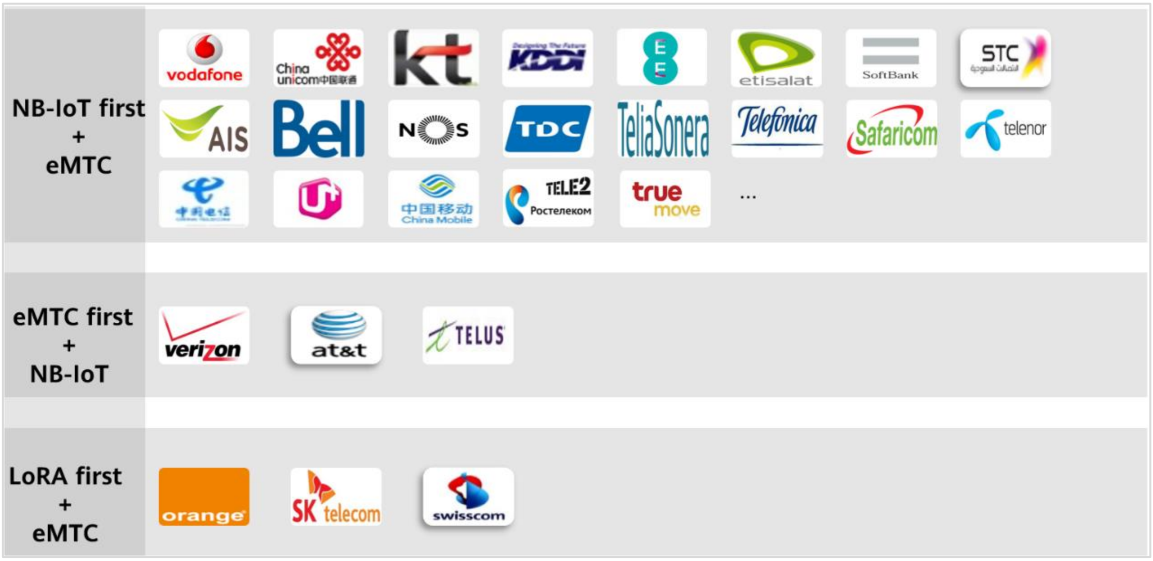


**Figure 5-1 LPWA technologies adopted by global operators**

As shown in Figure 5-1, during the selection of LPWA technologies, most operators choose to deploy NB-IoT networks first and then add an eMTC network later. This is because they would rather deploy a network that provides entirely new functions. Before the emergence of NB-IoT, there were no LPWA networks, and there were no networks specially designed for IoT devices.

The operator networks we are using today are designed for human beings. To facilitate communications between human beings, these networks provide voice communications and are always being optimized for higher and higher transmission rates. In contrast, NB-IoT networks provide very slow transmission rates. User experience would be poor if we tried to use NB-IoT networks for our mobile phones. However, NB-IoT networks are suitable for devices, since these devices are typically widely separated, need to run on limited power, and do not need to transmit much data. As for the eMTC network, it has a higher rate than NB-IoT and supports voice communications. It is similar to a 2G network that way. eMTC can be used to replace an old 2G network, but it is not as urgent as NB-IoT. This is why most operators choose to deploy NB-IoT first.

## 5.1.3 NB-IoT Spectrum Selections of Global Operators

In addition to technology selection, there is another problem that operators must consider, spectrum selection. The network frequency band must be low enough to meet requirements for both coverage and penetration. IoT devices at the sensing layer, such as gas meters or water meters, are often placed in cabinets in kitchens. They need frequencies with strong penetration.

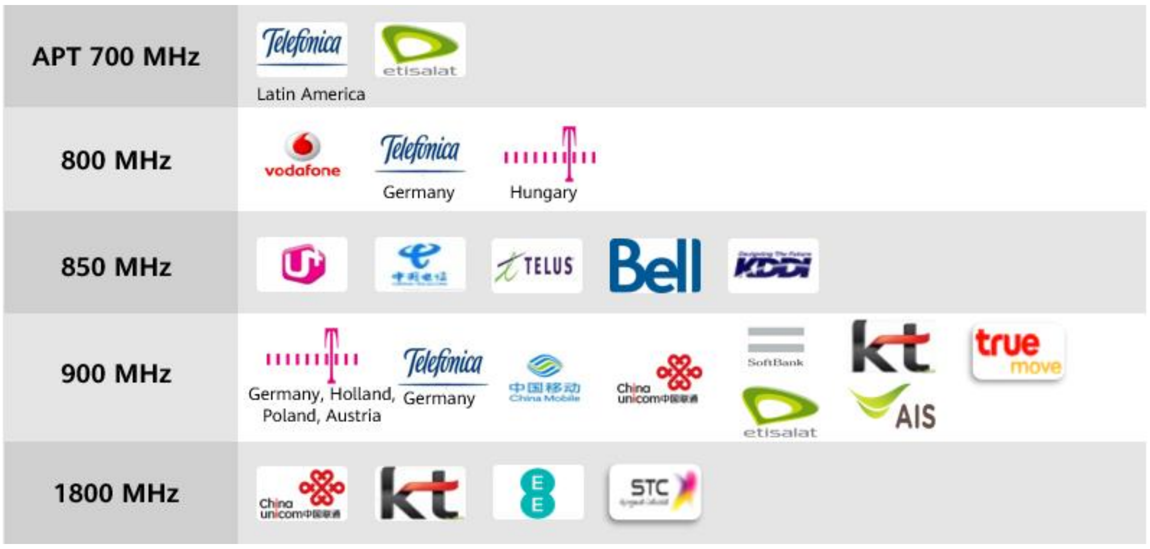

APT 700 MHz
Latin America
800 MHz
Germany
Hungary
850 MHz
900 MHz
Germany, Holland, Poland, Austria
Germany
1800 MHz

**Figure 5-2 NB-IoT spectrum selections of global operators**

A lower frequency band means stronger penetration. As shown in Figure 5-2, most operators deploy networks on the sub-GHz frequency band, on frequencies ranging from 700 MHz to 900 MHz. A few operators like China Unicom deploy their networks on the 1800 MHz band. NB-IoT networks are mainly deployed on the sub-GHz frequency band, but not all of them.

NB-IoT networks are based on 4G LTE networks. Operators can upgrade some 4G base stations to function as NB-IoT base stations. The case is different for China Unicom, because China Unicom's 4G base stations were upgraded from 3G base stations. They can directly upgrade their 3G 1800 MHz base stations to support NB-IoT, saving a lot of money. This is why China Unicom choose to deploy NB-IoT networks on the 1800 MHz frequency band.

## 5.1.4 NB-IoT Industry Development

In addition to network technologies, base stations, and frequency bands, chips that support connections between devices and base stations are required. Huawei launched Boudica 120 chips as early as R13. This chip supports the sub-GHz frequency band, but not mobility features introduced since R14. Huawei launched Boudica 150 chips to support the new R14 features.

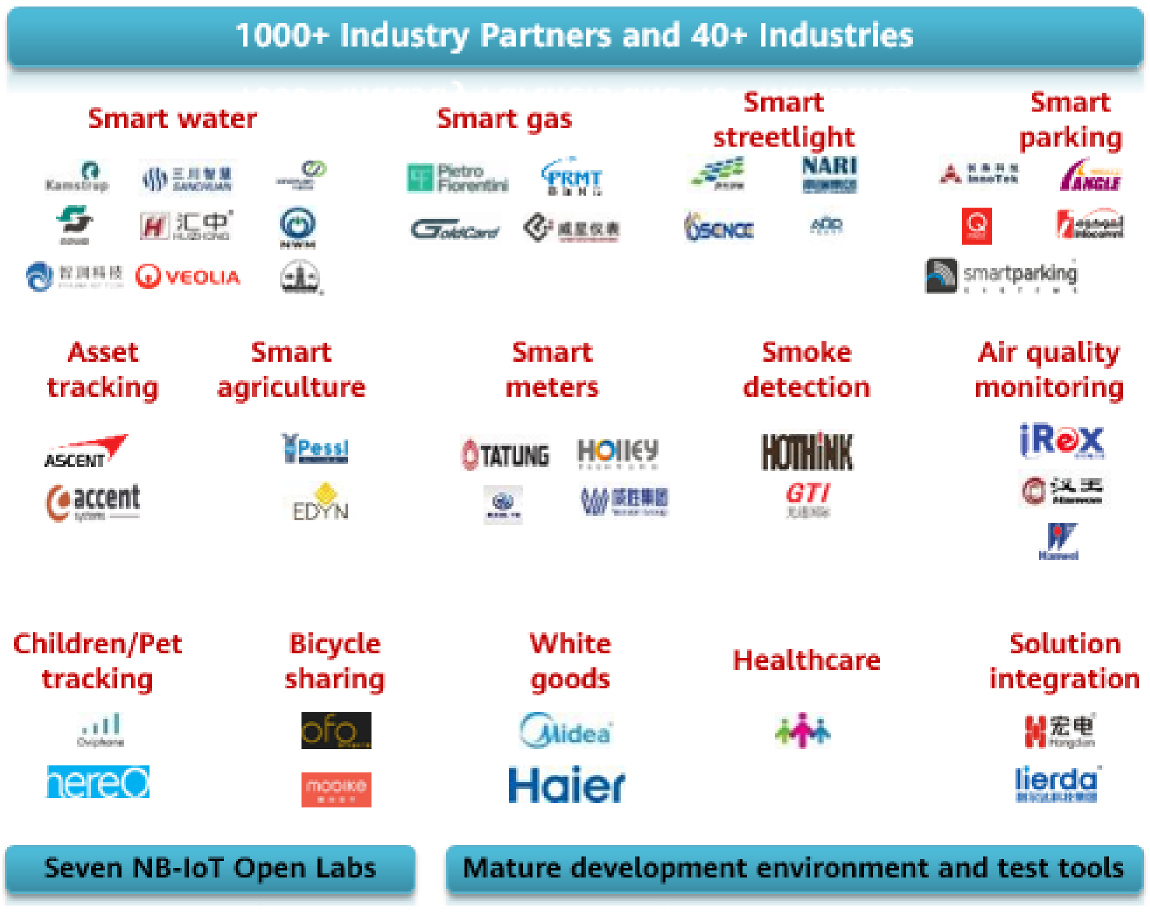


**Figure 5-3 NB-IoT ecosystem partner list**

Figure 5-3 shows the application of NB-IoT technologies. NB-IoT involves many fields like water meters, gas meters, street lamps, and smart parking. The following sections describe some NB-IoT solutions.

## 5.2 Key NB-IoT Technologies

This section describes the key technologies of NB-IoT.

NB-IoT stands for Narrowband Internet of Things. Why do we say narrowband? Its system bandwidth is just 180 kHz, which is even smaller than the 4G guard bandwidth. NB-IoT networks are based on 4G networks. NB-IoT uses the same OFDMA and SC-FDMA techniques for uplink and downlink multiplexing. Although NB-IoT is based on 4G LTE, NB-IoT is designed for a much lower data rate and to run much less bandwidth than a 4G network.

Many of the parts of a 4G network have been stripped out for NB-IoT. For example, NB-IoT has only two physical channels and one signal in the uplink and three physical channels and two signals in the downlink. The reduction of channels and signals is to achieve wider coverage with less power and for less money.

The following sections describe the four key features of NB-IoT and the technologies involved in these features.

## 5.2.1 Ultra-Low Cost

### 5.2.1.1 NB-IoT Deployment Modes

NB-IoT can be directly deployed on existing operator networks to save money. There are three deployment modes. The first is standalone deployment, which means that NB-IoT does not rely on existing LTE networks. Instead, GSM frequency bands are re-farmed. The channel bandwidth of GSM is 200 kHz, which is more than enough for the 180 kHz needed for NB-IoT.

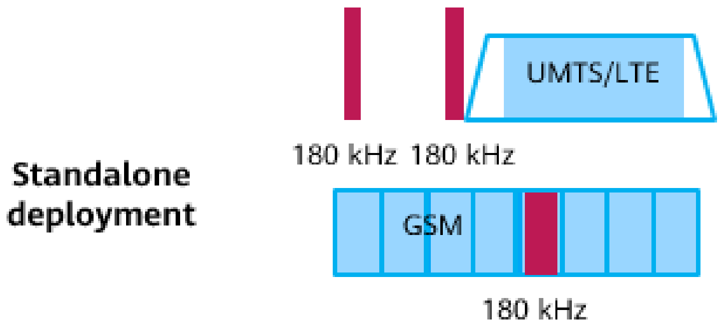


**Figure 5-4 Standalone deployment**

The second is guard band deployment. NB-IoT can be deployed on the guard band of the existing 4G frequency band, taking advantage of existing idle resources.

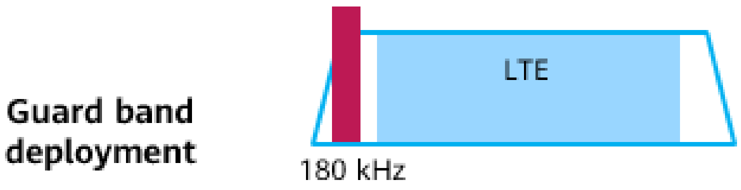


**Figure 5-5 Guard band deployment**

The third is in-band deployment, where NB-IoT is directly deployed in the 4G frequency band. In a 4G network, the frequency and time domains are divided into small resource blocks. NB-IoT is designed to be fully compatible with 4G. The bandwidth of each small resource block in 4G is 180 kHz. That is why the system bandwidth of NB-IoT was designed to be 180 kHz. Regardless of what deployment mode is used, NB-IoT does not depend on signal resources of any system.

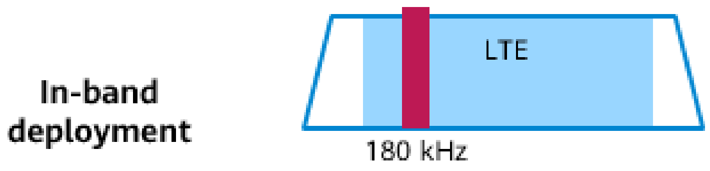


**Figure 5-6 In-band deployment**

This is one of the reasons for the ultra-low costs. Due to technical limitations, NB-IoT cannot be deployed in the in-band or guard band of the LTE system. Therefore, standalone deployment is more popular recently.

#### 5.2.1.2 Chip Design

The ultra-low cost is also partly thanks to some functions in the chip design being simplified. Huawei has rolled out Boudica 150 chips dedicated for the IoT through single-antenna, FDD half-duplex, and the like. For NB-IoT, single-antenna and FDD half-duplex are enough. This simplified design reduces the cost of NB-IoT modules.

### 5.2.2 Ultra-Low Power Consumption

NB-IoT supports eDRX and PSM. Discontinuous reception, or DRX is a paging mode used by many mobile phones. What type of paging should the mobile phone use to ensure that messages can be monitored in real time? Does that mean the mobile phone has to be waiting to be paged all the time, like a student in class waiting for the teacher to call on them?

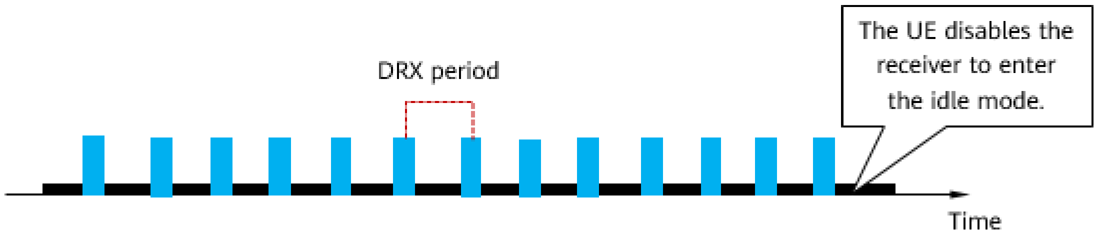


**Figure 5-7 DRX working principles**

However, this waiting uses a lot of power. DRX addresses this issue. The blue pulse shown in Figure 5-7 indicates that the mobile phone is being paged. After each paging message is received, the mobile phone enters the IDLE state and disables the receiver, like a student taking brief a nap in class. In DRX mode, the interval between each paging is called a DRX period, which can be 1.28s, 2.56s, 5.12s, or 10.24s.

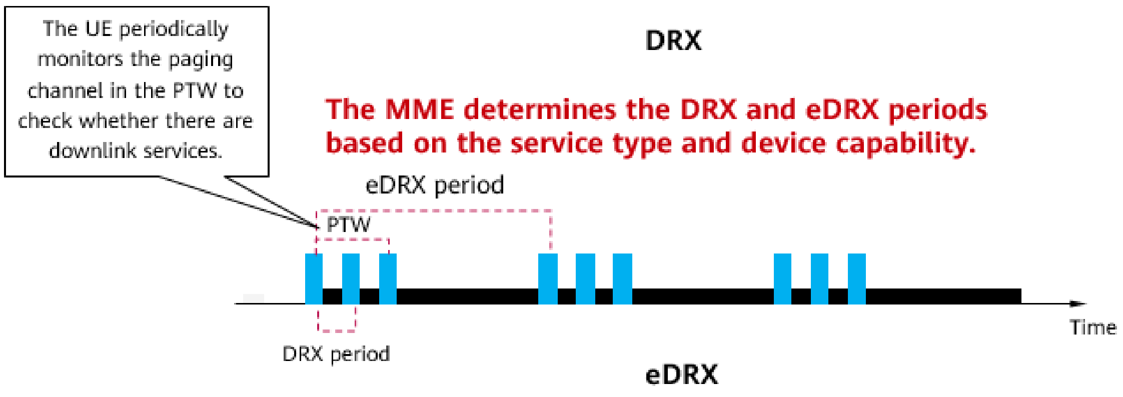

The UE periodically monitors the paging channel in the PTW to check whether there are downlink services.
DRX
The MME determines the DRX and eDRX periods based on the service type and device capability.
eDRX period
PTW
Time
DRX period
eDRX

Figure 5-8 eDRX working principles

DRX still uses too much power for IoT devices. Even if you only make calls or send SMS messages on a mobile phone, you still need to charge once every day in most cases, or at least every couple of days. IoT devices need to last several years or even more than a decade on a single charge. That why we developed eDRX based on DRX.

As shown in Figure 5-8, eDRX introduced a paging time window (PTW). Within each PTW, an IoT device makes three paging attempts, at the same interval as the DRX period. After these three paging attempts, the device enters sleep mode for a long time. This period is called an eDRX cycle. An eDRX cycle can be up to 2.92 hours long. Operators can set the eDRX cycle based on the actual situation of IoT devices and required data.

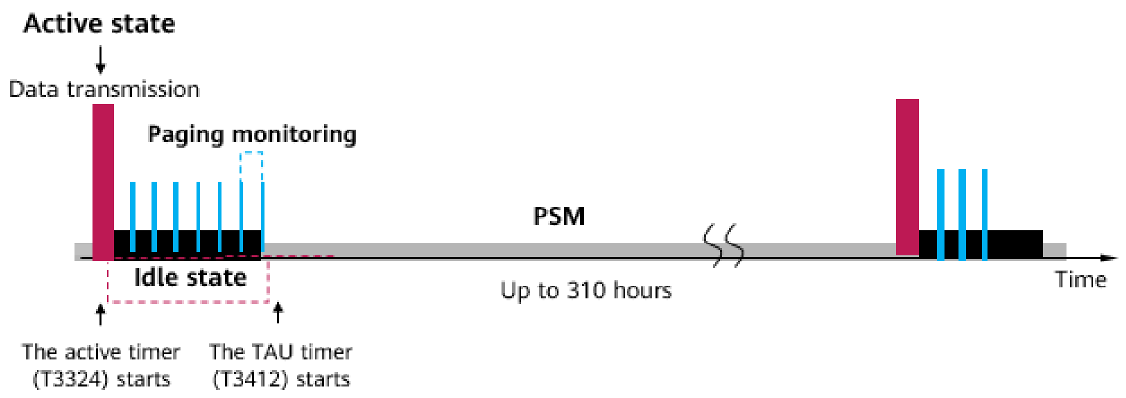


Figure 5-9 PSM working principles

However, even eDRX does not save enough power for some IoT devices, which is why PSM was introduced. With PSM, the sleep period can be up to 310 hours, or about 13 days, as illustrated in Figure 5-9. With PSM, if the service platform at the application layer delivers commands to a device, the device will not receive these commands immediately. These commands are temporarily stored on the IoT platform and will be issued after the device wakes up from sleep mode.

These three modes correspond to different IoT scenario requirements. For example, DRX is good for shared bicycles, because users do not want to wait for as long as 5 minutes before a bicycle automatically unlocks. A 5-minute waiting duration is required if eDRX is used. But eDRX is fine for scenarios such as logistics monitoring, because goods do not need to be monitored in real time during transport. You only need to check their location at certain intervals. PSM is good for water or electricity meters, since you only need to check the data on these devices about twice a month. NB-IoT low power consumption modes are closely related to scenarios. Developers need to select the power saving modes for different application scenarios.

### 5.2.3 Ultra-Wide Coverage

NB-IoT provides 20 dB better coverage than GPRS. That is three times as much coverage, and that is why NB-IoT signals can penetrate through physical obstacles better too. How does NB-IoT expand its coverage? In the downlink, NB-IoT enhances reliability through repeated transmission for more gain.

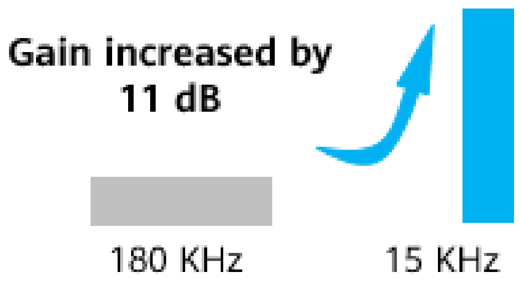


**Figure 5-10 Increased power spectral density**

In the uplink, NB-IoT also increases the gain through repeated transmission, just like it does in the downlink. In addition, NB-IoT uses a single subcarrier for transmission, a 15 kHz subcarrier. This is different from that of 4G. In 4G, 180 kHz is required at least for every transmission. So for any given amount of transmission power, you get better gain with narrowband, as shown in Figure 5-10. This is one of the reasons why NB-IoT can provide such wide coverage.

### 5.2.4 Massive Connections

What is the difference between the traffic model of IoT devices and that of mobile phones? IoT devices are vastly more numerous, but each device sends only very small data packets and latency is generally not a serious issue. To ensure the communications quality for all users, a base station can only serve a limited number of users. IoT devices do not require especially high quality communications, so more devices can be connected to the same base station. A single base station can handle up to 50,000 devices, partly because many of them are in the dormant state.

## 5.3 NB-IoT Solutions

Figure 5-11 shows the overall architecture of the NB-IoT solution. This architecture can be associated with the IoT architecture in the previous section. NB-IoT devices are deployed at the sensing layer. NB-IoT base stations are at the network layer. The IoT platform is at the platform layer. A variety of applications are provided at the application layer.

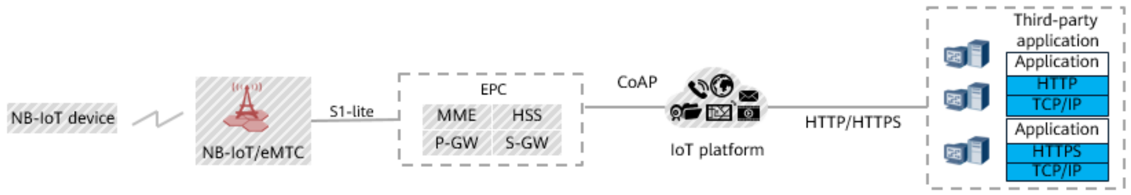


**Figure 5-11 NB-IoT solution architecture (see the Appendix for a reference image)**

The following section describes some NB-IoT applications from the perspective of the overall architecture of this solution.

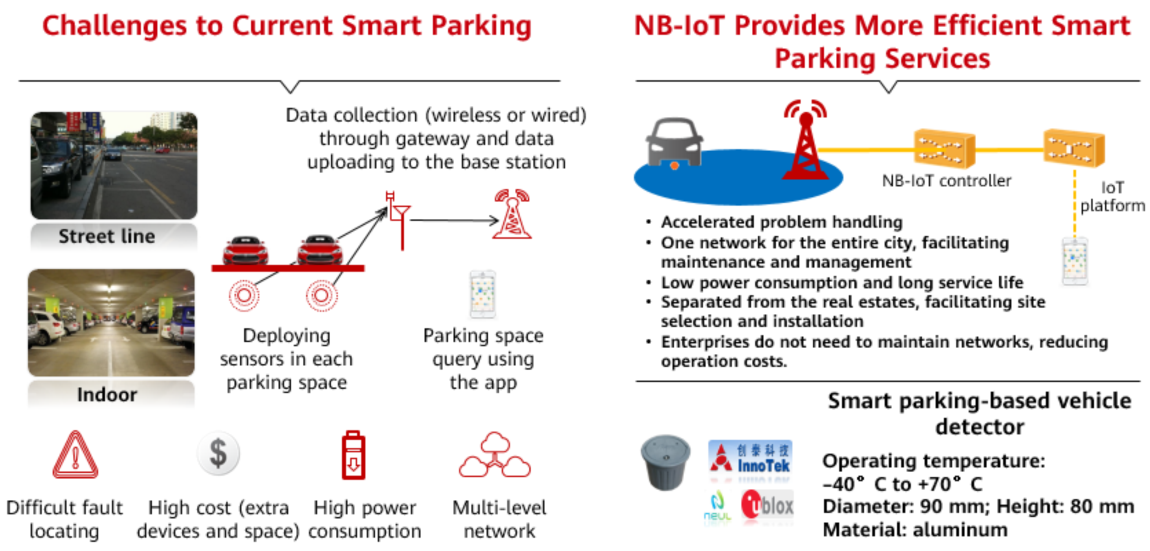

Challenges to Current Smart Parking
NB-IoT Provides More Efficient Smart Parking Services
Street line
Indoor
Data collection (wireless or wired) through gateway and data uploading to the base station
Deploying sensors in each parking space
Parking space query using the app
Difficult fault locating
High cost (extra devices and space)
High power consumption
Multi-level network
NB-IoT controller
IoT platform
Accelerated problem handling
One network for the entire city, facilitating maintenance and management
Low power consumption and long service life
Separated from the real estates, facilitating site selection and installation
Enterprises do not need to maintain networks, reducing operation costs.
Smart parking-based vehicle detector
Operating temperature: -40° C to +70° C
Diameter: 90 mm; Height: 80 mm
Material: aluminum
InnoTek
ublox

**Figure 5-12 Smart parking (see the Appendix for a reference image)**

The first case is smart parking, which is mentioned in the previous section too. Broadly speaking, there are two main types of parking: roadside parking and indoor parking. Roadside or indoor, there are all sorts of issues. For example, it is difficult to locate parking spots due to poor signal. The price of vehicle detectors is driven up by the expensive communications modules they require, and they require a lot of electricity to keep running.

NB-IoT eliminates the need for these expensive communications modules. These NB-IoT modules and base stations can also be used for a range of other applications as well. With NB-IoT, all of the devices in the entire city can be connected to the same network for easier maintenance and management. Thanks to the low power consumption of NB-IoT, the device batteries last a long time.

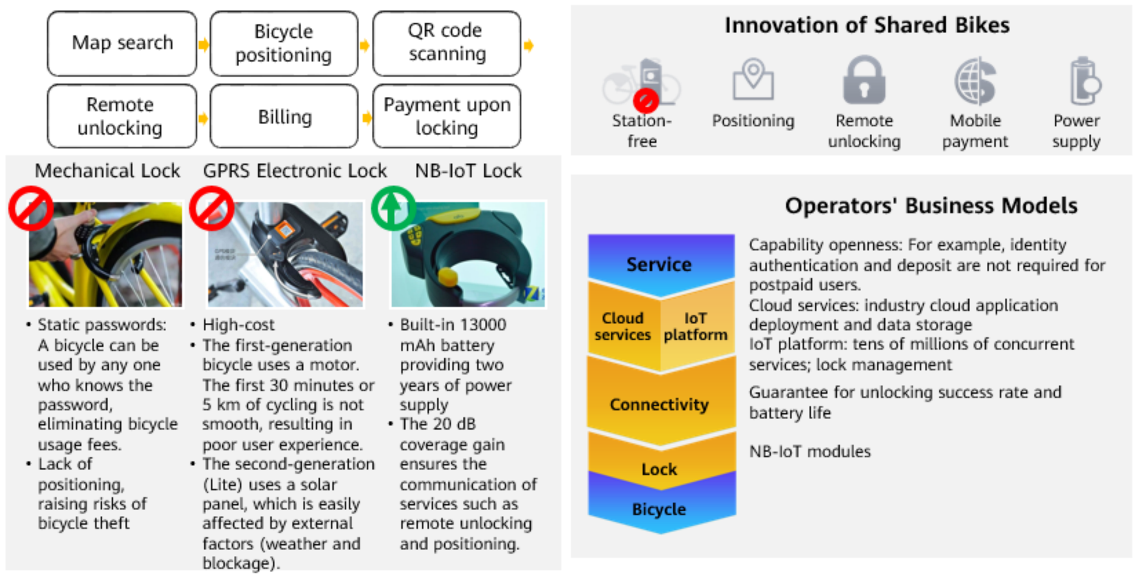

Map search
Bicycle positioning
QR code scanning
Remote unlocking
Billing
Payment upon locking
Mechanical Lock
GPRS Electronic Lock
NB-IoT Lock
Static passwords: A bicycle can be used by any one who knows the password, eliminating bicycle usage fees.
Lack of positioning, raising risks of bicycle theft
High-cost
The first-generation bicycle uses a motor. The first 30 minutes or 5 km of cycling is not smooth, resulting in poor user experience.
The second-generation (Lite) uses a solar panel, which is easily affected by external factors (weather and blockage).
Built-in 13000 mAh battery providing two years of power supply
The 20 dB coverage gain ensures the communication of services such as remote unlocking and positioning.
Innovation of Shared Bikes
Station-free
Positioning
Remote unlocking
Mobile payment
Power supply
Operators' Business Models
Service
Cloud services
IoT platform
Connectivity
Lock
Bicycle
Capability openness: For example, identity authentication and deposit are not required for postpaid users.
Cloud services: industry cloud application deployment and data storage
IoT platform: tens of millions of concurrent services; lock management
Guarantee for unlocking success rate and battery life
NB-IoT modules

Figure 5-13 Shared bikes (see the Appendix for a reference image)

The next use case is shared bikes. Before NB-IoT locks were introduced, bike sharing relied on mechanical locks and GPRS electronic locks. These two types of locks are not easy to use. Mechanical locks rely on static passwords are used. A bicycle can be used by anyone who knows the password. This is also true for GPRS electronic locks, which are expensive and difficult to bill for use, resulting in poor user experience.

To achieve low power consumption and reduce costs, Huawei uses NB-IoT for communications.

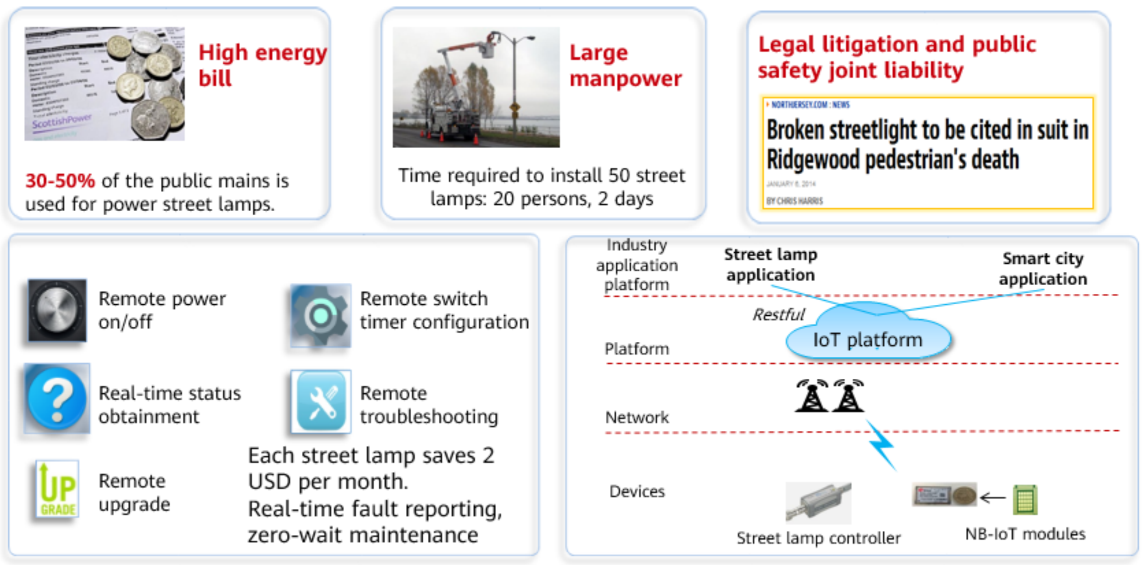

High energy bill
30-50% of the public mains is used for power street lamps.
Large manpower
Time required to install 50 street lamps: 20 persons, 2 days
Legal litigation and public safety joint liability
Broken streetlight to be cited in suit in Ridgewood pedestrian's death
Remote power on/off
Remote switch timer configuration
Real-time status obtainment
Remote troubleshooting
Remote upgrade
Each street lamp saves 2 USD per month.
Real-time fault reporting, zero-wait maintenance
Industry application platform
Street lamp application
Smart city application
Restful
Platform
IoT platform
Network
Devices
Street lamp controller
NB-IoT modules

Figure 5-14 Smart street lamps (see the Appendix for a reference image)

NB-IoT applications are also popular in smart street lamps and smart meter reading. Similar to the previous two cases, NB-IoT reduces the cost of solutions, improves use efficiency, and unifies management for enterprises or companies.

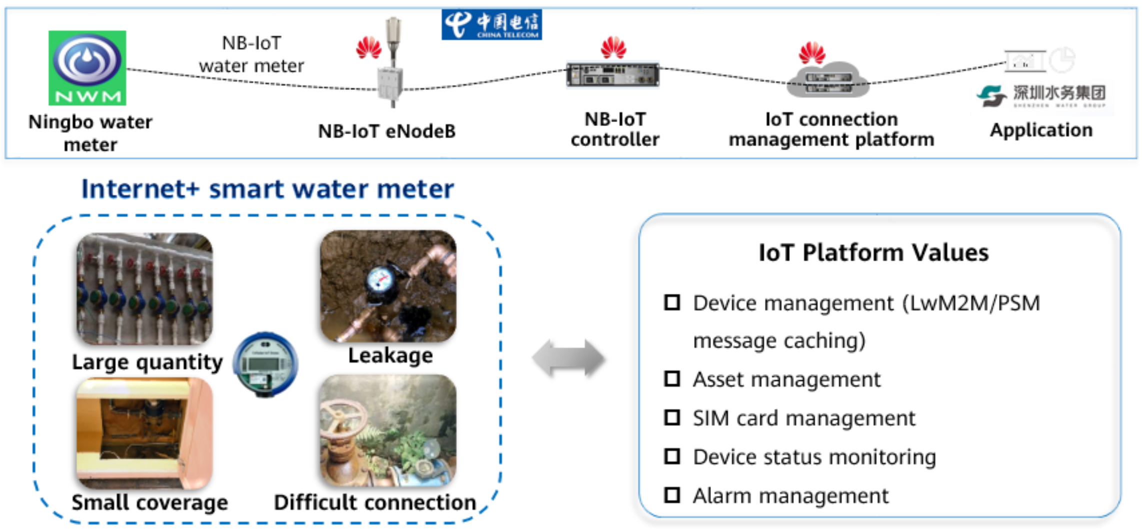


Figure 5-15 Smart meter reading (see the Appendix for a reference image)

# 6 5G Communications Technologies and Solutions

This chapter describes the key technologies and solutions of 5G communications technologies in wireless communications. 5G is a popular cellular wireless communications technology. The launch of 5G paved the way for a big lineup of mobile phones to adopt this technology. This chapter introduces the new technologies and transformations that 5G brings to the mobile Internet and IoT.

## 6.1 5G Standards Evolution and Industry Development

Over the past 10 years, operators and vendors have released digital and intelligent transformation strategies. 5G signals the most recent milestone in digital transformation as we move towards a fully connected, sensing, and intelligent society. 2025 will be a major turning point for the world as envisioned by Huawei's Global Industry Vision (GIV) 2025, which analyzes the future goals of key industries. Some of the findings indicate that:

- Smart home: 14% of the world's households will have smart home robots.
- AR/VR: The number of global VR/AR users will reach 337 million, and the number of enterprises using the same technologies will increase to 10%.
- Smart manufacturing: There will be an estimated 103 robots for every 10,000 employees.
- AI: 97% of large enterprises will use AI to create greater value.

5G is the engine for these changes, and by 2025, 6.5 million base stations will be deployed worldwide to serve 2.8 billion users. These industries will be characterized by all sensing, connected, and intelligent features.

5G will surely steer us to achieving these goals.

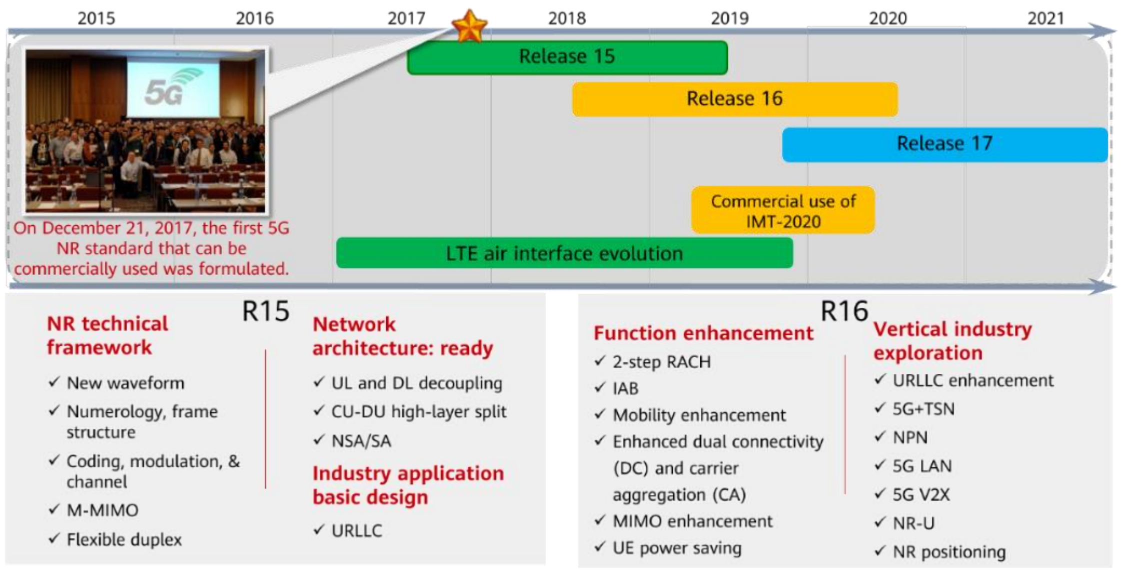

2015
2016
2017
2018
2019
2020
2021
Release 15
Release 16
Release 17
Commercial use of IMT-2020
LTE air interface evolution
On December 21, 2017, the first 5G NR standard that can be commercially used was formulated.
R15
NR technical framework
✓ New waveform
✓ Numerology, frame structure
✓ Coding, modulation, & channel
✓ M-MIMO
✓ Flexible duplex
Network architecture: ready
✓ UL and DL decoupling
✓ CU-DU high-layer split
✓ NSA/SA
Industry application basic design
✓ URLLC
R16
Function enhancement
✓ 2-step RACH
✓ IAB
✓ Mobility enhancement
✓ Enhanced dual connectivity (DC) and carrier aggregation (CA)
✓ MIMO enhancement
✓ UE power saving
Vertical industry exploration
✓ URLLC enhancement
✓ 5G+TSN
✓ NPN
✓ 5G LAN
✓ 5G V2X
✓ NR-U
✓ NR positioning

**Figure 6-1 Evolution of 5G standards (see the Appendix for a reference image)**

5G standards were first introduced in Release 15, which was released in 2017, covering the wireless technical framework, physical layer coding, multi-antenna system, and uplink and downlink network architecture. The following version, Release 16, introduced further enhancements to 5G standards, marking the complete version of 5G.

The evolution of 5G technology can be broken into two parts. The first part is evolution from LTE, specifically from LTE-Advanced to LTE-Advanced Pro, which in simple terms describes evolution from 4.5G to 5G. The second part is new 5G technology, which is we believe that 5G standards start from Release 15. These standards cover two aspects: the first being LTE Advanced Pro evolution and EPC evolution, and the second being 5G new radio and 5G core network.

Another advantage of 5G is that there is one unified global standard. In the 3G era, the top three standards commonly used in China are different. China Telecom uses CDMA2000, China Unicom uses WCDMA, and China Mobile uses TD-SCDMA. So, how does that affect us? Well, when you buy a new mobile phone, you need to be aware of whether the phone is SIM locked to a certain operator, otherwise other network SIM cards may be incompatible with the phone.

SIM-locked phones are typically cheaper than unlocked phones, so you need to be aware of this when buying a new phone.

The 4G era saw only two standards: LTE TDD and LTE FDD. And now in the 5G era we only have one, making mobile communications even more open and convenient.

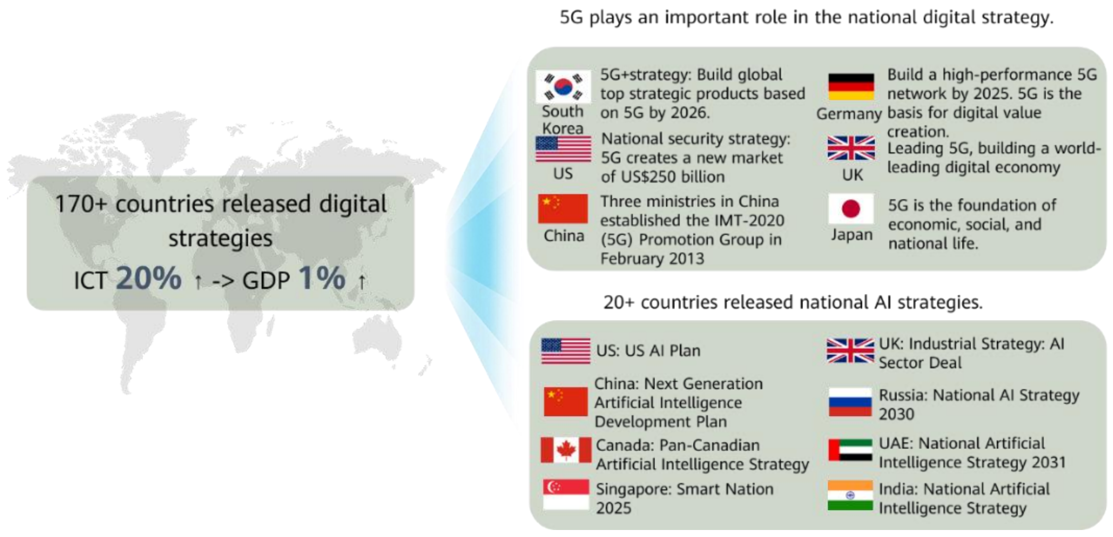

5G plays an important role in the national digital strategy.
170+ countries released digital strategies
ICT 20% ↑ -> GDP 1% ↑
South Korea
5G+strategy: Build global top strategic products based on 5G by 2026.
US
National security strategy: 5G creates a new market of US$250 billion
China
Three ministries in China established the IMT-2020 (5G) Promotion Group in February 2013
Germany
Build a high-performance 5G network by 2025. 5G is the basis for digital value creation.
UK
Leading 5G, building a world-leading digital economy
Japan
5G is the foundation of economic, social, and national life.
20+ countries released national AI strategies.
US: US AI Plan
China: Next Generation Artificial Intelligence Development Plan
Canada: Pan-Canadian Artificial Intelligence Strategy
Singapore: Smart Nation 2025
UK: Industrial Strategy: AI Sector Deal
Russia: National AI Strategy 2030
UAE: National Artificial Intelligence Strategy 2031
India: National Artificial Intelligence Strategy

**Figure 6-2 National strategies for 5G (see the Appendix for a reference image)**

Next, let us see how important 5G technologies are at the national level. As shown in Figure 6-2, many countries are aware of the benefits that 5G can bring to their countries, with many countries labeling 5G and AI as core strategies for achieving digitalization.

Statistics indicate that the most effective way to stimulate GDP growth is to invest more in ICT. According to research conducted by Huawei, 20% of ICT investment will bring 1% of GDP growth, highlighting the remarkable power of ICT investment. A large number of countries have realized the importance of 5G and AI, and regard them as key national development strategies. 5G construction is valuable for both operators and national infrastructure.

During the first half of 2018 there were only five 5G networks in the world, but this number increased to 30 in the second half of the same year, proving the rapid growth rate of 5G networks. By the end of 2019, China had 130,000 domestic 5G base stations, delivering 5G coverage across cities such as Beijing, Shanghai, and Guangzhou. This shows that countries and even the world attach great importance to 5G development.

Although rolling out 5G networks is a major step, it is meaningless without compatible products to go with it. In February 2018, Huawei released the world's first 5G commercial chip, Balong 5G 01, and corresponding UE, Huawei 5G CPE, which has an actual downlink rate of 2 Gbit/s. The Balong chip was then used in Huawei's maiden 5G foldable mobile phone, the Huawei Mate X. Following this, Huawei integrated 5G modules into the system-on-a-chip and launched Kirin 980, Kirin 990, and other chips, as well as releasing its latest 5G mobile phone, Huawei Mate 40.

## 6.2 Key 5G Technologies

The key performance indicators of 5G are the capabilities required in the three 5G general scenarios mentioned in earlier sections: 1 ms latency, 10 Gbit/s rate, and 1 million connections per square kilometer. The slicing-based network architecture is also a 5G capability, which will be detailed in a subsequent sub-section.

## 6.2.1 New Architecture

The key technologies of 5G can be summarized into three key innovations: new architecture, new radio, and full spectrum. These innovations will be elaborated on in their own sub-sections. In 5G, a single network carries services for hundreds of industries for a wide range of applications that pose a variety of network demands. To address this challenge for a single network, operators define combinations of requirements (including latency and bandwidth) to shape multiple slices with different capabilities to adapt to service requirements. Therefore, the 5G network must be flexible if slicing is required.

### 6.2.1.1 Network Slicing

Network Function Virtualization (NFV) technology is used in the 5G core network. NFV uses universal hardware (such as x86) and virtualization techniques for software processing of many functions, reducing the costs of expensive network devices. Using software-hardware decoupling and function abstraction, network functions become independent of dedicated hardware, and flexible resource sharing is enabled to promote quick development and deployment of new services. In addition, automatic deployment, auto-scaling, fault isolation, and self-healing are achievable to meet service requirements.

In other words, network functions are no longer restricted by hardware. In the past, the purchase of hardware resources (such as memory and CPU) determined the upper limit available for users. Usually, users purchased massive hardware resources to ensure sufficiency. However, this led to a waste of resources as such a large quantity of resources might not have been fully utilized in normal operation. A virtualized architecture is not constrained by hardware. It flexibly allocates the same hardware to different users. You can occupy resources on demand, and available resources can be used by others.

A virtualized network can process different services based on the requirements of slices. Software-defined networking (SDN) operates in a similar way. SDN defines a network by software to make it more intelligent and flexible. The core technology, OpenFlow, separates the control plane from the data plane on network devices to flexibly control traffic on the network. SDN makes the network more intelligent as a pipe and provides a favorable platform for innovations on the core network and applications.

We can see that, in 5G, network slicing creates multiple virtual E2E networks on one set of universal hardware. Each virtual network offers specific functions to meet service requirements.

### 6.2.1.2 NSA&SA Networking

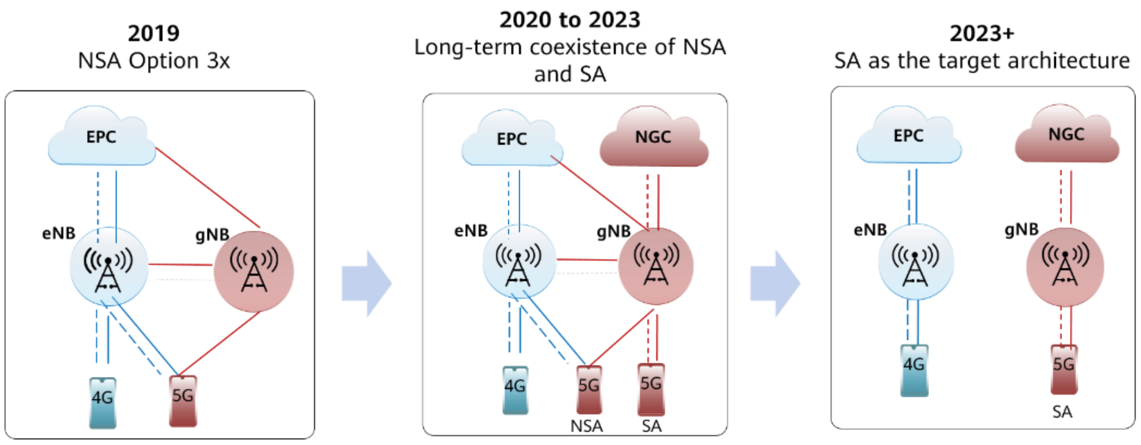

2019
NSA Option 3x
2020 to 2023
Long-term coexistence of NSA and SA
2023+
SA as the target architecture
EPC
NGC
eNB
gNB
4G
5G
NSA
SA

**Figure 6-3 5G networking mode (see the Appendix for a reference image)**

5G networking includes non-standalone (NSA) networking and standalone (SA) networking. As shown in Figure 6-3, the NSA network architecture consists of the 4G core network (EPC), 4G base station (eNodeB) and 5G base station (gNodeB). In this architecture, 4G and 5G base stations are controlled by the 4G core network. The SA network architecture consists of the 4G base station (eNodeB), 5G base station (gNodeB), 4G core network (EPC), and 5G core network (NGC).

The difference between the two networking modes is complex. I will use a hotel business to describe the difference. You are the owner of a hotel (called Hotel 4), and you have a chef (called Chef 4). As the business grows, the hotel attracts a lot of guests and you want to expand your business. But this is costly and you have to invest wisely. You have two solutions. Solution A is to buy another hotel (called Hotel 5) and employ another chef (called Chef 5). This is the simplest, most direct, but most costly solution. Solution B is to buy another hotel (Hotel 5) but not hire more chefs. Chef 4 is responsible for the two hotels. This solution saves money but is more complex to implement, and Chef 4 may be too busy to handle two hotels.

In the example, Hotel 4 and Hotel 5 represent 4G and 5G base stations, respectively; Chef 4 and Chef 5 represent the 4G and 5G core networks, respectively; Solution A and Solution B represent SA networking and NSA networking, respectively. Essentially, the fundamental difference between the two networking modes is the same as that between the two solutions in the example: the costs.

It is easy to have a pure and perfect 5G network provided that the expenses can be paid. All of the new 5G devices are deployed in a one-stop manner and run 5G services completely separate from the 4G network. This is how SA networking works. However, not all operators are tycoons. To help subscribers gradually enjoy 5G services, 3GPP proposes NSA networking. In a sense, SA and NSA networking are two service packages offered to operators.

### 6.2.1.3 5G Core Network (NGC)

The 5G core network has four technical features. First, the control plane and user plane are separated from each other. Why is this separation beneficial? The control plane is tasked with transmitting the signaling messages and the user plane is tasked with transmitting the service data actually used by users. With the two planes separated, software management and upgrade are greatly simplified for operators as they can easily locate problems on the control or user plane. The second technical feature is mobile edge computing (MEC). MEC enables cloud computing capabilities at the edge of a cellular network and is implemented by a cloud server running specific tasks. That is, MEC transfers some services of the core network onto the cloud server at the edge of the cellular network. This mechanism effectively reduces the bidirectional transmission time on the network, minimizing latency and saving bandwidth.

The third is network function reconstruction. Cloud computing is enabled across the entire 5G network, and the network architecture and functions of each part are replanned. In this sense, function reconstruction is a feature of the 5G network. The fourth and final feature is network slicing. As mentioned above, a network slice is an E2E network that can be flexibly customized to meet specific requirements.

### 6.2.2 New Radio

New radio involves having knowledge of the physical layer and will not be explained in detail here. To know more about new radio, you can continue to learn the course on 5G. In the course on 5G IoT, you only need to know about the new radio technologies used by 5G, including full duplex, massive MIMO, Polar coding, F-OFDM, and SCMA. In massive MIMO, multiple antennas are used at both the transmit and receive ends. In addition to the time and frequency domains, the space domain also uses multiplexing to improve the throughput.

### 6.2.3 Full Spectrum

As mentioned above, 3G uses the 1800 MHz band and 4G uses the 2600 MHz band. 5G directly jumps to the C-band, that is, 3.4 to 3.6 GHz. Although the band used by 4G is not a low-frequency band (2600 MHz means 2.6 GHz), it can still reach a coverage area of approximately two kilometers by means of signal amplification of the base station. However, this is not the case for 5G. In addition to the C-band, the millimeter wave (mmWave) band above 6 GHz is available in 5G. This results in weak coverage and penetration, as well as high energy consumption on the 5G network.

For 5G, the C-band has lower frequencies compared with the mmWave band and therefore the C-band is used for 5G base stations to provide comprehensive coverage (that is, outdoor coverage). Other available bands in 5G are used for capacity supplement and self-backhaul. A common 5G base station using the C-band provides a limited coverage area and serves a limited number of users. Therefore, the user data rates are very likely to decline to ensure successful network access for as many users as possible, if the base station serves an excessive number of users. This is often the case in large indoor densely-populated areas with few obstacles (such as transportation hubs and stadiums). In this case, a micro base station operating on a band above 6 GHz is used to expand capacity. Although high-frequency signals have poor penetration performance, their performance in reflection is relatively favorable. Therefore, high-frequency signals are suitable for these scenarios.

Today, fiber to the home (FTTH), for example, may still use optical fiber connections between the operator's equipment and the home or business. Next, operators will switch to a wireless connection to achieve wireless backhaul. An example of a wireless device involved is CPE, as mentioned above. The CPE functions as a base station to provide services for terminals, and also replaces optical fibers to achieve wireless self-backhaul. This is how fixed wireless access (FWA) works, as described in a later section. Overall, 5G deployment is a hybrid networking approach that combines high and low frequency bands. It aggregates all frequency bands and spectrums and uses them in different directions to provide a better 5G performance.

## 6.3 Three 5G Application Dimensions

### 6.3.1 eMBB

As covered in the preceding section, enhanced Mobile Broadband (eMBB) equips 5G with ultra-high transmission rates, which resonates most with you. eMBB can be applied on virtual reality (VR), augmented reality (AR), and mixed reality (MR). With VR, you use devices (such as VR glasses) to immerse yourself in a virtual environment.

AR, also through devices (glasses), combines virtual objects with reality for interactive experience. With MR, no glasses are required so you can project images in the air and control them by directly touching them.

A major advantage of AR and VR devices is that they are available off the shelf, but a drawback is that they are expensive and vary in quality. Thanks to ultra-high transmission rates and cloud-based 5G networks, you can enjoy VR without purchasing high-end computers. Ultra-high transmission rates ensure fast and reliable transmission of ultra HD VR images from the cloud to your device, ensuring that you stay immersed.

### 6.3.2 mMTC

By leveraging Massive Machine-Type Communications (mMTC), which is a dimension of IoT for massive connections, 5G can achieve 1 million connections per square kilometer. Meanwhile, the standardization of NB-IoT has seen 5G branch out from merely carrying high-speed and low-latency services as before. Though further development is not much for massive connection technologies, the recognition of NB-IoT is an important step to incorporating massive connections into 5G.

### 6.3.3 URLLC

Ultra-reliable low-latency communication (URLLC) underpins ultra-low latency services, as previously discussed, such as autonomous driving and remote surgery. Why is low latency important for these services? A high latency in autonomous driving, even if it is only dozens of milliseconds, can result in a long braking distance and even catastrophes.

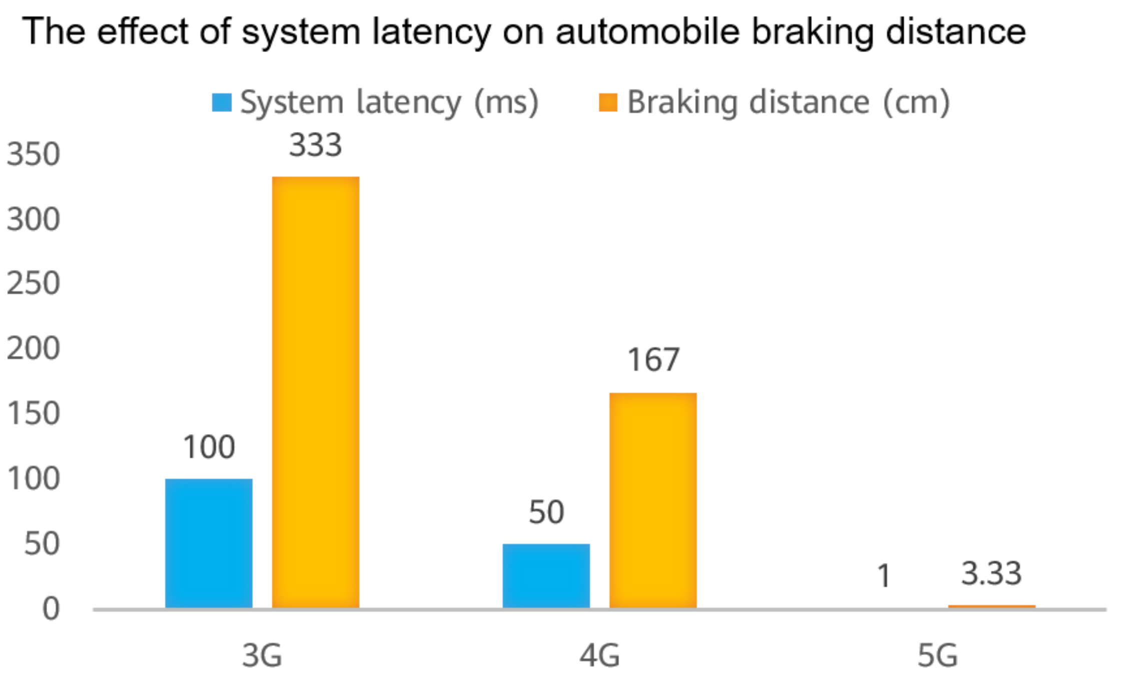


**Figure 6-4 The effect of system latency on automobile braking distance**

Here are some examples in practice. If a car drives at 80 km/h, the braking distance is about 333 cm under a 3G network latency of 100 ms. Likewise, the braking distance is about 167 cm under a 4G network latency of 50 ms, but with a 5G network latency of 1 ms the braking distance is dramatically cut to merely 3.33 cm.

Therefore, low latency is crucial for applications that require high security. In a remote surgery setting, the surgeon must obtain and monitor vital data of the patient immediately.

## 6.4 5G Commercial Solutions

Huawei continues to provide excellent B2C, B2B, and B2H solutions for its carriers, which will be introduced in this section.

### 6.4.1 B2C Solutions

According to business insights, there are three key drivers for carrier success in the B2C market, and these are high-quality networks, abundant contents, and flexible tariff plans. As shown in Figure 6-5, it took nine years and six years for 3G and 4G to reach 500 million subscribers, respectively. Based on this, Huawei forecasts that it will take only three years for 5G to achieve this same feat.

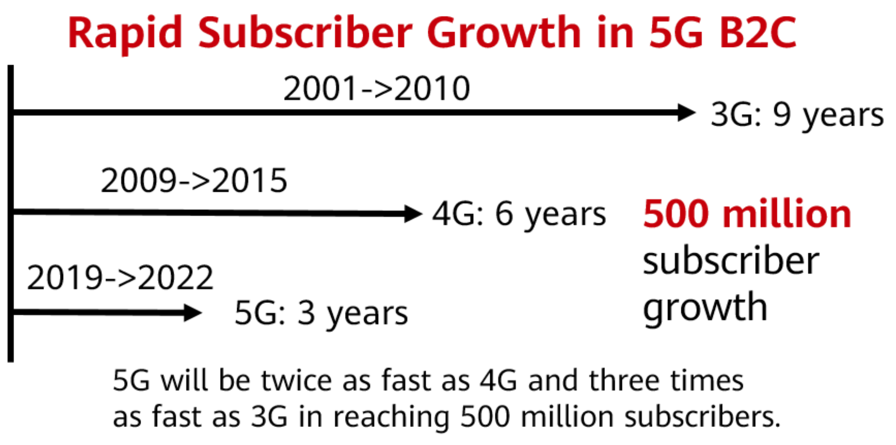

Rapid Subscriber Growth in 5G B2C
2001->2010
3G: 9 years
2009->2015
4G: 6 years
2019->2022
5G: 3 years
500 million subscriber growth
5G will be twice as fast as 4G and three times as fast as 3G in reaching 500 million subscribers.

**Figure 6-5 Time required for hitting 500 million B2C subscribers**

In the B2C market, 5G has a considerable number of potential subscribers. Within this field, high-quality networks are in high demand among consumers and are key to a carrier's success. Two networking options are available: non-standalone (NSA) and standalone (SA). As SA networking is able to deliver a superior 5G experience, carriers are able to both appeal to and fulfill consumer demands for high-quality networks.

On top of this, 5G is well equipped to carry a large number of new services, including new types of videos, live broadcast, and online gaming. These services open up new experiences for consumers, satisfying the demand for abundant content.

Apart from the large quantity of subscribers and new services, the final key is flexible tailored tariff plans. Consumers are most concerned with charging standards and tariffs, so by offering tailored tariff plans (for example, carriers can charge live broadcast subscribers for high uplink bandwidth and gaming subscribers for low latency and high downlink bandwidth) to consumers, carriers can ensure consumers are getting the service that they want.

Huawei provides its own VR and AR services. You can use VR glasses to connect to a VR cloud platform over a high-speed 5G network. This technology enables carriers to provide their own VR services, including VR music, VR gaming, and VR live broadcast, that are centrally managed by a cloud platform. This solution aims to help carriers develop fully converged video services and build core competitiveness to swing with the video era.

### 6.4.2 B2B Solutions

Industry research has identified an urgent need for helping carriers undertake 5G projects, which is seen as a major profit direction in the industry. Connections act as a potential springboard for embracing a "blue ocean".

Despite a huge 5G market capacity for B2B services, 5G B2B industry applications are still held back by immature business models. B2B services need to be developed step by step, starting from connections with mature business models and gradually building up to cloud and industry applications. Therefore, carriers might begin by targeting at typical scenarios such as private lines, private networks, and campuses, before replicating these business models and moving onto more comprehensive scenarios of ecosystem aggregation between networks and vertical industries.

A business solution can encompass many different scenarios. The AR 5G connection solution in the fixed private line scenario is used for enterprise private lines to offer secure VPN connections that are deployed quickly and economically. In addition, cameras with built-in 5G modules facilitate camera installation and provide HD video surveillance services for enterprises or public utilities.

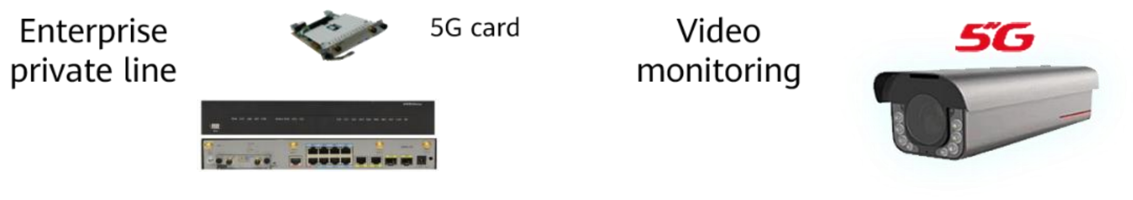


**Figure 6-6 5G fixed private line**

Industry CPE for remote control in the wireless private line scenario provides connection stability in complex environments. Moreover, backed by 5G's low latency, the CPE can remotely control high-precision instruments. In the live broadcast scenario, Huawei launched the video codec with built-in 5G modules, making 5G backpacks lighter and battery life longer. This improves the convenience of shooting HD videos outdoors for journalists.

**Figure 6-7 5G wireless private line**

## 6.4.3 B2H Solutions

How can carriers achieve success in B2H? As previously covered, Huawei's fixed wireless access (FWA) B2H solution is the answer. In fiber to the home (FTTH), optical fibers wire base stations to routers, and the routers provide you with wireless connectivity. However, to provide wireless services on 5G networks, carriers wirelessly connect base stations to routers at home, rather than through optical fibers.

In this case, there are three types of customers for carriers: households without network connections, households using copper networks, and households using optical networks. FWA is applicable to all three types of households for 5G networks.

5G CPEs help overcome this challenge. For example, mmWave CPEs cover dense urban areas, and common CPEs cover common urban areas, meaning various types of CPEs cater for your network requirements.

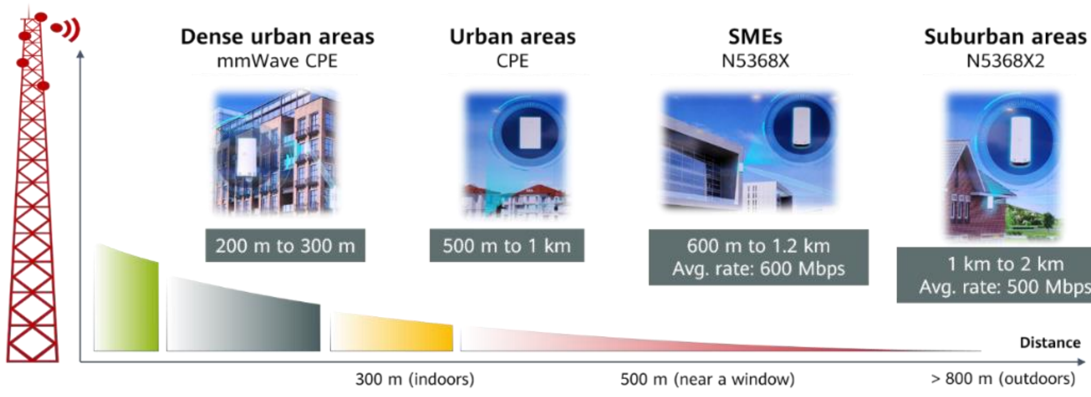


**Figure 6-8 FWA solution (see the Appendix for a reference image)**

# 7 Huawei Industrial IoT Gateways

This chapter describes the technical details of gateways for fixed network scenarios in the industrial field. The industrial field has always been critical for the IoT sector as it is closely related to production with many "things" involved in the process — for example, machines and robotic arms. In addition, the environments are harsher than in other IoT scenarios and require the use of a wide range of complex and demanding devices. As such, diverse interfaces need to be adapted to the industrial field. With the development of global industrial automation, industrial IoT devices that meet the preceding requirements play an increasingly pivotal role.

## 7.1 Overview of Industrial IoT Gateways

IoT is widely used in industrial scenarios, which includes not only factories, but also sub-scenarios such as electric power and transportation. For example, smart meter, smart power distribution, and smart transportation also fall under the industrial field category.

### 7.1.1 Challenges Faced by IoT in the Industrial Field

- Harsh environment: Temperatures can range from extremely high to extremely low, necessitating the need for IoT devices to be able to withstand temperatures from –40°C to +70°C. Furthermore, these devices are expected to be dustproof, waterproof, and resistant to electromagnetic interference. Environments such as power plants have very strong electromagnetic interference, which affects communication between devices and renders conventional enterprise-class routers ineffective.
- Diverse interfaces and protocols: There are a wide range of devices with different interfaces and protocols in the industrial field. Therefore, a device which can adapt to all these protocols and interfaces is required.
- Cyber security: Business secrets are invaluable to enterprises, making cyber security a top priority. Although most enterprises use private networks, they are still vulnerable to malicious attacks. Therefore, cyber and information security is a key consideration in the industrial field.
- Complex O&M: O&M complexity is proportional to the number of terminals. With a large number of terminals in the industrial field, maintenance is costly if experts are needed every time an issue occurs. As such, a simple and easy-to-use system is required to lighten the workload for O&M personnel and lower O&M expenses for enterprises.

## 7.1.2 Why Industrial IoT Gateways?

An easy-to-use and powerful IoT gateway is crucial to address the preceding challenges. What functions does an industrial IoT gateway provide? As shown in Figure 7-1, the IoT gateway is located between the terminal layer and the network layer and serves as a bridge between the two. It is connected to terminals on the left and connected to the network on the right. In addition to providing upstream transmission channels for terminals, the IoT gateway also offers edge computing capabilities, which will be described in detail later.

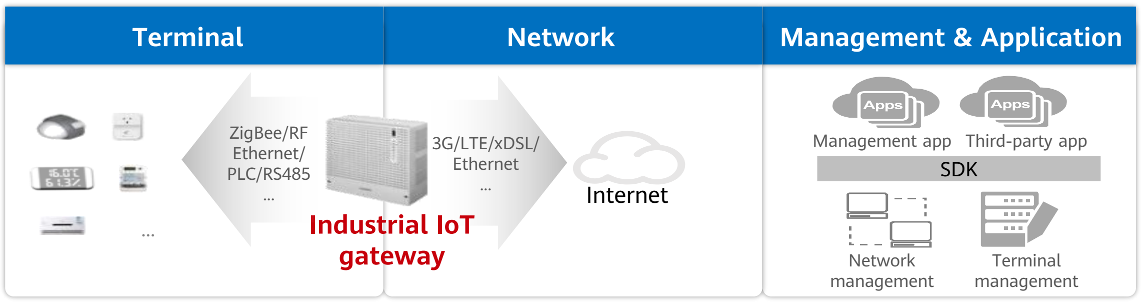


**Figure 7-1 Industrial IoT gateway**

The industrial IoT gateway provides a protocol conversion function. There are many protocols in the industrial field — covering all sectors — so conversion on the IoT gateway is needed before data is transmitted.

## 7.2 Edge Computing

What is edge computing? To put it simply, some of the calculations that are originally performed on the central node are performed on the edge node. Traditionally, the platform layer processes and calculates all data. But now, the IoT gateway can calculate some of the data that is not important and promptly provide the calculated data to the terminals. The result is lower latency, greater user privacy, and lowered costs. This is because for the central node, not all the data collected is useful, so some unnecessary data can be processed by the edge node. In this way, the edge node relieves some of the pressure at the central node.

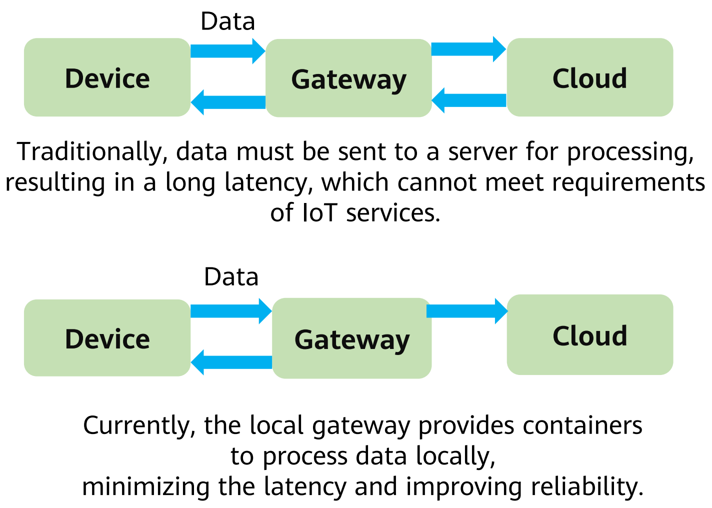

Data
Device
Gateway
Cloud
Traditionally, data must be sent to a server for processing, resulting in a long latency, which cannot meet requirements of IoT services.
Data
Device
Gateway
Cloud
Currently, the local gateway provides containers to process data locally, minimizing the latency and improving reliability.

**Figure 7-2 Implementation of edge computing**

In the past, data is processed as shown in Figure 7-2 where data is generated on devices and uploaded to gateways. However, the gateways did not have any computing capabilities and could only function as a transmission device to upload data to the cloud for processing. Now, as gateways themselves have computing capabilities, they can locally process some data with low requirements, instead of transmitting it to the central node. After the data is processed, the gateway still needs to notify the central node that the data has been processed.

Edge computing is an open platform that integrates connection, computing, storage, control, and application functions. Compared to the architecture of IoT, the architecture of edge computing can be divided into four domains: device, network, data, and application. Although edge computing is located between the sensing layer and network layer, the edge computing architecture can be divided into four layers to provide capabilities similar to the four-layer architecture of IoT.

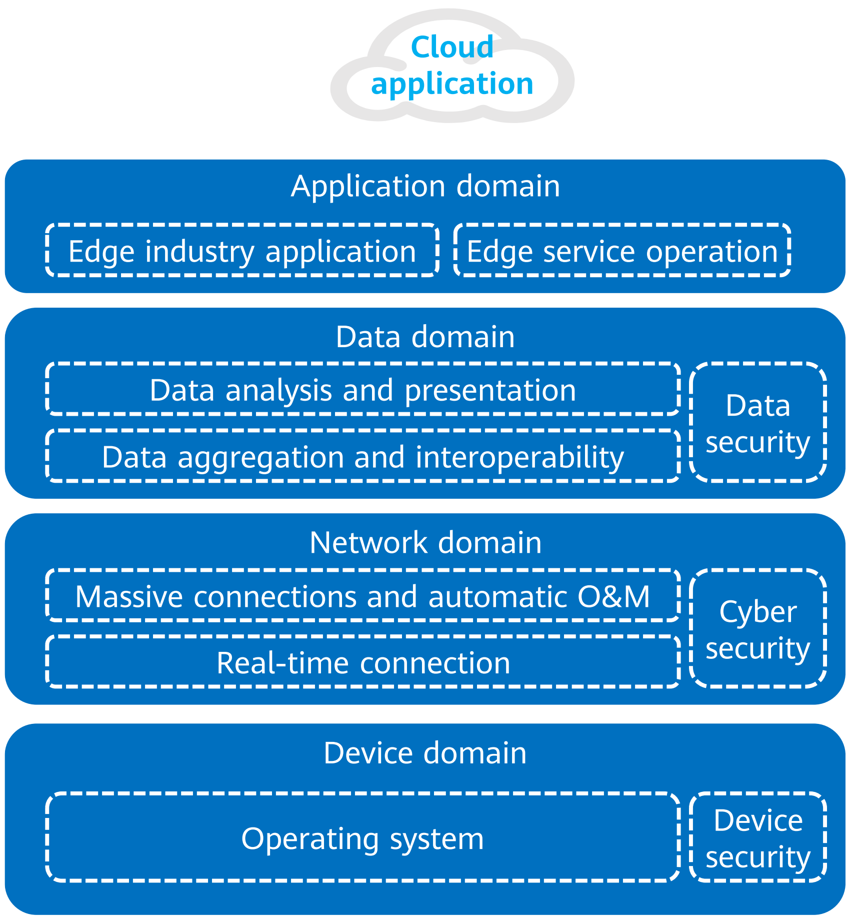


**Figure 7-3 Edge computing architecture**

The two layers between the device and gateway in the IoT architecture are further divided into four layers. In the hierarchical edge computing architecture, the device domain is equivalent to the sensing layer; the network domain refers to the network between devices and gateways; the data domain refers to gateways, which process data in a similar way to an IoT platform; and the application domain refers to the diverse applications in edge computing.

## 7.3 Network Topology

A network topology refers to the layout of computers and devices on a communication network, representing the physical or logical arrangement of network elements (NEs). If two networks have the same connection model, their network topologies are of the same type, though the physical connections inside the two networks and the distances between nodes may be different. This section describes common network topologies as well as mesh — a networking technology widely used in the IoT field.

## 7.3.1 Star Topology

The star topology is a topology in which all nodes are connected through a central network device such as a hub or a switch.

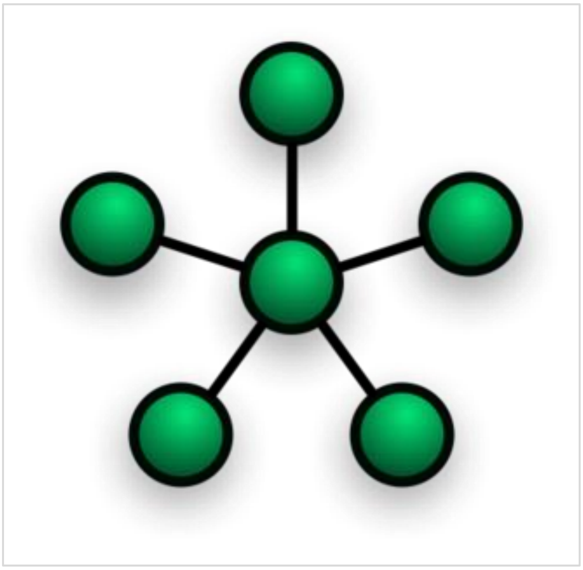


**Figure 7-4 Star topology**

Basic features:

- Easy implementation with heavy installation and maintenance workloads and high costs: Typically, twisted pairs or coaxial cables are used for transmission. Each site needs to be directly connected to a central network device, which requires a large number of cables, resulting in heavy workloads for installation and maintenance.
- Easy node expansion and movement: To add a node, you only need to connect the node to the hub or switch using a cable. To move a node, you only need to move the node to the new position as needed.
- Easy fault diagnosis and isolation: If one node is faulty, connections of the other nodes are not affected, and the faulty node can be removed as needed.
- Potential bottleneck on the central node and low distributed processing capability of each site: The load of the central node is heavy, which may cause a bottleneck. When the central node fails, the entire network is affected.

Advantages:

- Simple network structure streamlines management, maintenance, and commissioning.
- Easy control enables the quick addition or removal of sites.
- Centralized management simplifies service provisioning and network reconfiguration.
- Direct connection of each node to the central node facilitates fault diagnosis and isolation.

Disadvantages:

- Low line utilization: A line is used only by the central node and the node on the line.

- Heavy workload of the central node: When the central node is faulty, the entire network fails, imposing stringent requirements on the reliability and redundancy of the central node.
- High installation and maintenance costs: A large number of cables are needed.

### 7.3.2 Ring Topology

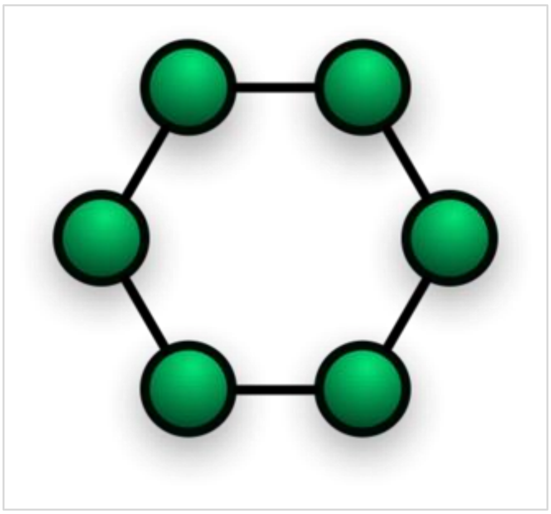


**Figure 7-5 Ring topology**

The ring topology is widely used on LANs. On a ring network, packets of data travel from one device to the next until they reach the destination. This structure eliminates the dependency on a central node for communication between end users.

The ring topology has the following characteristics:

- Information flows towards a fixed direction on a network with only one path between two nodes, simplifying path selection.
- The control software is simple and easy to use.
- Information is transmitted over the nodes on the ring network sequentially. If there are a large number of intermediate nodes, the information transmission rate will be lowered, increasing the network response time.
- The ring network is closed, making expansions difficult.
- A faulty node will cause the entire network to break down, resulting in low reliability.
- Faults on branch nodes are hard to locate.

### 7.3.3 Bus Topology

Bus topology (or bus network) is commonly referred to as a "linear bus" because all the nodes are physically connected in a straight line.

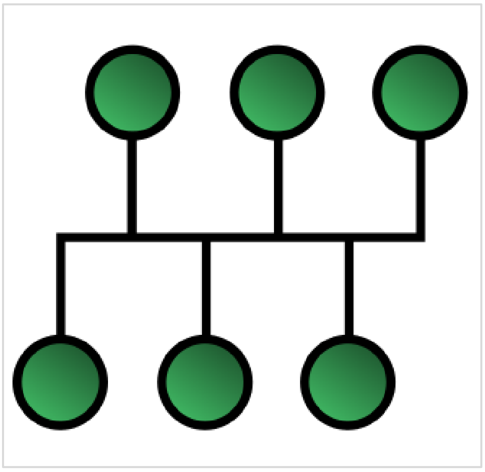


**Figure 7-6 Bus topology**

The bus topology has the following characteristics:

- Simple structure facilitates expansion. To add a node, only one branch interface needs to be added on the bus to connect to the branch node. When the maximum bus load is reached, other buses can be expanded.
- Only a small number of cables are needed, simplifying installation.
- Simple and reliable devices are used.
- Faults on branch nodes are hard to locate.

Advantages:

- Easily connects to a computer or peripheral linear bus.
- Requires shorter cables than the star topology.
- Suitable for small networks.

Disadvantages:

- The entire network will shut down if there is a break in the main cable.
- Terminators are required at both ends of the main cable.
- Difficult to identify the root cause of network shutdowns.
- The transmission speed slows when a large number of devices are added to the network.

### 7.3.4 Mesh Topology

A wireless mesh network (or multi-hop network) is a new wireless network technology that is radically different from a traditional wireless network. It does not rely on a preset infrastructure — allowing for temporary networking and rapid deployment — and is resilient to damage, even without a control center.

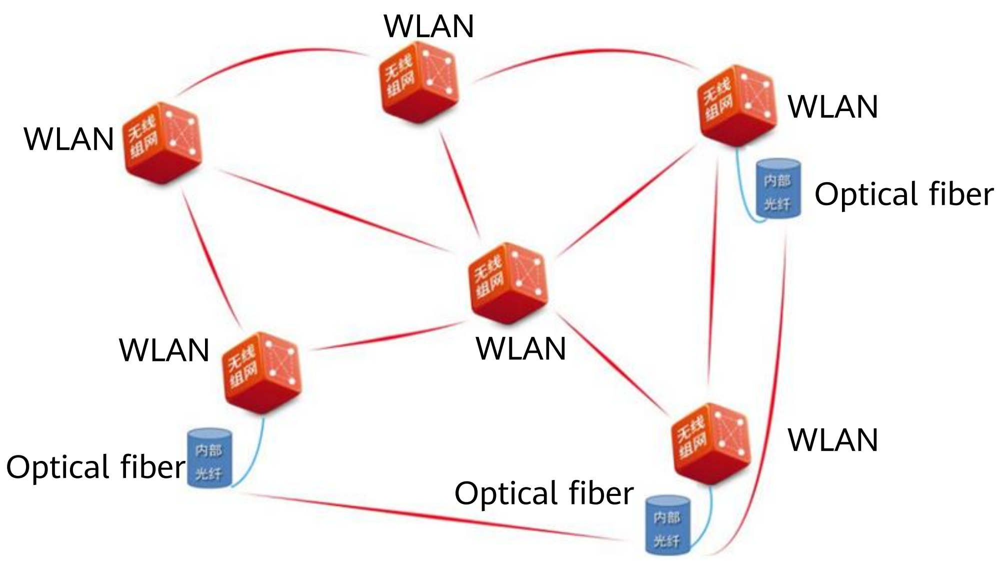


**Figure 7-7 Mesh topology**

A wireless mesh network uses a multipoint-to-multipoint network topology called mesh. In this topology, network nodes are connected in a wireless multi-hop manner through adjacent network nodes.

Mesh technology can relay signals, extend radio signal coverage, and support network self-organization, self-recovery, and traffic self-balancing. Therefore, mesh technology is recommended for RF networks.

Mesh technology can significantly increase the network bandwidth:

Due to the physical characteristics of wireless communications, a shorter transmission distance of a radio signal makes it easier to achieve a higher bandwidth. This is because, the longer the wireless transmission distance, the higher the likelihood of data loss from a number of factors such as interference. If multiple short hops are used to transmit data, the bandwidth of each hop increases, improving the total bandwidth.

On a mesh network, each node sends and receives information, and functions as a router to forward information to neighboring nodes. With a growing number of interconnected nodes and more paths for data transmission, the total bandwidth increases.

# 8 Home IoT Gateways

This chapter builds on the previous chapter and describes the application of home IoT gateways in home scenarios. Although many smart home appliances exist on the market, a smart device is needed to control all these devices, and this is known as a smart home gateway. A smart home gateway is the heart of a smart home and is also the unified egress of smart devices, which highlights its importance in the home.

## 8.1 Home IoT Gateway Overview

### 8.1.1 Home Network Development

Before smart devices exploded onto the scene, a simple home network set-up consisted of a desktop computer connected to the Internet through a dial-up modem, which was a simpler, point-to-point structure. Now, the popularization of FTTH has led to optical modems being widely used in homes, with which wireless Internet has become a mainstay. Not only has accessing the Internet become easier, but also the devices we use to access the Internet has become smaller and more diverse, such as computers, smartphones, tablets, and set-top boxes. A typical home network nowadays is more representative of the star structure described in the last chapter.

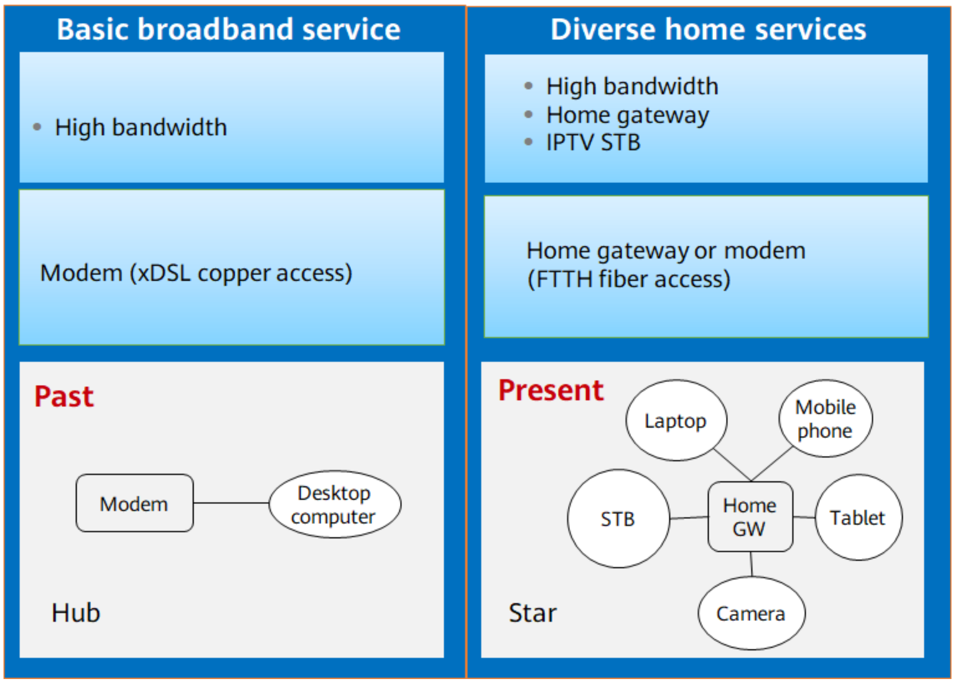


**Figure 8-1 Basic broadband service and diverse home service (see the Appendix for a reference image)**

In the future, a smart home gateway, or ONT (as shown in Figure 8-2) will act as the home terminal, whereby almost all home devices will connect to the Internet through the smart ONT, creating a complete smart home system. This in turn will form a mesh network. It lacks a control center, so all devices can be intelligently connected to other devices to exchange data. All future networks are expected to evolve towards this kind of structure.

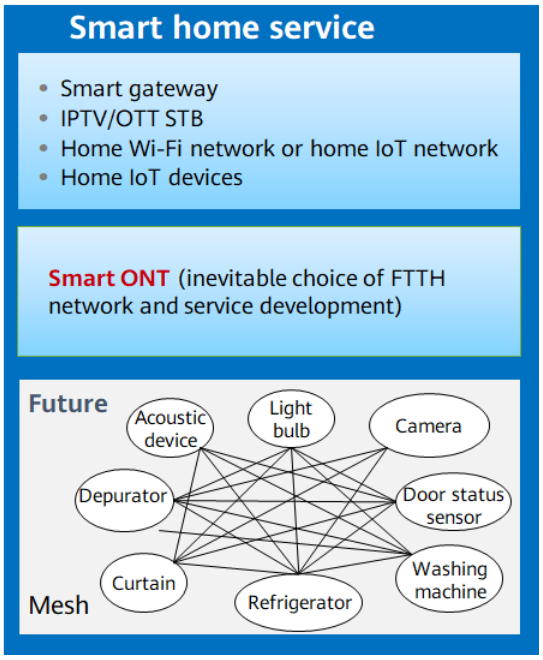


**Figure 8-2 Smart home service (see the Appendix for a reference image)**

The smartphone revolution in 2006 not only boosted the development of wireless communications technologies, but also kick started the development of home networks. As phones became smarter and tablets were introduced to consumers, a home terminal was needed to connect them all to the Internet. On top of this, fixed broadband gradually evolved to fiber access, meaning smart ONTs not only provide high-speed Internet, but also lay the foundation for the development of other home services.

## 8.1.2 Positioning of Smart Home Gateways

The smart home network will continue to run over a broadband connection, but one thing that will change is how IoT devices within the home connect to the Internet. Although most devices will connect to the home network through Wi-Fi, certain devices, such as smart fridges and door locks, will connect through ZigBee or Z-Wave.

For an environment that is home to various types of connections, a Wi-Fi router is simply no longer enough. Rather, a smart gateway that can connect devices at different standards and implement unified management and control is therefore needed.

## 8.1.3 Key Services of Smart Home Gateways

A smart home gateway enables smart acceleration, giving home users who watch and download videos, or play online games the power to optimize speeds for these services. This can be achieved by installing a smart acceleration plug-in that will accelerate video streaming and download speeds, as well as reduce latency for online gaming.

Another important service is seamless Wi-Fi coverage throughout the whole home. Although some home IoT devices will use different standards and networks, 80% of them will still be connected through Wi-Fi, and this simply highlights the reliance on Wi-Fi in the home.

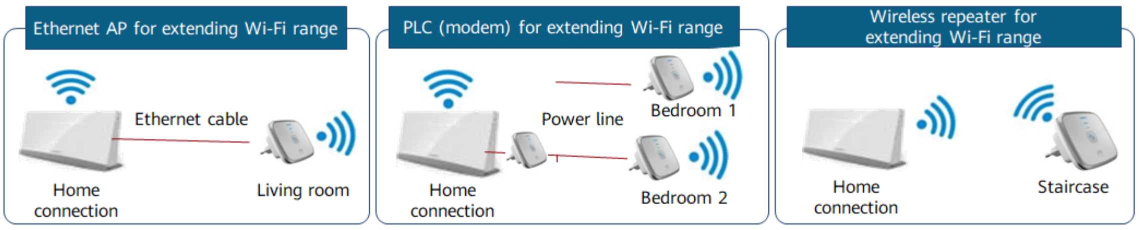

Ethernet AP for extending Wi-Fi range
Ethernet cable
Home connection
Living room
PLC (modem) for extending Wi-Fi range
Power line
Bedroom 1
Home connection
Bedroom 2
Wireless repeater for extending Wi-Fi range
Home connection
Staircase

**Figure 8-3 Three Wi-Fi extension modes**

In many existing home networks, many rooms require a router to be installed inside each room to achieve strong Wi-Fi coverage. Even then, the coverage distance is limited and unlikely to be picked up by a room on a different floor. To overcome this problem, Huawei proposes these 3 solutions, as shown in Figure 8-3.

The first and simplest solution is to install the primary router in a large room, such as the living room, to ensure strong signal coverage. Another solution is to install PLC modems in the bedrooms to extend Wi-Fi coverage through the power lines, while a third solution is to install a wireless repeater on the staircase.

The smart ONT also offers intelligent interconnection. Future homes are shifting more towards smart automation in an effort to cut energy consumption and bring greater convenience to our lives. A smart ONT is needed to connect all of these appliances and devices, regardless of their transmission standards, and provide a platform to control them in one place. On top of this, we also need a cloud platform that will allow us to control the devices through a smartphone, even when we outside the home.

And last but not least is intelligent O&M. Through the cloud platform, we can control, run, and maintain home devices at a click.

The smart ONT (the core) provides two open capabilities - open home and open elastic pipe - that comprise the advanced structure of this technology, helping make smart homes a reality. Open home capabilities include smart Wi-Fi coverage and ZigBee & Z-Wave, enabling ubiquitous connectivity, while open pipe capabilities that extend flexible pipes from operators to homes, integrating applications and building a platform.

## 8.2 Huawei Smart Home Solution

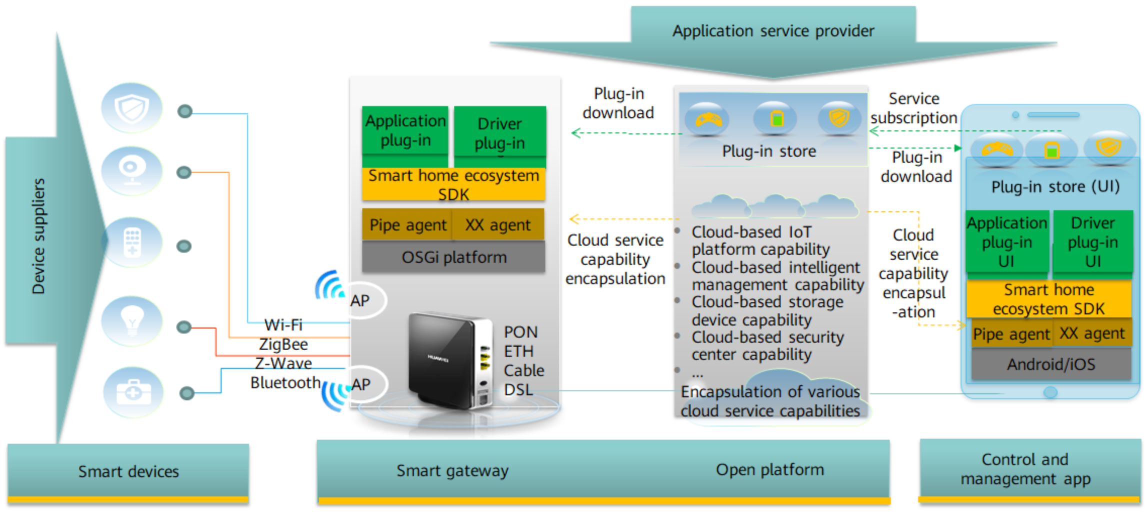


**Figure 8-4 Smart home solution (see the Appendix for a reference image)**

This section describes Huawei's smart home solution, which can be summarized as openness. This solution provides a collaborative platform for all partners across various industries to build a new industry environment.

In terms of hardware, Huawei cooperates with smart device vendors and chip module vendors to research home devices, and ensures home devices are connected to smart gateways. Regarding the IoT platform, Huawei opens up capabilities and works with cloud service providers to build an intelligent cloud platform. As for service applications, Huawei works with app developers and service providers to build a joint app framework for smart homes. At the upper layer, Huawei offers global operators a smart home solution that delivers network assurance for cloud, pipes, and devices.

Figure 8-4 illustrates the Huawei smart home solution architecture. Using the open mode of the smart gateway is similar to using a PC. A PC usually has standard physical PCI slots. To begin with, install an operating system and hardware to the PC, and install drive to the PC to control the hardware. Then install applications if required.

The operating system of the PC acts as the brain, interacting with different software, hardware, and applications. Huawei uses this same structure to develop the smart home system, where physical PCI slots are replaced with wireless protocols such as ZigBee and Wi-Fi. The smart gateway acts as the brain, interacting with different software, hardware, and applications.

A smart gateway has three types of plug-ins (drivers, agents, and applications).

- A driver is a program that must be installed to enable communication between installed hardware and the operating system. Huawei uses OSGi (open-source specifications defined by the ITU) instead of drivers, and this functions as a Java virtual machine. By simply writing a driver specifically for a device in Java to the OSGi, a third party can operate their own device, achieving all-round device integration.

- Agents are another type of plug-in that drive the cloud. An agent offers an environment for operators and third parties to share capabilities (for example, facial recognition) that can be used by partners to develop applications.
- The final type of plug-ins is applications.

Operators can set up a store for gateway apps, providing a one-stop place for consumers to download the required drivers, agents, and apps to get services operational. With the necessary drivers and devices, agents can invoke cloud-based capabilities, while apps implement home services.

In short, Huawei OpenLife solution imports and integrates hardware vendors and service/application content for operators.

## 8.3 HiLink Platform

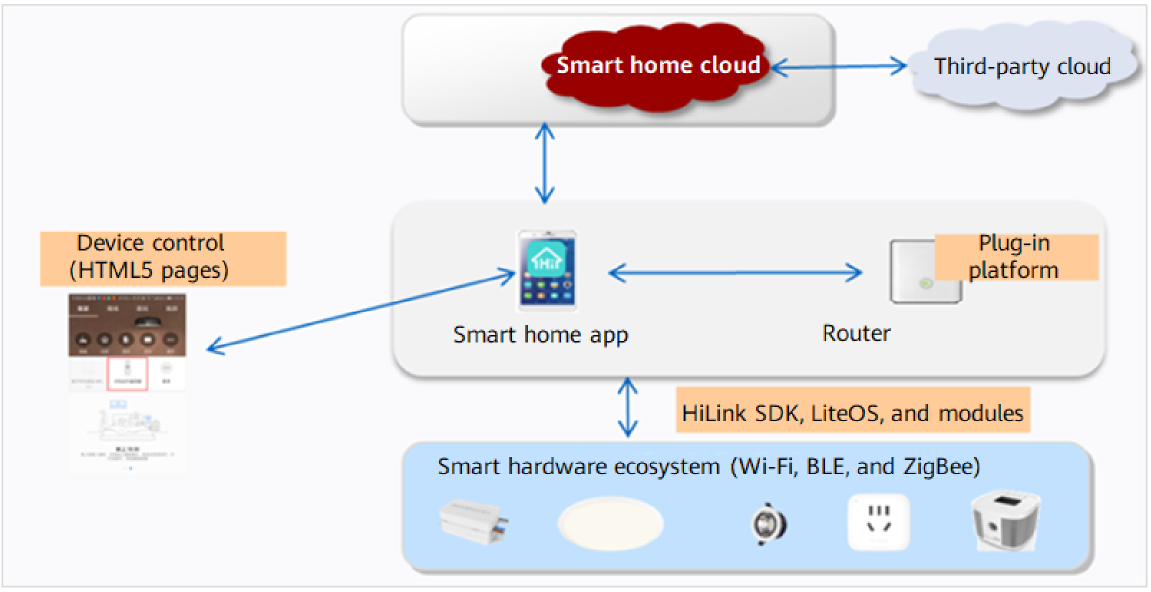

Smart home cloud
Third-party cloud
Device control (HTML5 pages)
Smart home app
Router
Plug-in platform
HiLink SDK, LiteOS, and modules
Smart hardware ecosystem (Wi-Fi, BLE, and ZigBee)

**Figure 8-5 Huawei HiLink technical solution**

Huawei HiLink platform provides developers with a complete set of development solutions, from protocols used by smart hardware, device control pages on the smart home app, all-round device control on the platform, and hardware data sharing on the cloud. The HiLink platform supports the development of these solutions in just one place.

In a nutshell, the platform is open, shared, easy-to-use, secure, reliable, and affordable. The goal of the Huawei smart home solution is to build an open platform that nurtures collaboration among all developers, and to facilitate the rapid development of smart hardware. The ultimate goal is to get home services on the market as fast as possible, and quickly build an open, smart home ecosystem.

Developers use HiLink Access to connect such devices due to its cross-compatibility with partners' products. For example, in the home appliance field, China is home to many major companies, such as Gree and Midea. The smart home products of these manufacturers have their own platforms, but for consumers who buy devices belonging to different brands, it is difficult to use one app or platform to control all of these devices. Huawei HiLink Access offers a solution to this issue.

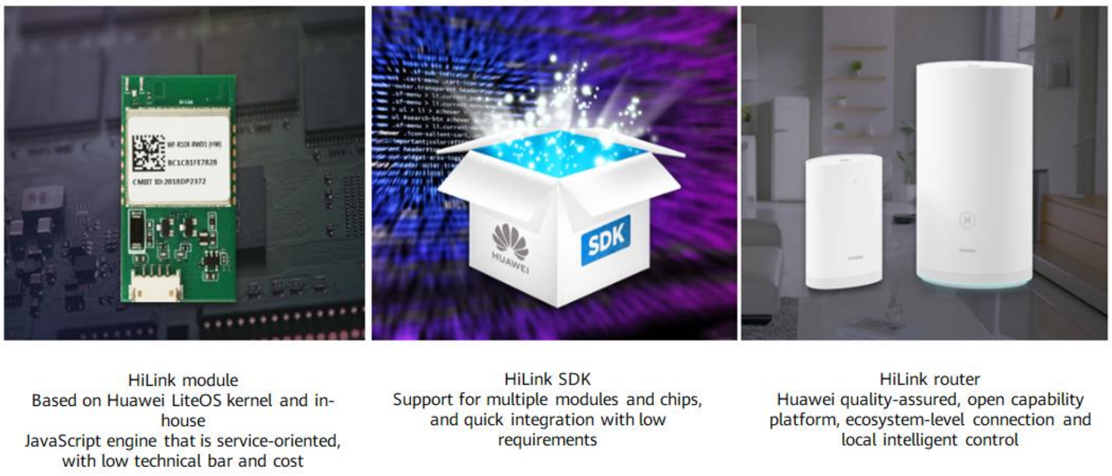

HiLink module
Based on Huawei LiteOS kernel and in-house
JavaScript engine that is service-oriented, with low technical bar and cost
HiLink SDK
Support for multiple modules and chips, and quick integration with low requirements
HiLink router
Huawei quality-assured, open capability platform, ecosystem-level connection and local intelligent control

**Figure 8-6 Huawei HiLink Access (ecosystem connection)**

As shown in Figure 8-6, three different connection methods are available for the HiLink ecosystem. For electrical appliances that lack communication capabilities, HiLink modules can be embedded into the appliances to connect to the platform. Similarly, vendors who have their own platforms may have their own modules. So, if they want to connect to the Huawei ecosystem, they can burn HiLink SDK to their devices to connect to the platform. If a consumer purchases products from different companies but wants to connect to the Huawei ecosystem, they can use HiLink routers to control smart devices and connect them to the HiLink platform.

The Huawei HiLink platform provides two access modes. One is to connect through hardware interconnection, so that devices can connect to the Huawei platform through the HiLink SDK or Huawei LiteOS-based hardware modules. The other option is to connect through cloud interconnection. We can log in to the HiLink cloud using a personal account, and connect devices to the HiLink cloud, delivering cloud capabilities to Huawei compatible devices.

To connect a device to the Huawei HiLink platform, the device must have the Huawei HiLink Device SDK integrated. The Huawei HiLink Device SDK applies to devices that run a supported operating system as well as devices that do not have microprocessors but are directly developed in modules.

The HiLink platform is an open platform, meaning that Huawei provides developers and consumers with multiple access modes. Developers can leverage these modes for application and service development, and consumers can access the platform through multiple methods. This platform offers consumers an environment to connect and control products belonging to different brands on just one platform.

# 9 IoT Platform

## 9.1 Overview of Ecosystem and Platform

There are many interpretations of the concepts of "platform" and "ecosystem" in the market. Nevertheless, the industry has reached consensus that an ecosystem is based on an open platform. The following describes the platform model and ecosystem from the perspective of industry value to illustrate the importance of a platform-based ecosystem.

### 9.1.1 Platform Model

The platform model refers to an open, collaborative system built by an enterprise with the enterprise itself as the core. An enterprise that builds a platform is the platform owner and is responsible for the overall platform support and operations. Internal and external roles of the enterprise, such as capital, employees, partners, and users, can collaborate with the enterprise in real time after obtaining certain permissions.

Unlike traditional business cooperation, the platform model enables open, real-time collaboration by leveraging web 2.0 or mobile Internet. In the IoT platform model, platform participants can collaborate with each other.

### 9.1.2 Ecosystem

An ecosystem is a type of business collaboration network developed on the platform model. It is spontaneous and self-consistent, and has an internal value chain. In an ecosystem, enterprises use modern information technologies to implement network-like loose coupling. Similar to the platform model, enterprises in the ecosystem can collaborate with each other. An ecosystem can reduce transaction costs and share business opportunities. It also promotes the transfer of the internal value chain between its members and enables them to contact as many potential partners as possible.

Transaction costs are reduced and more business opportunities emerge as the number of internal transactions and data flows in the ecosystem increase. To sum up, the value that an ecosystem brings to every participant is often determined by the total scale of its system.

The platform model is centered on enterprise-dominated operations, and the ecosystem is an overall view of network-like collaboration based on the platform model. A platform-based ecosystem is the product of the platform model as the platform model develops. It is also the overall view of the value chain formed by multiple platforms that serve different objects.

The IoT is used in diverse industries. How to integrate the value chain, attract developers, enrich applications, and expand user groups is the key to realizing the value of the IoT.

## 9.2 Origin of the IoT Platform

The world is witnessing a new technological revolution and industry transformation. Information technologies represented by the IoT, big data, and artificial intelligence (AI) are accelerating their integration with the real economy. They are being transformed into productivity and upgraded into infrastructure and key elements that reshape production organization and growth models. During China's 13th Five-Year Plan period, the IoT will enter a new phase of cross-border integration, integrated innovation, and scale development. A plethora of development opportunities will emerge. Since 2017, the IoT industry around the world has been engaged in a new round of development, driven by new network and platform technologies. More IoT technology innovations will be forthcoming. We need to study the development trends and direction of the IoT from the perspectives of technology and industry.

### 9.2.1 IoT Is Ushering In Industry Innovation and Transformation

From 2005 onwards, the global IoT went through a market cultivation phase (phase 1.0) focused on "concept exploration, government-led, and application demonstration". With more technological breakthroughs and requirements, significant changes have taken place in the development conditions and technical composition of the IoT. By now, the IoT has entered its industrial outbreak phase (phase 2.0), which is known as "cross-border integration, integrated innovation, scale application, and ecological acceleration". There is an obvious trend of systematic and integrated innovation in the convergence of IoT, big data, AI, and other new information technologies. Core technology systems, such as hardware, software, and services, are being reconstructed and iterated rapidly. Edge computing and fog computing support self-sensing, self-decision-making, self-optimization, and self-execution. Blockchain supports multi-party storage and exchange of data. Virtual reality (VR) and augmented reality (AR) provide three-dimensional, intuitive display. These new technologies are integrated with the IoT to provide innovative means of IoT sensing, data processing, and data presentation. They also serve industry-specific applications such as industrial autonomous control and maintenance, intelligent transportation, and smart building. In this new development phase, the IoT features "Internet of everything, ubiquitous intelligence, platform ecosystems, and data operation". A new turning point is arriving.

Currently, 60% of enterprises are using or about to deploy IoT services, 24% of IT budgets are being used for IoT investment, and 63% of enterprises have gained significant return on IoT investment. See Figure 9-1.

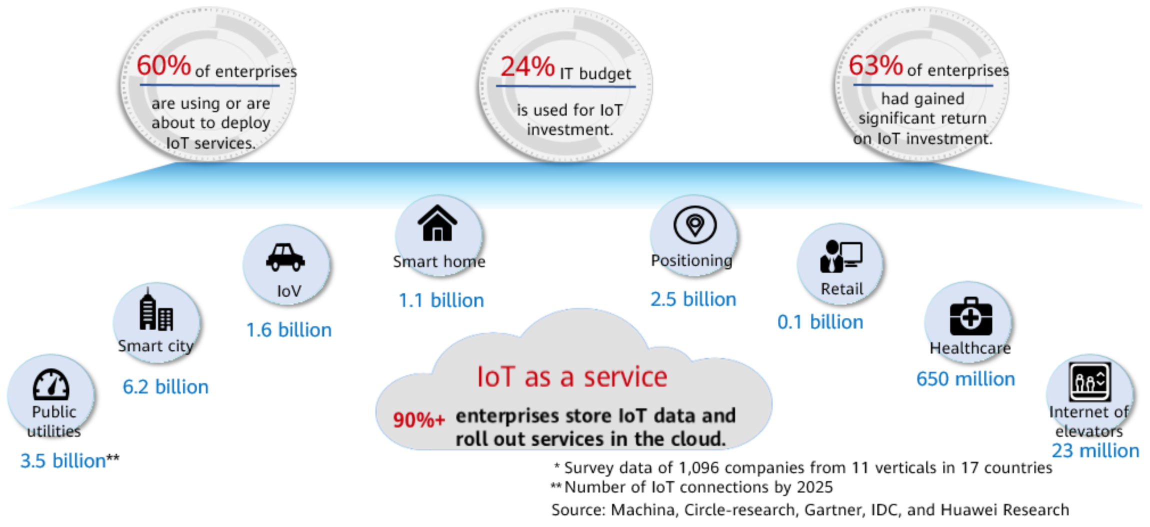

60% of enterprises are using or are about to deploy IoT services.
24% IT budget is used for IoT investment.
63% of enterprises had gained significant return on IoT investment.
Public utilities 3.5 billion**
Smart city 6.2 billion
IoV 1.6 billion
Smart home 1.1 billion
Positioning 2.5 billion
Retail 0.1 billion
Healthcare 650 million
Internet of elevators 23 million
IoT as a service
90%+ enterprises store IoT data and roll out services in the cloud.
* Survey data of 1,096 companies from 11 verticals in 17 countries
** Number of IoT connections by 2025
Source: Machina, Circle-research, Gartner, IDC, and Huawei Research

**Figure 9-1 IoT is ushering in industry innovation and transformation (see the Appendix for a reference image)**

## 9.2.2 Major Challenges Facing the IoT Industry

The IoT has witnessed scale development in recent years, and countries around the world have invested heavily in IoT research and development. However, there are still many problems to solve in terms of technology, management, costs, policies, and security.

Traditional Internet standards are not suitable for the IoT. Data at the IoT sensing layer has multiple sources and is heterogeneous. Different devices have different interfaces and technical standards. The network layer and application layer use different protocols and architectures. Establishing a unified IoT architecture and unified technical standards is a challenge facing the IoT. Figure 9-2 shows major challenges in the IoT industry.

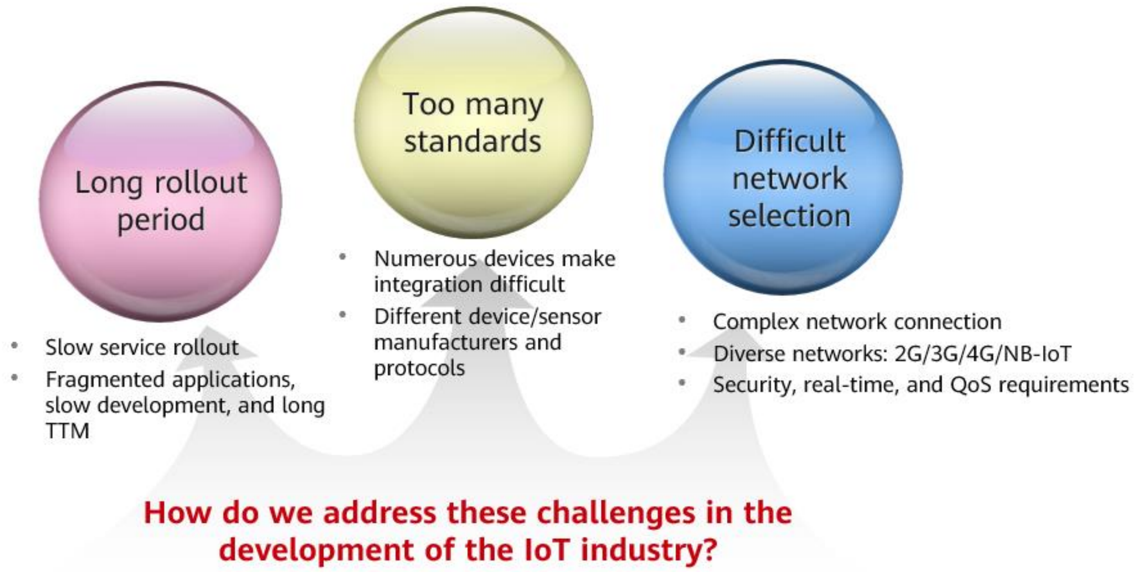

Long rollout period
Slow service rollout
Fragmented applications, slow development, and long TTM
Too many standards
Numerous devices make integration difficult
Different device/sensor manufacturers and protocols
Difficult network selection
Complex network connection
Diverse networks: 2G/3G/4G/NB-IoT
Security, real-time, and QoS requirements
How do we address these challenges in the development of the IoT industry?

**Figure 9-2 Major challenges facing the IoT industry**

## 9.2.3 Requirements for the IoT Platform

The IoT itself is a complex network system, and its wide application in different industries makes intersections inevitable. If such a network system does not have a dedicated, comprehensive platform to manage information by category, a large amount of redundant data and repeated work will occur. Applications in each industry are independent of each other, resulting in high costs and low efficiency. All these hinder the development of the IoT. The IoT is in urgent need of a unified management platform that can integrate resources of different industries to form a complete industry chain.

Countries around the world are proactively supporting the IoT, but there are few IoT projects that can secure investment and be used on a large scale. For example, RFID tags and readers are far more expensive than enterprises expected; sensing networks are multi-hop self-organizing networks that are vulnerable to environmental or human threats. Costly network maintenance is required to ensure stable network quality and reliable, secure, and real-time data transmission. Unacceptable high costs would make IoT development unfeasible.

Today, the Internet still has security vulnerabilities. As an emerging product, the IoT has even more prominent security issues, due to its complex architecture and lack of unified standards. Sensing networks are vulnerable to their environments. Sensors are placed in natural environments, including harsh ones. Sensing networks must support self-healing to ensure long-term network integrity. Sensing networks are also vulnerable to human impact.

RFID is another key technology. RFID enables real-time monitoring by attaching a tag to an item. This exposes private data. In addition, collaboration between enterprises and between countries is common nowadays. There could therefore be severe consequences if a network is attacked. It is crucial to strike a balance between informatization and security during the use of the IoT.

To develop IoT services, we require a secure, reliable platform that supports device access decoupling and capability openness. Figure 9-3 shows the requirements for the IoT platform.

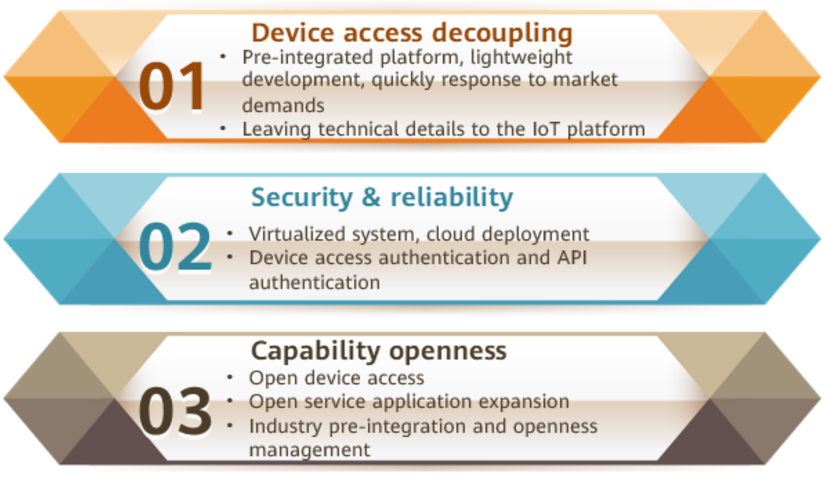

01
Device access decoupling
• Pre-integrated platform, lightweight development, quickly response to market demands
• Leaving technical details to the IoT platform
02
Security & reliability
• Virtualized system, cloud deployment
• Device access authentication and API authentication
03
Capability openness
• Open device access
• Open service application expansion
• Industry pre-integration and openness management

**Figure 9-3 Requirements for the IoT platform**

## 9.3 Introduction to the HUAWEI CLOUD IoT Platform

### 9.3.1 IoT Platform Overview

The rapid development of the IoT requires a secure, reliable platform that supports device access decoupling and capability openness. The HUAWEI CLOUD IoT platform is such a platform. It is access-agnostic, secure, reliable, open, and scalable. It helps enterprises and industry users quickly integrate applications and build end-to-end IoT solutions.

The IoT platform architecture consists of four layers: device layer, access layer, platform layer, and application layer. The platform layer can be further divided into the device connectivity layer and service enablement layer. Figure 9-4 shows the functional architecture of the HUAWEI CLOUD IoT platform.

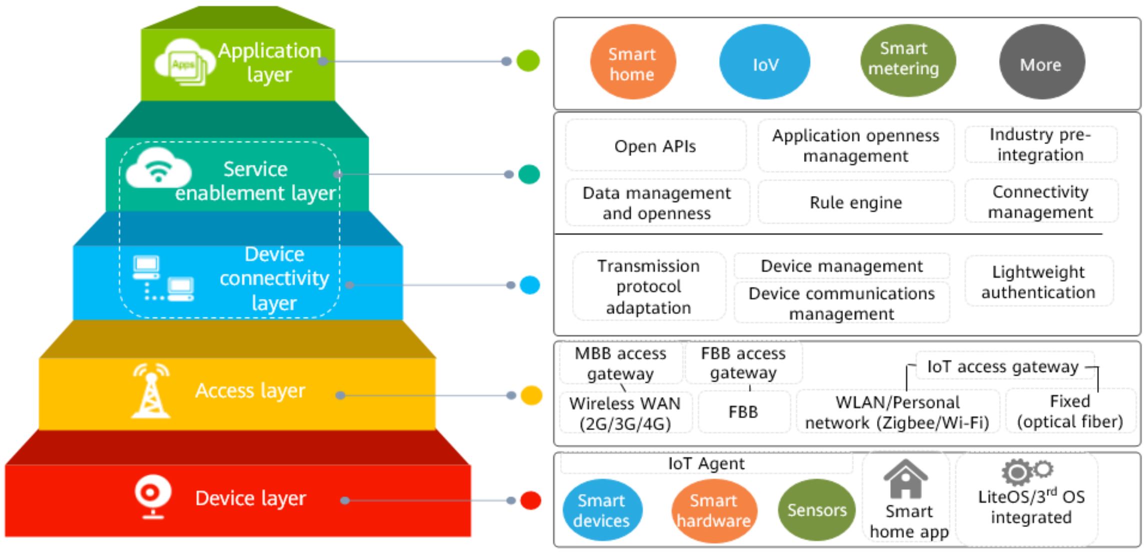


**Figure 9-4 Functional architecture of the HUAWEI CLOUD IoT platform**

The device layer provides IoT Agents that are adaptable to OSs and smart devices provided by different manufacturers. It is pre-integrated with valuable industry-specific applications. Table 9-1 describes these devices.

**Table 9-1 Devices at the device layer**

| Peripheral/Device | Description |
|---|---|
| Smart device | Smart devices, such as smartphones, tablets, and computers, are used to remotely control intelligent hardware. |
| Intelligent hardware | Intelligent hardware is connected to the Internet using various types of sensors, such as RFID devices, photoacoustic or electromagnetic sensors, and laser scanners, to perform data collection, convergence, and processing. |
| Sensor | A sensor is a detection apparatus that responds to a stimulus and |

| Peripheral/Device | Description |
|---|---|
| | generates a signal that can be measured or interpreted, to meet requirements for data transmission, processing, storage, recording, and control. |
| Industry app | Industry apps can be installed on devices, such as mobile phones, computers, and tablets, to control intelligent hardware. |
| IoT Agent | IoT Agents use various communications protocols to provide network access for devices. The Agents isolate applications from OSs and hardware.<br>● The Agents provide the bottom layer with SDKs, which can be adapted to different OSs and hardware.<br>● The Agents provide a variety of APIs independent of underlying resources. |

The access layer supports both wireless and wired access. A large number of devices can be connected. Table 9-2 describes the access layer.

**Table 9-2 Access scenarios supported by the access layer**

| Access Scenario | Description |
|---|---|
| MBB (P-GW/GGSN) | Wireless wide area network (WWAN) |
| FBB (BRAS) | FBB and fiber to the x (FTTx) |
| IoT (enterprise AR) | Wireless local area network (WLAN)<br>Wireless personal area network (WPAN)<br>Fixed, including FTTx, hybrid fiber coaxial (HFC), and electrical power cable |

The device connectivity layer provides unified device access, asset and device management, and SIM card connectivity management. Table 9-3 describes the device connectivity layer.

**Table 9-3 Functions provided by the device connectivity layer**

| Function | Description |
|---|---|
| Device management | Carriers can manage all sensor nodes at the bottom layer, obtain information about each node, and implement remote control. |
| Device communications management | Device communications management provides the following functions:<br>● Device login |

| Function | Description |
|---|---|
| | ● Data communications<br>● Device-initiated subscription<br>● User-initiated subscription |
| Transmission protocol adaptation | The IoT platform supports HTTP, MQTT, and CoAP. |
| Lightweight authentication | The IoT platform simplifies the authentication process for devices and applications. |

The service enablement layer provides functions such as open API gateways, data management, and a rule engine. Table 9-4 describes the service enabler layer.

**Table 9-4 Functions provided by the service enablement layer**

| Function | Function |
|---|---|
| Open applications and APIs | Functioning as an independent module, the open API gateway provides API searching, help, and lifecycle management. |
| Data management and openness | Native device data is converted according to industry-defined data models that can be identified by the IoT platform. Based on the data, carriers can use rule engine and service orchestration modules to define or improve their business practices. |
| Rule engine | The rule engine enables end users to use predefined rules to customize rules on a user-friendly UI. |

The IoT platform supports multiple types of open APIs and pre-integration of multiple industry-specific applications, including smart home, IoV, smart metering, and third-party applications. Table 9-5 describes the application scenarios.

**Table 9-5 Applications pre-integrated at the application layer**

| Application | Function |
|---|---|
| Smart home | Home appliances and devices are remotely monitored and controlled for more secure, convenient living. |
| IoV | Traffic information, remote vehicle diagnosis, and E-Call/B-Call services are provided. |
| Smart metering | NB-IoT communications modules are embedded in water and electric meters. These meters can be upgraded based on existing base stations to provide smart metering, which features low power consumption, wide coverage, and low-cost deployment. |

| Application | Function |
|---|---|
| Third-party application | Third-party applications are developed by organizations or individuals rather than software editors. Typical third-party applications include smart grid, smart building, and smart business. |

The IoT platform connects and manages a large number of devices. It works with other HUAWEI CLOUD services to quickly build IoT applications. A complete IoT solution consists of the IoT platform, devices, and applications. Figure 9-5 shows the architecture of the HUAWEI CLOUD IoT platform.

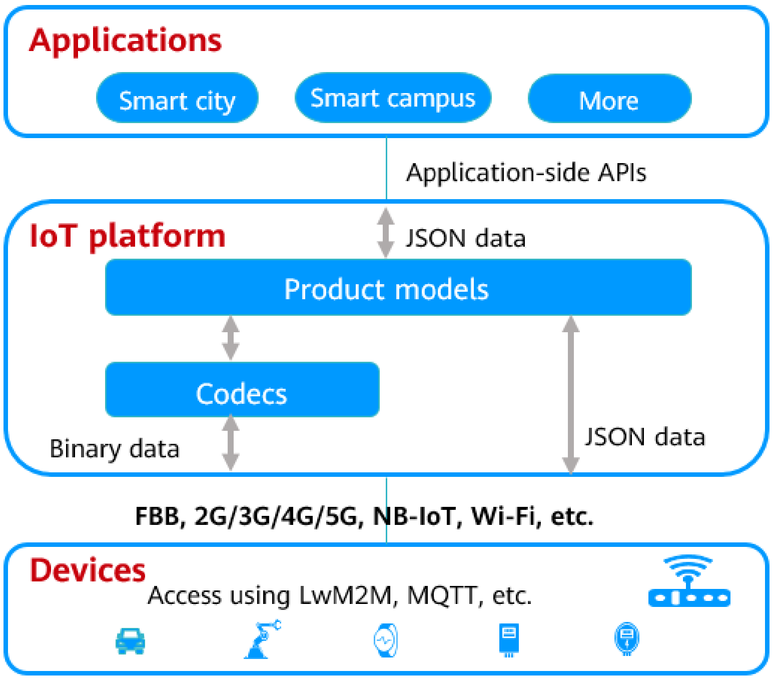


**Figure 9-5 HUAWEI CLOUD IoT platform**

The IoT platform is located between applications and devices. It hides differences between device interfaces to enable quick device access. It provides robust capabilities to help developers quickly construct diverse IoT applications.

Devices can access the platform via FBB, 2G/3G/4G/5G, NB-IoT, and Wi-Fi networks. They can report service data to the platform using MQTT or LwM2M over CoAP. Devices can also receive commands from the platform.

Applications call platform APIs for device management, data reporting, command delivery, and other service scenarios.

Table 9-6 describes the services provided by HUAWEI CLOUD IoT.

**Table 9-6 HUAWEI CLOUD IoT services**

| Service | Category | Description |
|---|---|---|
| IoT Device Access | Device connectivity | Devices can connect to the IoT platform using multiple protocols in multiple access modes. |
| | Device management | The IoT platform can manage devices and device data. |
| | Data forwarding | The IoT platform can forward data to other HUAWEI CLOUD services. |
| | Application integration | Applications can access the IoT platform and call APIs provided by the platform. |
| IoT Studio | | Developers can construct IoT applications without coding. |
| IoT Analytics | | The IoT platform can analyze device data. |
| Other related services | IoT Edge | The IoT platform can work with other IoT services to build a solution. |
| | Global SIM Link | |

## 9.3.2 IoT Device Access

### 9.3.2.1 Overview

IoT Device Access (IoTDA) is a basic service of the HUAWEI CLOUD IoT platform. IoTDA provides functions such as device fleet access, bidirectional communications between devices and the cloud, batch device management, remote control and monitoring, Over-the-Air (OTA) upgrades, and device linkage rules. It can flexibly transfer device data to other HUAWEI CLOUD services. Using IoTDA, you can quickly connect devices to the platform and integrate your applications. As shown in Figure 9-6, the IoT system based on the HUAWEI CLOUD IoT platform consists of the following parts: devices, device access, message communications, device management, data forwarding, IoT applications, and interworking with other HUAWEI CLOUD services.

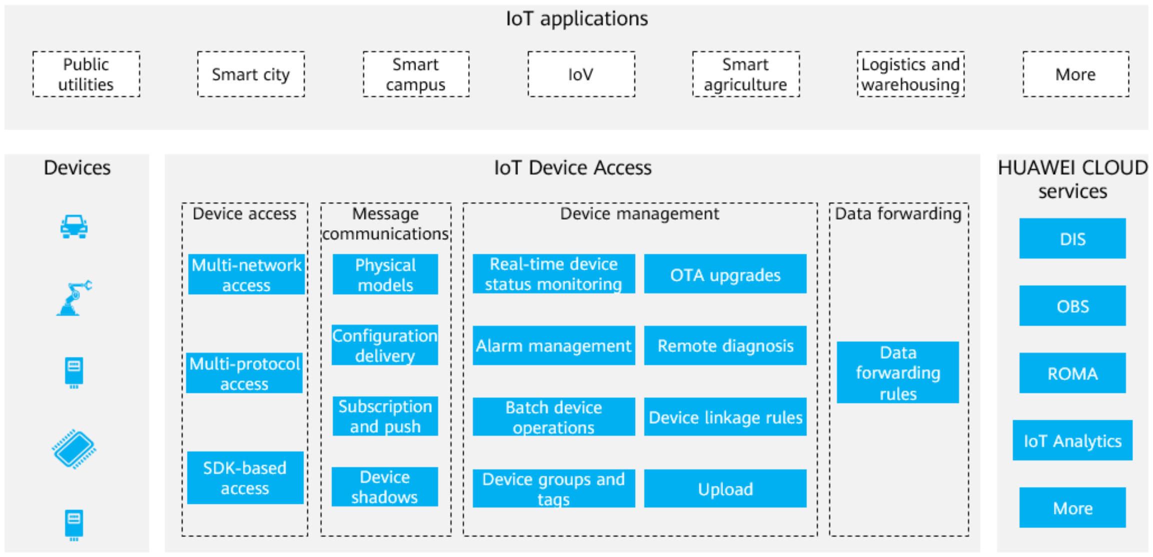


**Figure 9-6 IoT system based on the HUAWEI CLOUD IoT platform (see the Appendix for a reference image)**

### 9.3.2.2 Typical IoT Protocols

**HTTP**

The HyperText Transfer Protocol (HTTP) is an application-layer protocol used for communications between web servers and browsers. It makes the browser more efficient and reduces the amount of data to transmit. It ensures that a computer correctly and quickly transmits hypertext documents and determines the priority of the document contents to transmit and display, for example, to display text prior to images.

HTTP runs at the application layer. It is a request-response protocol in the standard client-server model.

HTTP is stateless. There is no mapping between two consecutive requests. An HTTP server does not know whether two requests came from the same client. To solve this problem, the cookie mechanism is introduced.

HTTP is a synchronization protocol, which means that the client needs to wait for a response from the server. Web browsers can meet this requirement, at the expense of scalability. In the IoT field, synchronous communication is difficult to implement because of device fleets, unreliable networks, and high latency. An asynchronous protocol is more suitable for IoT applications. Sensors send data, where the network determines the best route and time to deliver the data to the target device and service.

HTTP is a heavyweight protocol with many headers and rules and is not suitable for restricted networks.

Figure 9-7 shows the three-way handshake process.

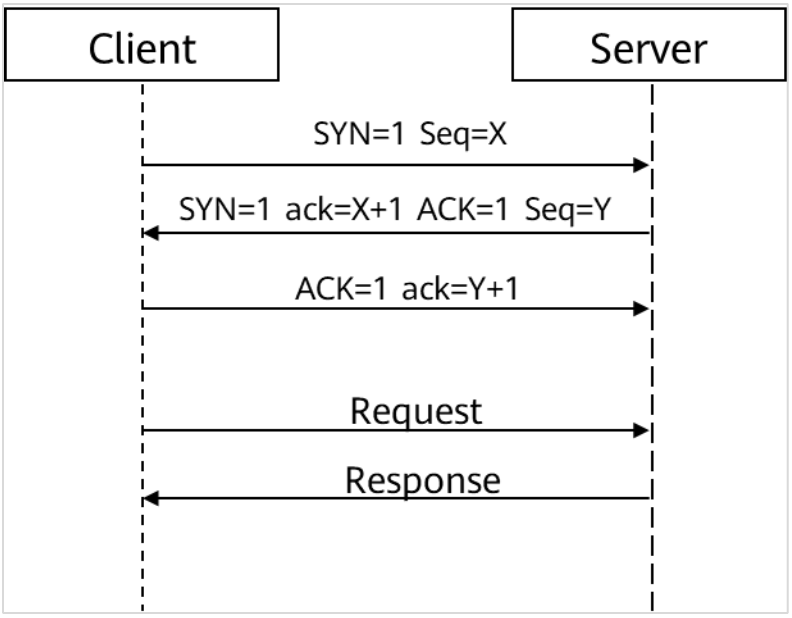


**Figure 9-7 HTTP three-way handshake process**

Note: SYN stands for synchronization, Seq stands for sequence, ACK stands for acknowledgment, and ack stands for acknowledgment sequence.

First handshake:

A client sends a connection request packet to set up a connection with the server. As stipulated in TCP, a packet segment with SYN=1 cannot carry data and needs to consume a sequence number. The client generates a random initial sequence number (X), and includes both SNY=1 and Seq=X in the packet. The client enters the SYN-SENT state.

Second handshake:

The server sends an ACK packet to accept the connection request. The server generates a random initial sequence number (Y), and includes SYN=1, ACK=1, ack=X+1, and Seq=Y in the packet. The server enters the SYN_RCVD state.

Third handshake:

The client sends an ACK packet to the server to acknowledge the receipt of the ACK packet from the server. In this ACK packet, ACK is 1 and ack is Y+1. The TCP connection is established, and the client enters the ESTABLISHED state.

Necessity of the three-way handshake:

After the client sends the first connection request packet, the server may fail to receive the packet on time due to network congestion. When it finally receives the packet, which is invalid, the server still sends an ACK packet to the client and waits for subsequent requests from the client. If a two-way handshake is used, the client ignores the ACK packet sent by the server, but the server continues to wait for requests from the client. This wastes resources. When a three-way handshake is used, the server disconnects from the client when it cannot receive the ACK packet from the client.

**MQTT**

Message Queuing Telemetry Transport (MQTT) is an instant messaging protocol developed by IBM. MQTT uses the publish-subscribe pattern. The client subscribes to desired information from the server, and the server pushes the information to the client.

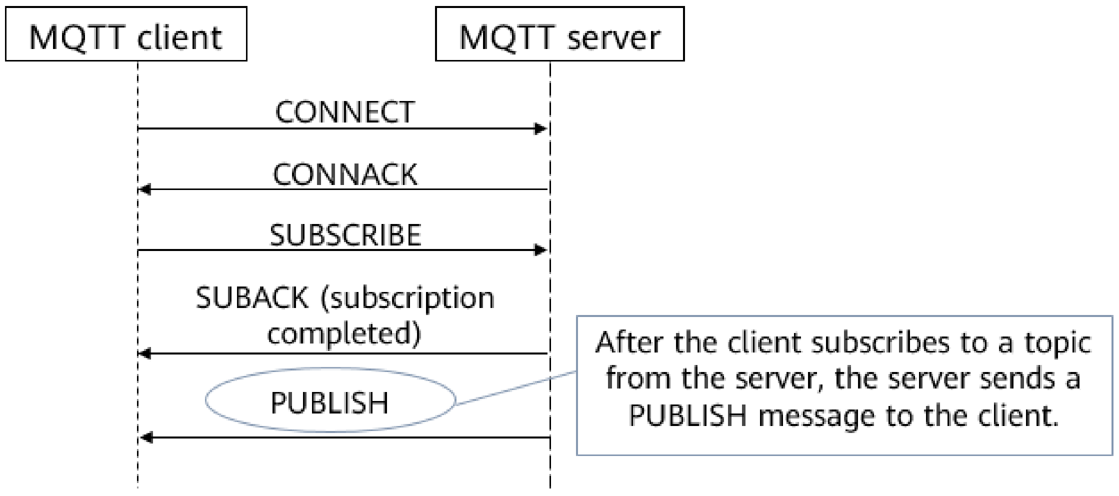

MQTT client
MQTT server
CONNECT
CONNACK
SUBSCRIBE
SUBACK (subscription completed)
PUBLISH
After the client subscribes to a topic from the server, the server sends a PUBLISH message to the client.

**Figure 9-8 MQTT interaction process**

MQTT has the following characteristics:

- MQTT is simple and lightweight. (A message can contain only two bytes. MQTT has low requirements for hardware configuration and is suitable for devices with limited hardware resources such as CPU. This protocol helps reduce device costs and promote industry development.)
- MQTT supports reliable message transmission based on TCP/IP.
- MQTT uses persistent connections and a keep-alive mechanism to reduce the cost of link re-establishment.
- MQTT supports real-time message notifications and rich push content.
- The keep-alive mechanism prevents a device from entering sleep mode, which consumes much power.
- MQTT meets requirements of wireless sensor networks and IoT. It is widely used in the smart home solution.

MQTT client

- Both publishers and subscribers are MQTT clients responsible for publishing or subscribing to messages, respectively.
- An MQTT client can be any device (from a micro controller up to a full-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

MQTT broker

- The MQTT broker is the core of any publish-subscribe protocol and can concurrently process millions of connected MQTT clients.
- The MQTT broker is responsible for receiving all messages and sending the messages to all subscribed clients.

- One responsibility of the broker is to keep the session data of all clients that have persistent sessions, including subscribed and missed messages.
- Another responsibility is client authentication and authorization.

**CoAP**

Constrained Application Protocol (CoAP) is designed for resource-limited devices (such as sensor nodes) and networks (such as NB-IoT and LoRa). CoAP is based on HTTP. CoAP uses a request/response model, in which the client initiates a request and the server responds to the request. CoAP optimizes the packet length and offers reliable communications to address the issues that may occur on HTTP in restricted conditions.

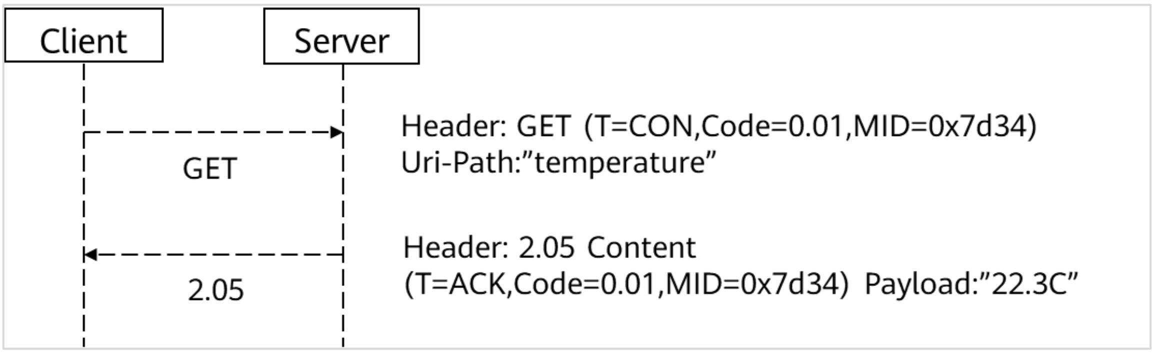

Client
Server
GET
Header: GET (T=CON,Code=0.01,MID=0x7d34)
Uri-Path:"temperature"
2.05
Header: 2.05 Content
(T=ACK,Code=0.01,MID=0x7d34) Payload:"22.3C"

**Figure 9-9 CoAP interaction process**

CoAP has the following characteristics:

- The header is compressed, the packet format is simple, and a message can be short as 4 bytes.
- CoAP uses UDP at the transport layer to reduce network overhead and support multicast.
- To make up for the unreliability of UDP transmission, CoAP provides message retransmission.
- Persistent connections are not supported, and no heartbeat messages are sent. When no service is running, a CoAP device does not need to send messages to external systems.
- The device may need to be woken up before processing services, resulting in poor real-time performance.

Unlike MQTT, CoAP does not need to maintain persistent connections or send heartbeat messages continuously. It is more suitable for devices that use the sleep/wakeup mechanism in the IoT scenario. The device can stay in the sleep mode for a long time, saving power. For example, one battery can be used for 10 years or longer. CoAP is mainly used in solutions such as smart water meters, smart electricity meters, smart agriculture, and smart parking.

## 9.3.3 Device Management

### 9.3.3.1 Device Registration and Access Authentication

Device registration and access authentication are required to ensure that devices connected to the IoT platform are secure and reliable.

Device registration: Users register device details on the console or call the Registering Devices API to register device information. These devices can then connect to the IoT platform for communications.

Device access authentication: The IoT platform authenticates a device when it attempts to connect to the platform. The platform authenticates integrity and security of device data, device access data, and messages exchanged between the device and platform.

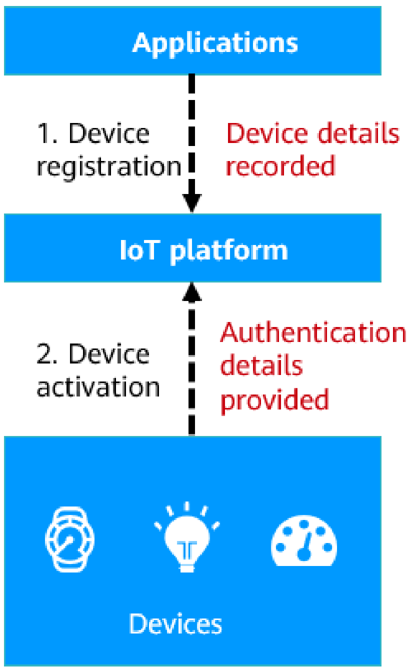


**Figure 9-10 Device registration and access authentication**

### 9.3.3.2 Command Delivery

The product model of a device defines commands that can be delivered by the IoT platform to the device. The platform modifies the service properties of the device when delivering commands.

The IoT platform has two ways to deliver commands, as described in Table 9-7.

**Table 9-7 Command delivery**

| Mechanism | Definition | Application Scenario | NB-IoT Devices | Devices Integrated with the AgentLite SDK or Native MQTT Devices |
|---|---|---|---|---|
| Immediate delivery | The IoT platform delivers commands to a device regardless of whether the device is online. If the device is offline or the device does not receive the command, the delivery fails. Commands can be delivered to devices bound to the current applications and other authorized applications. | Commands must be executed in real time, for example, turning on a street lamp or turning off a gas meter switch. Applications should determine the appropriate time to deliver a command. | Applicable. The working mode of the device must be set to **DRX** or **eDRX**, and an Internet Protocol Security (IPsec) tunnel must be established between the IoT platform and the EPC network. | Applicable |
| Delayed delivery | The IoT platform caches a command and delivers it to a device when the device goes online or reports data. If a device has multiple cached commands, the IoT platform delivers the commands in sequence. Commands can be delivered to devices bound to the current application and other authorized applications. | Delayed delivery applies to commands that do not need to be executed immediately, for example, configuring water meter parameters. | Applicable. The working mode of the device must be set to **PSM**. | Not applicable |

In immediate delivery mode, all commands are directly delivered to devices. In delayed delivery mode, commands are delivered to devices in serial mode after the devices go online or report data to the IoT platform. Cached commands are delivered to devices one by one based on the time they are cached. Figure 9-11 shows the status transition of a device command over the entire lifecycle.

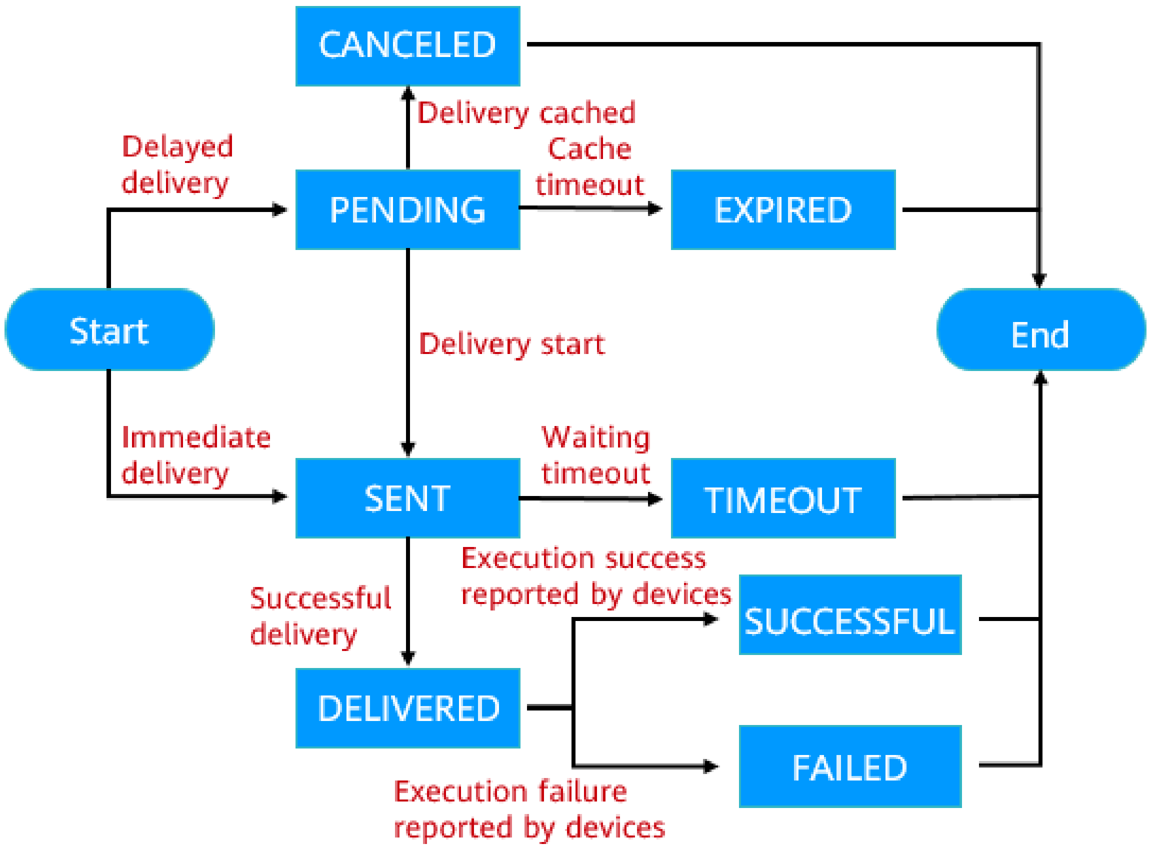


**Figure 9-11 Device command status transition**

Table 9-8 describes the command status.

**Table 9-8 Device command status description**

| Status | Description |
|---|---|
| PENDING | For an NB-IoT device in delayed delivery mode, the platform caches a command if the device has not reported data. The command status is **PENDING**.<br>This status does not exist for an NB-IoT device in immediate delivery mode.<br>This status does not exist for an MQTT device. |
| EXPIRED | For an NB-IoT device in delayed delivery mode, if the platform does not deliver a command to the device within a specific time, the command status is **EXPIRED**. The expiration time is subject to the value of expireTime carried in the command request. If expireTime is not carried, |

| Status | Description |
|---|---|
| | the default value (48 hours) is used.<br>This status does not exist for an NB-IoT device in immediate delivery mode.<br>This status does not exist for an MQTT device. |
| CANCELED | If you cancel a pending command, the command status is **CANCELED**. |
| SENT | For an NB-IoT device in delayed delivery mode, the platform sends a cached command when receiving data reported by the device. In this case, the command status changes from **PENDING** to **SENT**.<br>For an NB-IoT device in immediate delivery mode, if the device is online when the platform delivers a command, the command status is **SENT**.<br>If an MQTT device is online when the IoT platform delivers a command, the task status is **SENT**. |
| TIMEOUT | If the platform does not receive a response within 180 seconds after delivering a command to an NB-IoT device, the command status is **TIMEOUT**.<br>This status does not exist for an MQTT device. |
| DELIVERED | If the platform receives a response from a device, the command status is **DELIVERED**. |
| SUCCESSFUL | If the platform receives a result indicating that the command execution succeeded, the command status is **SUCCESSFUL**. |
| FAILED | If the platform receives a result indicating that the command execution failed, the command status is **FAILED**.<br>For an NB-IoT device in immediate delivery mode, if the device is offline when the platform delivers a command, the command status is **FAILED**.<br>If an MQTT device is offline when the IoT platform delivers a command, the task status is **FAILED**. |

You can use command delivery to remotely control devices. Command delivery is applicable when you need to perform an operation on a large number of devices or on a device that is located in a remote place.

#### 9.3.3.3 Device Linkage Rules

You can customize rules based on predefined rule scenarios.

Rules can be bound with devices, applications, or alarms. When a specific condition is met, the corresponding action is performed. Using the rule engine, abnormal events can be promptly notified and handled. Rule engine helps end users maintain and monitor devices and ensure fast recovery of system services. Location tracing events or events that occur when a threshold is reached can be defined as triggers for rules and configured with corresponding actions.

Figure 9-12 shows the triggers and actions of device linkage rules.

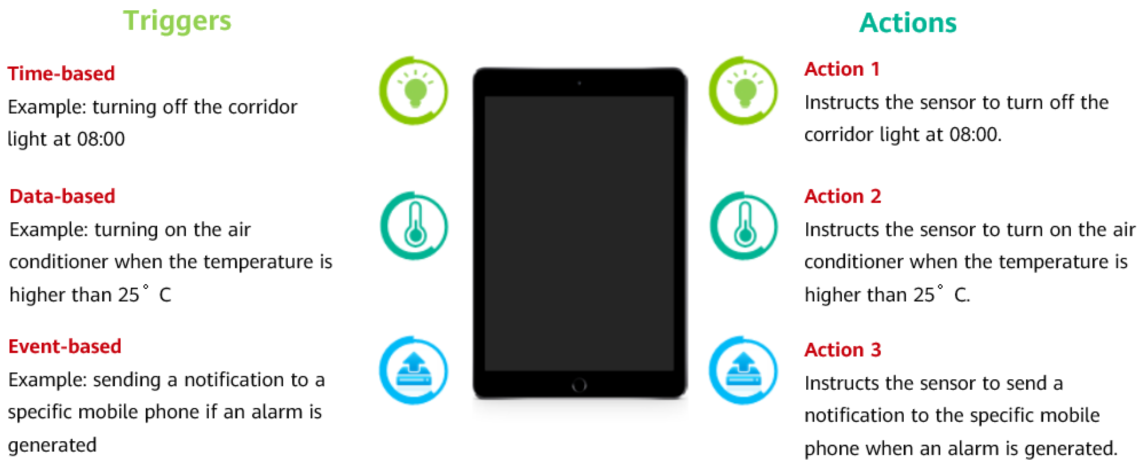

Triggers
Time-based
Example: turning off the corridor light at 08:00
Data-based
Example: turning on the air conditioner when the temperature is higher than 25° C
Event-based
Example: sending a notification to a specific mobile phone if an alarm is generated
Actions
Action 1
Instructs the sensor to turn off the corridor light at 08:00.
Action 2
Instructs the sensor to turn on the air conditioner when the temperature is higher than 25° C.
Action 3
Instructs the sensor to send a notification to the specific mobile phone when an alarm is generated.

**Figure 9-12 Device linkage rules**

### 9.3.3.4 Firmware and Software Upgrades

The IoT platform can upgrade the firmware and software of devices in OTA mode.

- Firmware upgrade

  Firmware upgrade, also called firmware over the air (FOTA), allows users to upgrade the firmware of LwM2M devices and CoAP devices in OTA mode. The firmware upgrade protocol is LwM2M.

- Software upgrade

  Software upgrade, also called software over the air (SOTA), allows users to upgrade the software of LwM2M devices and CoAP devices in OTA mode. The firmware upgrade protocol is PCP.

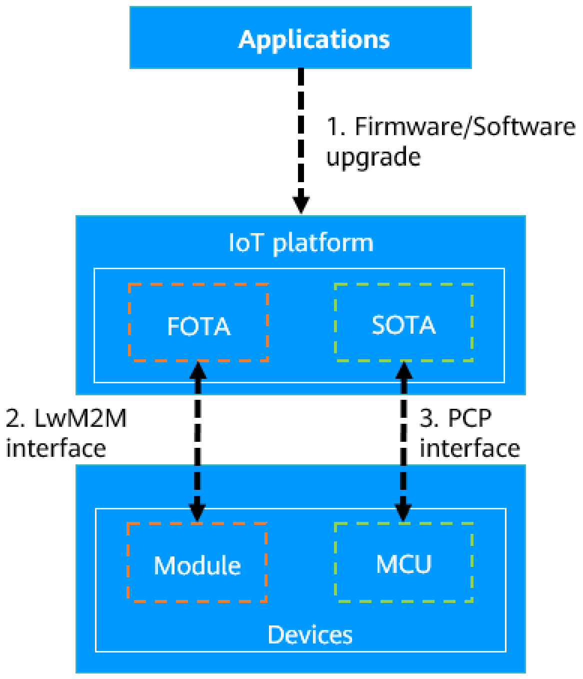


**Figure 9-13 Firmware and software upgrades**

### 9.3.3.5 Batch Operations

The IoT platform supports batch operations on devices. The batch operations include batch device registration, command delivery, location upload, device configuration, and software and firmware upgrades.

Batch device registration: It takes a long time to register a large number of devices one by one. Batch registration is recommended to speed up the process.

Batch command delivery: To deliver commands to devices in batches, create a batch command delivery task through the API and view the execution status, operator, and success rate of the task on the console.

Batch location upload: The locations of a large number of devices can be uploaded in batches. This operation applies to devices that are installed at a fixed location, for example, water meters.

Batch device configuration: A large number of devices can be configured in batches.

Batch software/firmware upgrades: You can create a batch software/firmware upgrade task on the platform to upgrade software/firmware of devices in batches.

Batch operations enable the IoT platform to manage devices in a unified manner. Batch operation applies to the scenario in which a large number of devices want to register with the IoT platform or a user wants to perform the same operation on all devices or devices in a group. A maximum of 30,000 devices can be registered or have their locations uploaded in batch. A maximum of 10,000 devices can be configured and upgraded in batch. A command can be delivered to a maximum of 10,000 devices in batch.

## 9.3.4 Data Forwarding

The IoT platform can work with other HUAWEI CLOUD services to process and forward device data on demand. You do not need to purchase servers to store, calculate, and analyze device data. Figure 9-14 shows the data forwarding process.

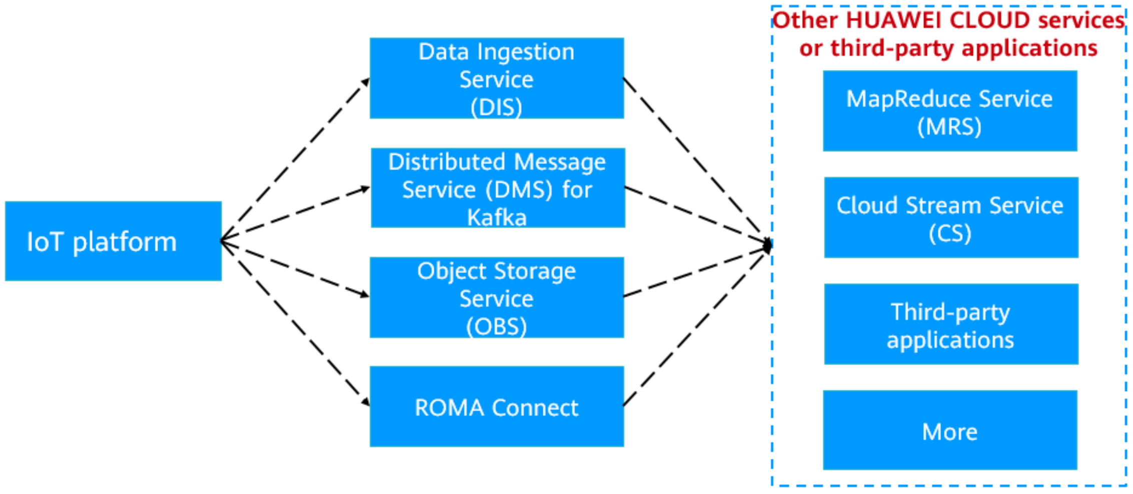

IoT platform
Data Ingestion Service (DIS)
Distributed Message Service (DMS) for Kafka
Object Storage Service (OBS)
ROMA Connect
Other HUAWEI CLOUD services or third-party applications
MapReduce Service (MRS)
Cloud Stream Service (CS)
Third-party applications
More

**Figure 9-14 Data forwarding process**

## 9.3.5 Application Integration

### 9.3.5.1 Application-side RESTful APIs

The IoT platform provides more than 40 RESTful APIs for third-party application developers to quickly integrate IoT platform functions. Currently, the IoT platform provides the following types of APIs, as described in Table 9-9.

**Table 9-9 Typical API categories and application scenarios**

| API Category | Description |
|---|---|
| Subscription management APIs | Used by applications to subscribe to device information. If the subscribed device information changes, the IoT platform pushes messages to the applications. |
| Tag management APIs | Used by applications to manage tags. Only device tags are supported. |
| Batch task APIs | Used by applications to perform batch operations on connected devices. Only batch software or firmware upgrades are supported. |
| Device CA certificate management APIs | Used by applications to manage device CA certificates, which are used for device access authentication. |
| Group management APIs | Used by applications to manage groups and group members. Groups are used to manage devices by group. |
| Device message APIs | Used by applications to deliver messages to devices. The difference between messages and commands is that messages can be customized and do not need to be defined in product models. |
| Product management APIs | Used by applications to manage products. Products created using APIs do not include codecs. |
| Device management APIs | Used by applications to manage devices, including adding, deleting, modifying, and querying devices and resetting device secrets. |
| Device shadow APIs | Used by applications to manage device shadows, including querying shadow data and setting desired values. |
| Device command APIs | Used by applications to deliver commands to devices. Command names must be defined in the product models. |
| Device property APIs | Used by applications to query and modify device properties. |
| Rule | Enables applications to manage rules. Different APIs are used to |

| API Category | Description |
|---|---|
| management APIs | create condition-triggered rules and scheduled rules. |

### 9.3.5.2 Application Registration Authentication

IoT applications developed based on the HUAWEI CLOUD IoT platform can call APIs provided by the platform. The platform uses token authentication provided by HUAWEI CLOUD Identity and Access Management (IAM) to ensure that only authorized users can access the platform and use resources and service suites for application development.

Before calling an API, an application must carry API credentials (such as the account name, username, and password) to obtain a token. The application then uses the token to call the API to implement services. The token is valid for 24 hours. After the token expires, the application needs to obtain a new token.

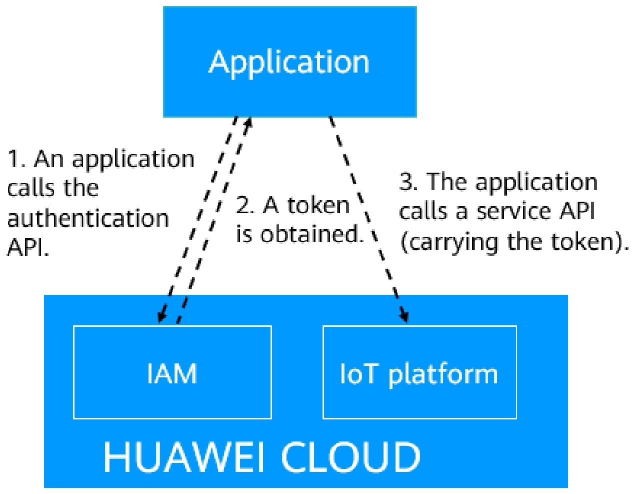

Application
1. An application calls the authentication API.
2. A token is obtained.
3. The application calls a service API (carrying the token).
IAM
IoT platform
HUAWEI CLOUD

**Figure 9-15 Application registration authentication**

### 9.3.5.3 Subscribe/Push

An application can send a subscription request to the IoT platform through an API to notify the platform of the types of notifications to receive, for example, a change in device service details, device data, or device registration.

When device details are updated on the IoT platform, the IoT platform pushes messages to the application over HTTP/HTTPS.

In any given push message, the IoT platform functions as a client and the application functions as the server. The IoT platform calls the API of the application and pushes messages to the application.

In this case, if the subscription callback URL is an HTTPS address, you need to upload the CA certificate to the IoT platform. The CA certificate is provided by the application.

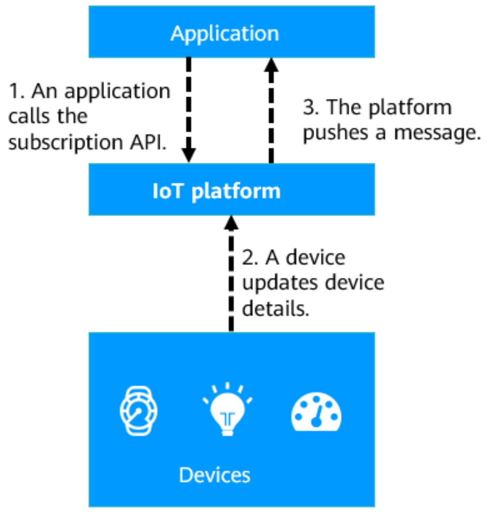
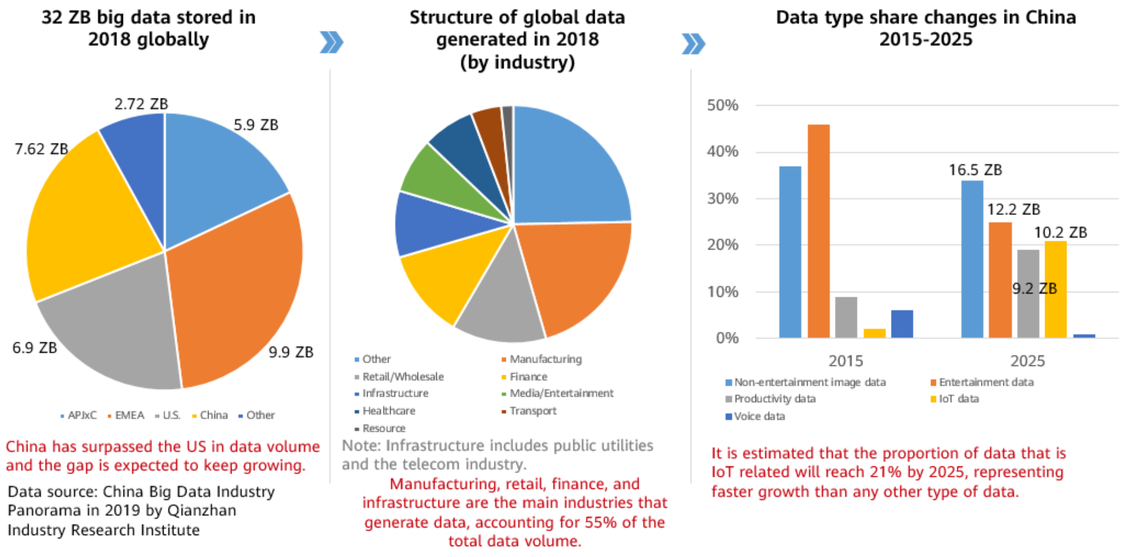

Application
1. An application calls the subscription API.
3. The platform pushes a message.
IoT platform
2. A device updates device details.
Devices

Figure 9-16 Subscribe/Push

## 9.3.6 IoT Analytics

HUAWEI CLOUD provides a series of professional IoT cloud services such as IoT Device Access and IoT Analytics. IoT Analytics is dedicated for IoT data analysis scenarios. Based on IoT asset models, the service integrates, cleans, stores, analyzes, and visualizes IoT data. One-stop IoT data analysis reduces the development threshold, shortens the development period, and accelerates IoT data monetization. Figure 9-17 shows why data analysis is required.



China has surpassed the US in data volume and the gap is expected to keep growing.

Data source: China Big Data Industry Panorama in 2019 by Qianzhan Industry Research Institute

Note: Infrastructure includes public utilities and the telecom industry.

Manufacturing, retail, finance, and infrastructure are the main industries that generate data, accounting for 55% of the total data volume.

It is estimated that the proportion of data that is IoT related will reach 21% by 2025, representing faster growth than any other type of data.

Figure 9-17 Why data analysis is required (see the Appendix for a reference image)

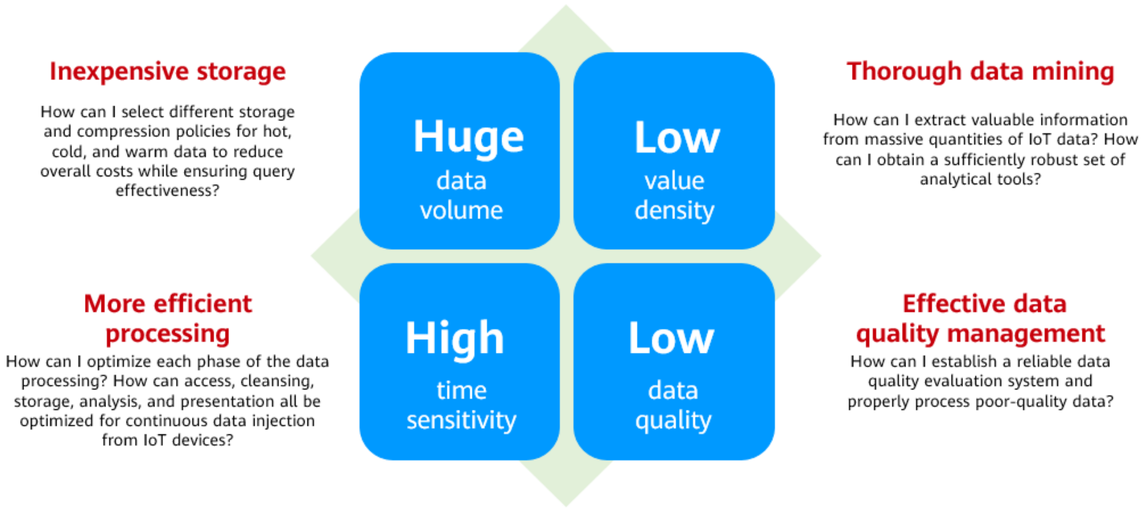

Inexpensive storage
How can I select different storage and compression policies for hot, cold, and warm data to reduce overall costs while ensuring query effectiveness?
Huge data volume
Low value density
Thorough data mining
How can I extract valuable information from massive quantities of IoT data? How can I obtain a sufficiently robust set of analytical tools?
More efficient processing
How can I optimize each phase of the data processing? How can access, cleansing, storage, analysis, and presentation all be optimized for continuous data injection from IoT devices?
High time sensitivity
Low data quality
Effective data quality management
How can I establish a reliable data quality evaluation system and properly process poor-quality data?

**Figure 9-18 IoT data characteristics (see the Appendix for a reference image)**

- Asset model

  Building an asset model is the basis for understanding IoT data. Building an asset model is to establish relationships between things, between things and space, and between things and people, so that data can be understood in context. An asset model is the mapping of assets in the physical world to the digital world. Data on both sides is synchronized in quasi-real time to implement digital twins. IoT Analytics abstracts data reported by different devices based on the asset model and converts the data into a format that can be understood by services. See Figure 9-19.

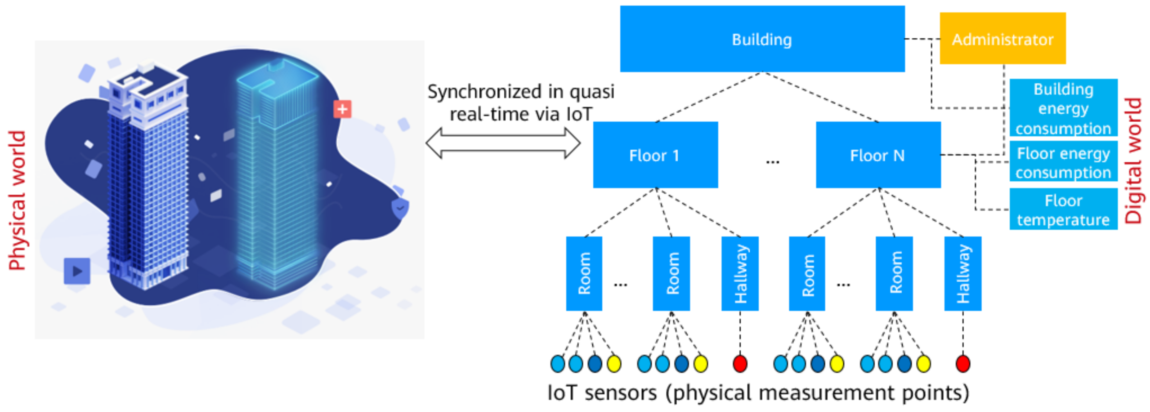


**Figure 9-19 Asset model (see the Appendix for a reference image)**

- Time series data processing is the key

  Write performance: How do we meet the requirements for high concurrency and real-time write for a large number of devices?

  Compression ratio: Some IoT devices may generate a large amount of data. Higher compression directly reduces costs.

Query efficiency: How do we meet high-performance query requirements, especially time-based aggregation query, for IoT data accumulated over a long period of time?

- Multi-temperature data management maximizes processing efficiency

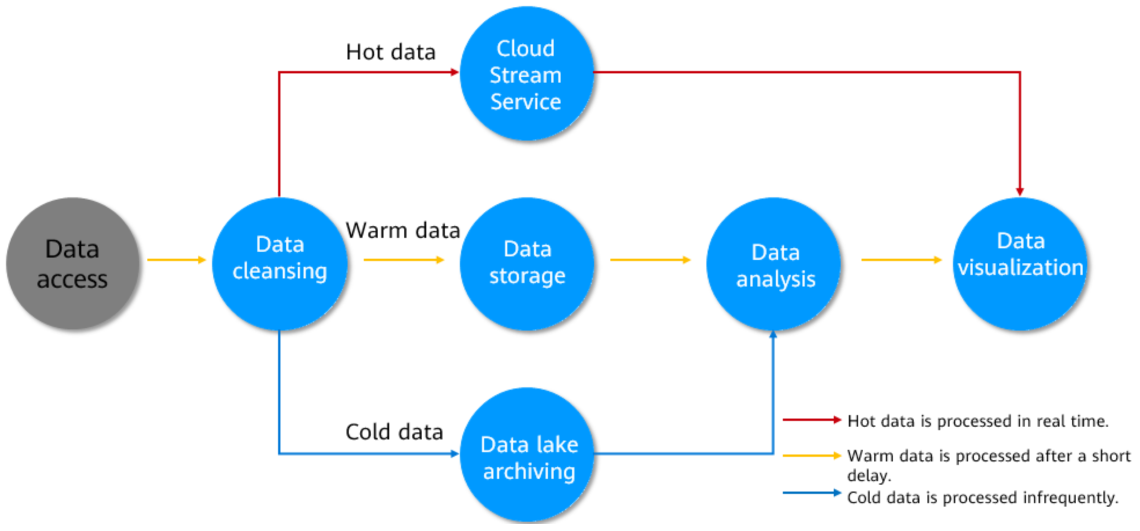


**Figure 9-20 Data processing by layer**

- Efficient data cleansing provides high-quality data for analysis

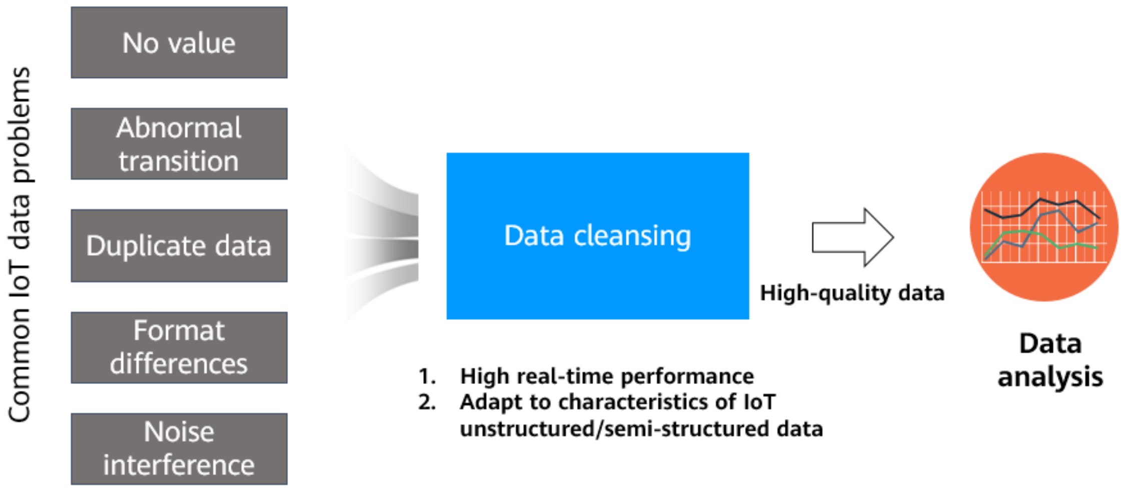


**Figure 9-21 Data cleansing**

Unlike the solution of forwarding device data to the general data analysis service, IoT Analytics is designed for IoT scenarios.

IoT Analytics integrates, archives, and analyzes data from IoT Device Access and third-party services. The analysis engine understands data based on asset models and supports real-time, time series, and offline analysis. In addition to data analysis, IoT Analytics provides industry analysis suites for industries like smart logistics and new-energy vehicles. The analyzed data can be used by the AI platform or provided to third-party applications or other HUAWEI CLOUD services. Figure 9-22 shows the architecture of IoT Analytics.

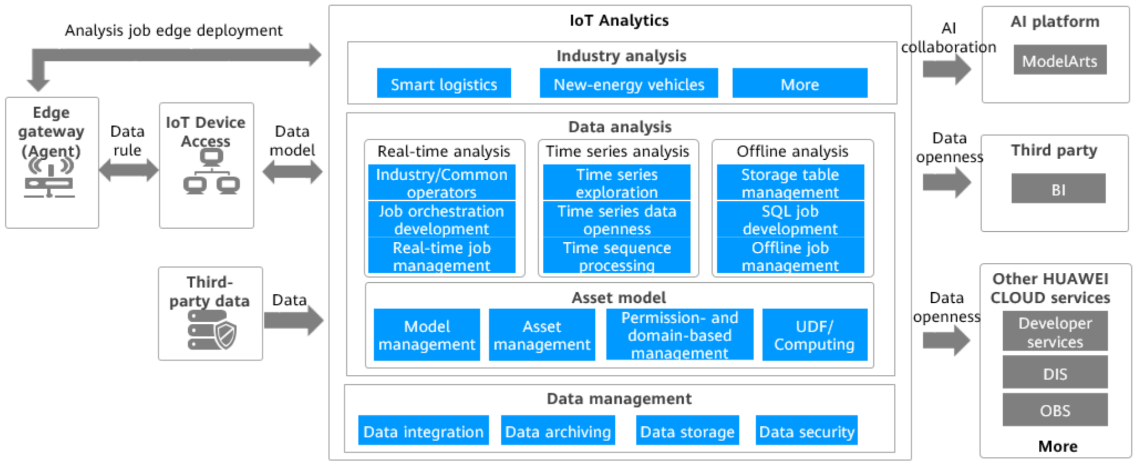


**Figure 9-22 IoT Analytics structure (see the Appendix for a reference image)**

- Offline analysis

  IoT data developers can quickly build an IoT data lake. They can use standard SQL jobs to develop IoT data analytics tasks and easily process TBs or even EBs of IoT data.

  The data lake supports cost-effective, massive data storage. It can seamlessly connect to IoT Device Access for instant access to a wealth of IoT data sources. It can also preprocess IoT data based on the IoT asset models to prepare for data analysis.

  With the highly available and horizontally expandable framework of big data, big data analytics uses the in-memory computing model, directed acyclic graph (DAG) scheduling framework, and efficient optimizer to deliver a comprehensive performance 100 times higher than that of the traditional MapReduce model. Developers can complete batch analysis on IoT data with ease.

  Standard SQL APIs are provided. IoT data developers do not need to worry about the deployment and O&M of the SQL processing engine. They only need to focus on IoT services, develop analysis jobs, and configure job scheduling policies. See Figure 9-23.

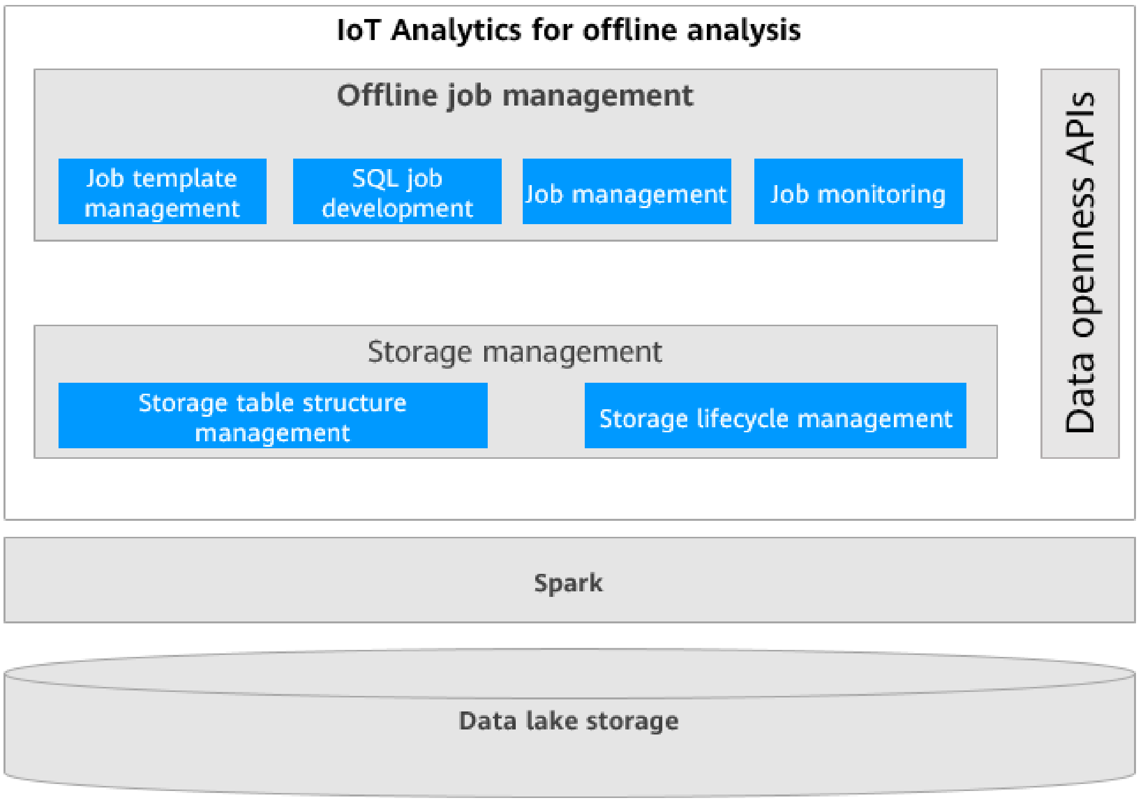


**Figure 9-23 Offline analysis**

- Real-time analysis

  Real-time analysis on IoT data is provided based on the big data stream computing engine. To reduce skill requirements for stream analysis job development, IoT Analytics provides graphical stream orchestration so you can quickly develop and roll out services by dragging and dropping components.

  Graphical stream orchestration: Visual stream orchestration IDE allows you to define IoT stream analysis jobs through drag-and-drop operations without writing SQL statements.

  IoT stream orchestration operators: Based on typical IoT application scenarios, common operators are encapsulated, such as data access, data filtering, and data conversion. See Figure 9-24.

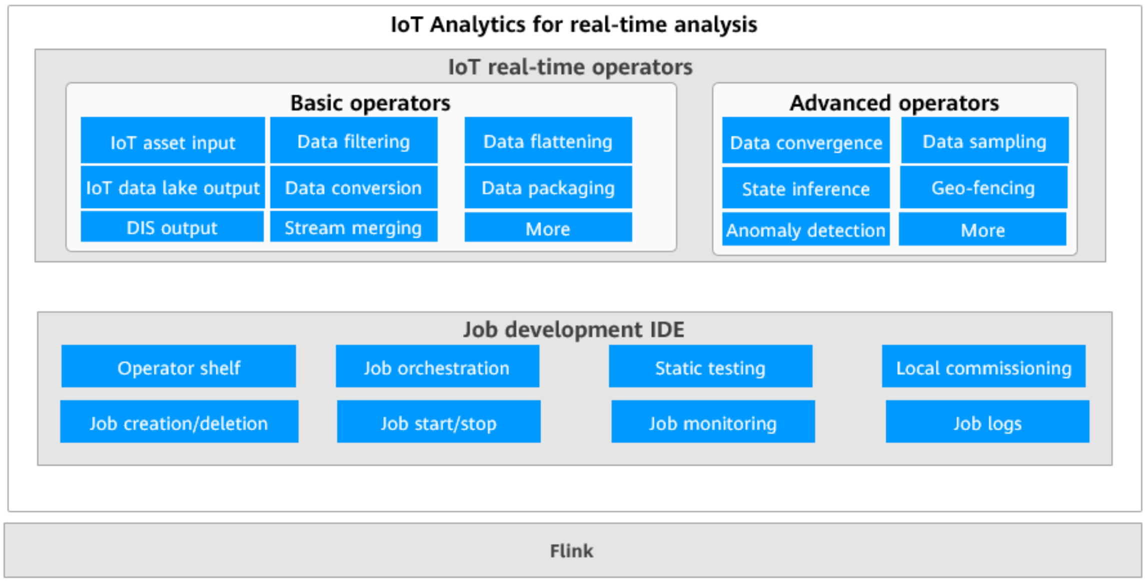
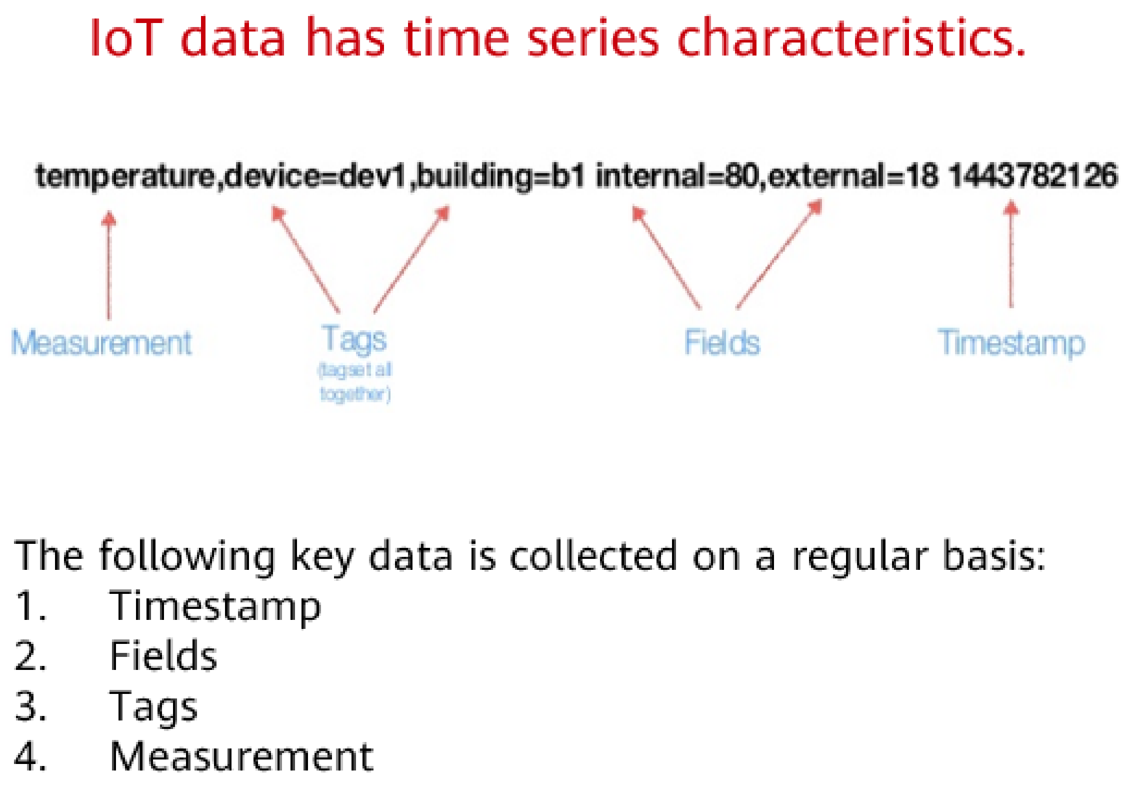


**Figure 9-24 Real-time analysis (see the Appendix for a reference image)**

- Time series analysis

  Figure 9-25 illustrates the time series characteristics of IoT data.



IoT data has time series characteristics.

temperature,device=dev1,building=b1 internal=80,external=18 1443782126
Measurement
Tags
(tagset all together)
Fields
Timestamp

The following key data is collected on a regular basis:

1. Timestamp
2. Fields
3. Tags
4. Measurement

**Figure 9-25 Time series characteristics of IoT data**

You can use IoT time series data processing functions, for example, time series data storage with high compression ratio, efficient time series query, and massive timeline.

- Massive access: hundreds of millions of timelines
- Time series storage: columnar storage and dedicated compression algorithms with a high compression ratio
- Efficient query: Time-based multi-dimensional aggregation, and near-real-time analysis and query
- Data visualization: Time series insight tools provided for IoT data analysts to explore time series data

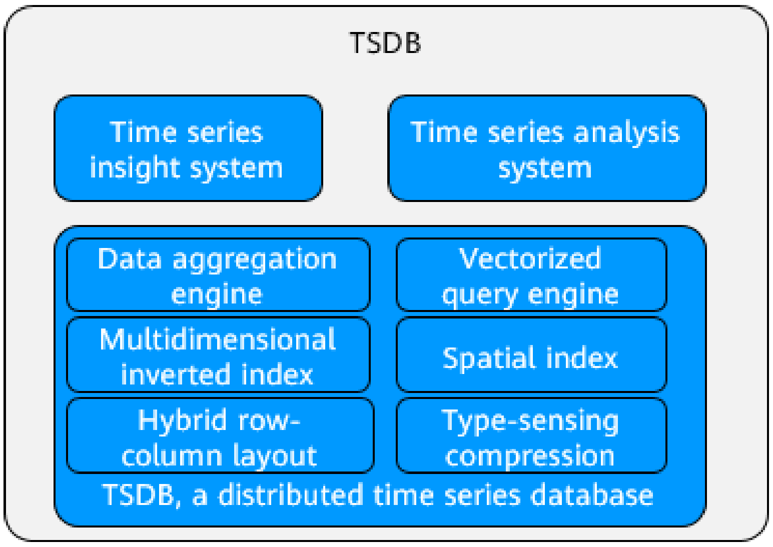

TSDB
Time series insight system
Time series analysis system
Data aggregation engine
Vectorized query engine
Multidimensional inverted index
Spatial index
Hybrid row-column layout
Type-sensing compression
TSDB, a distributed time series database

**Figure 9-26 Time series analysis**

## 9.3.7 IoT Studio

IoT Studio is a one-stop application development platform for IoT developers. It provides visualized development and online application hosting to help enterprises quickly build web applications and easily manage global devices.

Pain Points

Small- and medium-sized enterprises want to quickly build applications with limited up-front investment.

Small- and medium-sized device manufacturers have limited personnel available for software development. Software development, application construction, and IoT service rollout are slow and expensive.

Key Features

- Quick build: An IoT web application can be built in just 5 to 10 minutes, thanks to 30+ visualized components
- Low cost: on-demand, free industry templates
- Easy O&M: professional O&M support for application hosting from Huawei

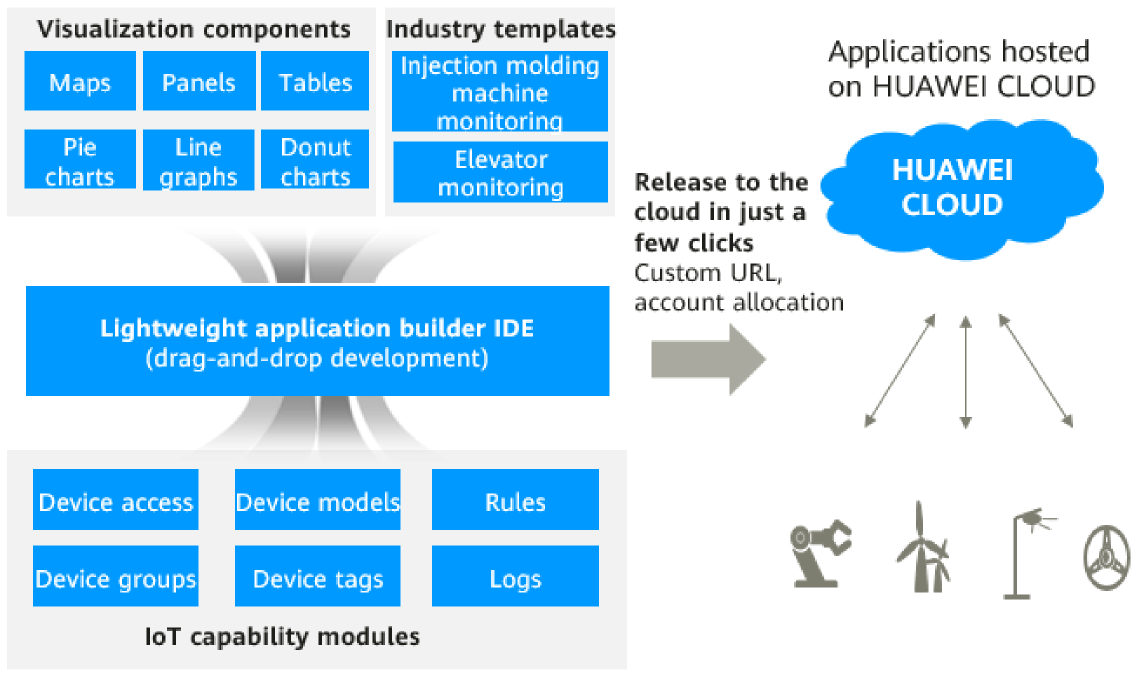

Visualization components
Maps
Panels
Tables
Pie charts
Line graphs
Donut charts
Industry templates
Injection molding machine monitoring
Elevator monitoring
Lightweight application builder IDE
(drag-and-drop development)
Device access
Device models
Rules
Device groups
Device tags
Logs
IoT capability modules
Release to the cloud in just a few clicks
Custom URL, account allocation
Applications hosted on HUAWEI CLOUD
HUAWEI CLOUD

**Figure 9-27 IoT Studio**

## 9.3.8 Other Related Services

### 9.3.8.1 Edge Computing

Edge computing addresses the pain points of direct device-cloud interconnection.

An edge is a part of an entity or a logical concept that is far away from the center and close to the boundary. In the data processing field, edge computing is derived from cloud computing. It refers to the process of building compute nodes that integrate network, compute, storage, and application functions on the side close to data sources to provide data processing nearby, instead of transferring all data to the cloud for processing. Cloud computing aims to centralize data to the cloud for processing. Why do we need edge computing? Let's use an IoT scenario as an example.

Let's say an enterprise wants to upgrade its industrial park into a smart campus and connect all devices in the campus to the IoT platform for unified management. There are too many types of devices such as video surveillance devices, access control devices, firefighting devices, and water and electricity devices. All these devices report data based on their own industry protocols. The IoT platform supports a limited number of communications protocols. A heavy workload is required to convert these industry-specific protocols into those supported by the IoT platform.

In addition, employee data in the access control system is private data, and transferring such data to the cloud violates the security policies of the enterprise. Connecting all systems except the access control system to the cloud would result in high maintenance costs and poor linkage between systems.

Let's say a vehicle enterprise has encountered similar problems. This enterprise produces self-driving vehicles that interact with the IoT platform through an in-vehicle system to obtain self-driving instructions. Unstable network signals on the road are a big problem. Due to poor signals, it may take several seconds for a vehicle to report data to the cloud and receive an instruction from the cloud. This latency may result in death or injury.

The preceding problems are the pain points of direct cloud-device interconnection. They can be solved by edge computing.

What can edge computing do in the IoT?

- Open ecosystem

  Edge computing is a concept. When discussing what it can do, we need to assign it an entity. In this section, we will use the IoT Edge service provided by the Huawei IoT platform as the entity. IoT Edge is deployed at the edge close to devices or data sources to provide real-time services. It integrates network, compute, storage, and application capabilities to make applications intelligent and data secure. The core capabilities of IoT Edge can be described from three aspects.

  An edge node is a server with independent access and compute functions. It is like the "cerebellum" of the IoT. IoT Edge does not require a specific physical object. As long as the hardware meets the requirements, the IoT Edge software package can be deployed in Docker container mode to support device access, device linkage, and low-latency local management.

  IoT Edge allows users to develop third-party access protocol drivers and deliver them to edge nodes through the cloud so that devices using these protocols can access the platform. IoT Edge also allows users to develop and deploy third-party edge container applications to expand the function of edge nodes. With open hardware, protocols, and applications, IoT Edge can meet custom requirements in different IoT scenarios.

- Edge-cloud synergy

  Although edge nodes have independent computing functions, the value of the "cerebellum" can be brought into full play only through cooperation with the "cerebrum". Edge-cloud synergy is the trump card of IoT Edge. IoT Edge provides a graphical operation interface on the cloud. On this interface, you can manage edge nodes, and perform unified application deployment, node O&M, and service management.

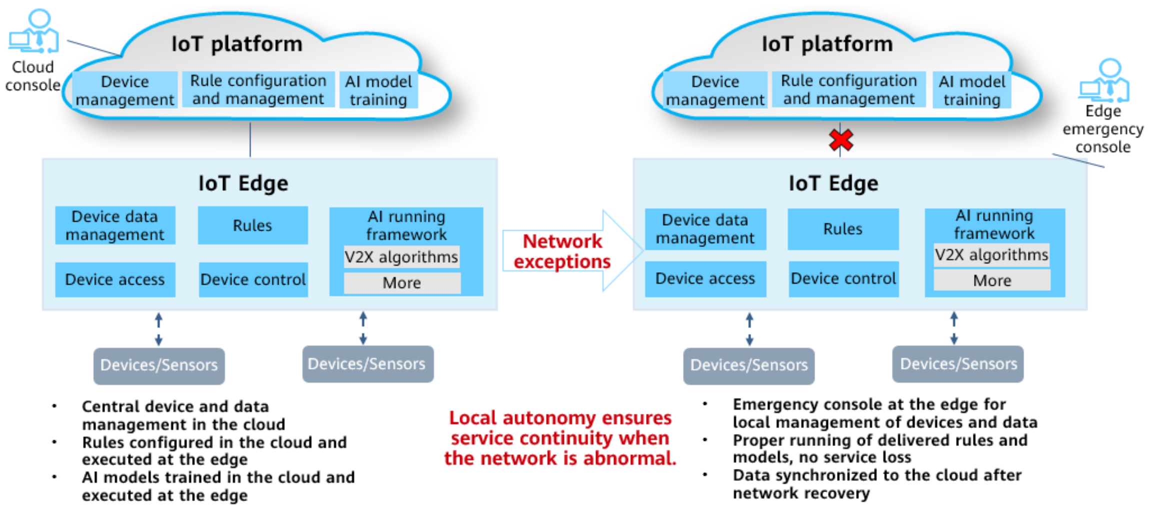

Cloud console
IoT platform
Device management
Rule configuration and management
AI model training
IoT Edge
Device data management
Rules
AI running framework
V2X algorithms
More
Device access
Device control
Devices/Sensors
Devices/Sensors
Network exceptions
IoT platform
Device management
Rule configuration and management
AI model training
Edge emergency console
IoT Edge
Device data management
Rules
AI running framework
V2X algorithms
More
Device access
Device control
Devices/Sensors
Devices/Sensors
• Central device and data management in the cloud
• Rules configured in the cloud and executed at the edge
• AI models trained in the cloud and executed at the edge
Local autonomy ensures service continuity when the network is abnormal.
• Emergency console at the edge for local management of devices and data
• Proper running of delivered rules and models, no service loss
• Data synchronized to the cloud after network recovery

**Figure 9-28 Edge computing features (see the Appendix for a reference image)**

After an edge node is managed, it can work with the cloud to implement layered service processing. For example, in IoV scenarios, for services that need to be processed in a timely manner, such as automated driving and V2X, the edge nodes directly calculate data and return the result. For services that are insensitive to latency and involve a large amount of data, such as large-screen monitoring and big data analytics, data is transferred to the cloud for processing. Another example is campus scenarios where private user data is collected, processed, and stored on the local node. After being cleaned and summarized, non-private data is uploaded to the cloud for machine learning and training, so local intelligent algorithms can be continuously optimized. Layer-based service processing maximizes the value of the IoT and powers IoE.

The edge and cloud are connected through the network. If the network is faulty, the edge is disconnected from the cloud. How do we ensure edge services in this case? IoT Edge provides local autonomy for edge nodes. If the network is abnormal, the rules and AI models delivered by the cloud can still run properly at the edge to ensure service continuity. After the network is recovered, the data is synchronized to the cloud. In addition, the edge node provides a local console, which can be logged in to without connecting to the network, allowing for management autonomy.

- Edge intelligence

  The edge can clean data. Specifically, the edge filters, deduplicates, and aggregates data reported by devices and then reports the data to the cloud. Data cleaning is applicable to scenarios where data needs to be selectively reported to the cloud to reduce the bandwidth, storage, and computing resources used.

  Filter criteria are used for data filtering. For example, for a filter criterion with property A greater than 10, all data that meets this criterion is filtered. You can specify multiple criteria, with the relationship between them set to AND or OR. If the relationship is AND, the data that meets all the criteria is filtered. If the relationship is OR, the data that meets any criterion is filtered. If a device continuously reports messages with duplicate property values, the edge node reports only the first message to the cloud. This is deduplication. Aggregation means that the edge node aggregates data reported by each device in a time window (for example, one hour) into one piece of data. The user can specify the aggregation method for each property in the data, for example, obtaining the maximum/minimum value, the sum, and the average. The priorities of the three types of data cleaning rules are as follows: filtering > deduplication > aggregation. If you set all the three types of rules, data is filtered, deduplicated, and then aggregated before being reported.

### 9.3.8.2 Global SIM Link

Global SIM Link provides cellular IoT directional data plans and intelligent network selection using eSIM/vSIM cards. The service enables customers to use local data plans on a platform that is globally accessible with single-point access.

Pain points:

- Difficult deployment: IoT devices are installed in different locations, such as on the top of a building or in a passageway. Cabling and ditch digging are not allowed in these locations, and wireless connection is required.
- Difficulty obtaining cost-effective data plans in foreign countries: If an enterprise produces IoT devices in country A and sells them in country B, they need to use data plans of carriers in country B. The negotiation process is time-consuming.
- Difficulty changing SIM cards: IoT devices may be deployed in locations covered by signals from multiple carrier networks. They are expected to use a network with a better signal without changing SIM cards.
- Difficult card management: Enterprises need to connect to multiple carriers around the world in SIM card management. This poses great challenges to integration.

Solution:

Global SIM Link provides 2G/3G/4G and NB-IoT connections for IoT devices and IoT SIM card data plans in and outside of China (covering more than 100 countries and regions). Remote eSIM/vSIM provisioning enables on-demand selection of networks operated by more than 80 carriers around the world.

## 9.3.9 Characteristics of the HUAWEI CLOUD IoT Platform

### 9.3.9.1 Access Agnostic

The IoT platform uses multi-layer ID management and authentication to manage IDs of developers, applications, devices, and users. It ensures seamless compatibility through precise database model selection and collaboration between load balancing and data management modules. The IoT platform provides both wireless and wired access to a variety of devices and mainstream IoT gateways.

### 9.3.9.2 Reliable

- System reliability

  The IoT platform is installed and deployed based on HUAWEI CLOUD. The system reliability includes system cluster and VM reliability, as described in Table 9-10.

**Table 9-10 System reliability**

| Item | Description |
|---|---|
| Full system cluster | Each node in a cluster is equivalent and runs the whole system. Based on different networking, nodes in a cluster can be expanded as hardware is expanded.<br>The front-end load balancer in the cluster can evenly send user requests to the IoT platform. Specifically, the load balancer distributes and schedules user requests based on the corresponding algorithm to achieve load balancing in a cluster. |
| VM reliability | Installation and deployment of the cloud platform are supported. On the cloud platform, VMs are deployed in two modes:<br>1+1: used to deploy the active and standby instances of one service in two VMs. |

| Item | Description |
|---|---|
| | N+M: used to deploy a group of load-sharing services in multiple VMs. |

- Network reliability

  Table 9-11 describes the network reliability policies of the IoT platform.

**Table 9-11 Network reliability policies**

| Item | Description |
|---|---|
| 1+1 mutual backup | Half of the capacity of each site is reserved for taking over services from the other site. |
| Pool mode | The number of sites is N. Each site has 1/N idle capacity to take over services from a faulty site. |

- Service reliability

**Table 9-12 Service reliability policies**

| Item | Description |
|---|---|
| Flow control | Communication is interrupted or suspended to ensure normal service running when the system is overloaded. |
| Data encryption | Sensitive data is encrypted before being stored. For example, phone numbers are encrypted or obfuscated before being stored. Secure storage configuration is added for device data items in service domain objects. Security processing is performed based on the secure data configuration. |
| Number shielding | Phone numbers are shielded when users query user data or export user lists on the web portal. |

- Module reliability

**Table 9-13 Module reliability policies**

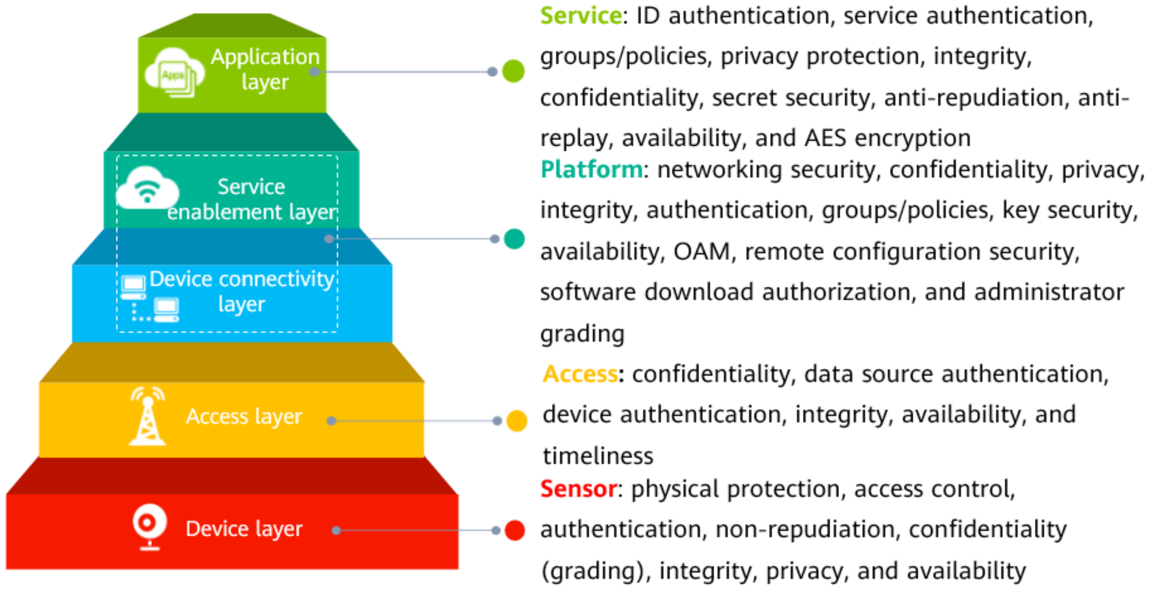
| Item | Description |
|---|---|
| Rule engine module | Embedded in applications, the rule engine module allows users to make policy decisions as needed. Scalability and maintainability of the IoT platform are enhanced. |
| API Server module | OAuth authorization provides security isolation between users and application servers. Only authorized users can access services provided by the application servers. |

| Item | Description |
|---|---|
| Database module | The database module of the IoT platform uses the active-standby pattern to ensure data reliability. |

### 9.3.9.3 Secure

The IoT platform has its own security strategies, including device, networking, service, access, data storage, and access security strategies.

The IoT platform provides open security capabilities, including ID authentication, authorization, and security associations. The security strategies are designed based on security specifications defined by standards organizations and best practices in the industry. These specifications include ITU X.805 hierarchy and threat analysis, oneM2M/ETSI security solutions and risks, 3GPP access security, IETF TLS/DTLS, OWASP security standards, and STRIDE security threat identification methods. The security strategies are also designed using the minimum authorization and multi-level protection principles. See Figure 9-29.



Service: ID authentication, service authentication, groups/policies, privacy protection, integrity, confidentiality, secret security, anti-repudiation, anti-replay, availability, and AES encryption

Platform: networking security, confidentiality, privacy, integrity, authentication, groups/policies, key security, availability, OAM, remote configuration security, software download authorization, and administrator grading

Access: confidentiality, data source authentication, device authentication, integrity, availability, and timeliness

Sensor: physical protection, access control, authentication, non-repudiation, confidentiality (grading), integrity, privacy, and availability

**Figure 9-29 Secure**

### 9.3.9.4 Scalable

The IoT platform uses a cloud-based architecture and can be easily deployed on the cloud platform to provide a variety of services.

Flexible platform deployment: The IoT platform can be flexibly deployed on servers or the cloud.

Aggregated vertical capabilities: The IoT platform can serve one or more industries.

Compatibility: The IoT platform is compatible with smart devices from different manufacturers.

Modularized services and flexible combination: A single service or multiple services can be deployed.

#### 9.3.9.5 Open

The IoT platform provides a variety of open APIs, including communication capability, network capability, and user data APIs. The IoT platform also provides open device management capabilities and an entrance for end users to access the platform and perform operations. It provides SDKs so that developers only need to drag and drop SDKs to implement service logic.

Open APIs: The IoT platform provides a variety of open APIs for third-party application integration, eliminating repeated development.

Diverse IoT Agents: Agents are provided to adapt to different OSs and hardware.

Pre-integration with high-value industries: The IoT platform provides partners with pre-integrated high-value industry services, including smart home, IoV, and smart metering.

Open device management: Service providers use a web-service interface to access and remotely operate devices.

# 10 IoT Platform Secondary Development

## 10.1 Introduction to Platform Secondary Development

The HUAWEI CLOUD IoT platform enables southbound and northbound data exchange. Developers need to perform secondary development using this platform to provide end-to-end IoT services.

## 10.2 Product Development

In the IoT platform integration solution, the IoT platform provides open APIs for applications to connect devices that use various protocols. To provide richer device management capabilities, the IoT platform needs to understand the device capabilities and the formats of data reported by devices. Therefore, you need to develop product models and codecs on the IoT platform. To create an IoT solution based on the HUAWEI CLOUD IoT platform, you must perform the operations described in Table 10-1.

**Table 10-1 Device access process**

| Operation | Description |
|---|---|
| Product development | Manage products, develop product models and codecs, and perform online debugging on the IoT Device Access (IoTDA) console. |
| Development on the application side | Carry out development for interconnection between applications and the platform, including calling APIs, obtaining service data, and managing HTTPS certificates. |
| Development on the device side | Integrate and interconnect devices with the IoT platform, including connecting devices to the IoT platform, reporting service data to the platform, and processing commands delivered by the platform. |

The process of using IoTDA, including product, application, device, and routine management.

**Product development**: You can perform development operations on the IoTDA console. For example, you can create a product or device, develop a product model or codec online, perform online debugging, carry out self-service testing, and release products.

**Development on the device side**: You can connect devices to the platform by integrating SDKs or modules, or using native protocols.

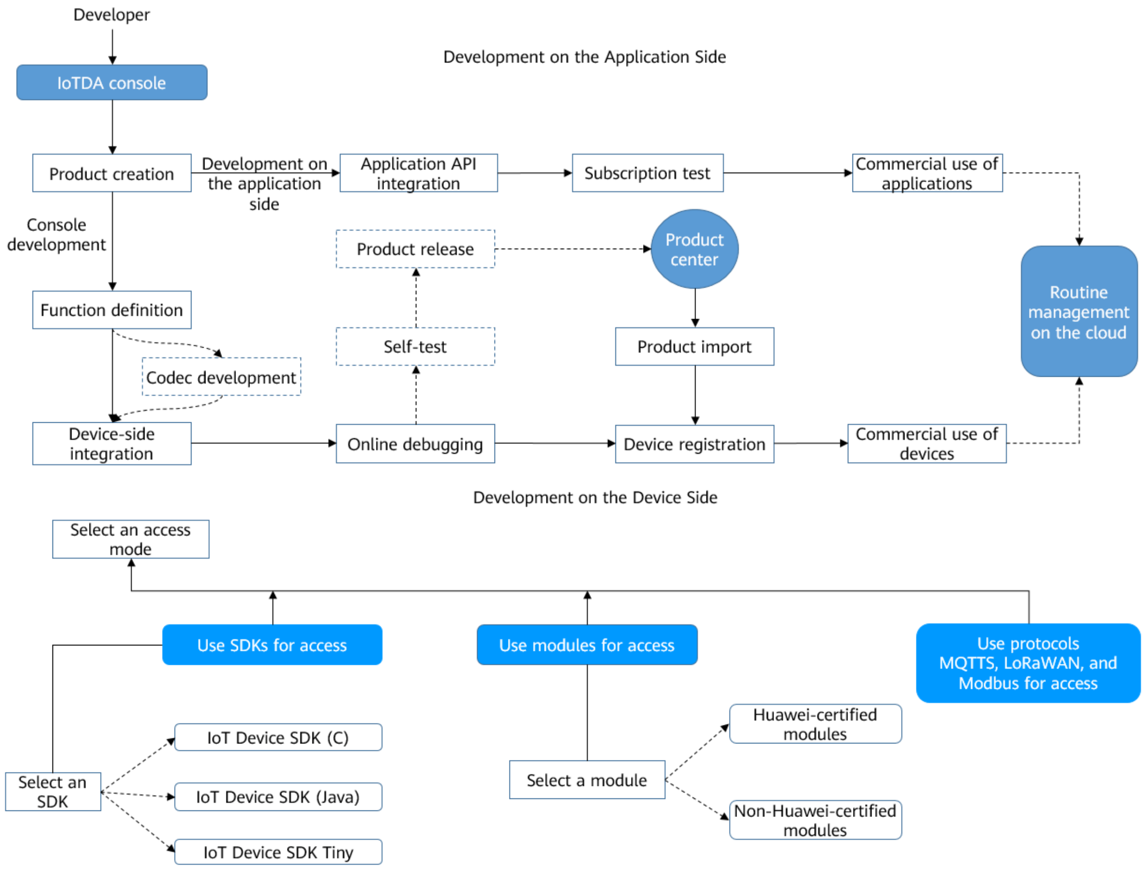


**Figure 10-1 IoT solution development process based on the IoT platform (see the Appendix for a reference image)**

**Development on the application side**: The platform provides robust device management capabilities through APIs. You can develop applications based on the APIs to meet requirements in different industries such as smart city, smart campus, smart industry, and IoV.

**Routine management**: After a physical device is connected to the platform, you can perform routine device management on the IoTDA console or by calling APIs.

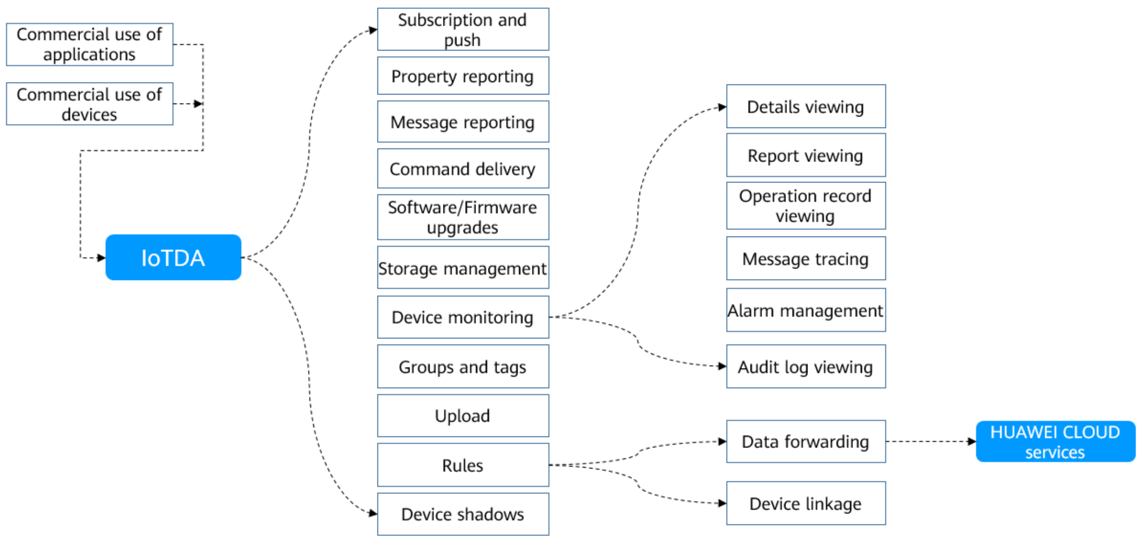


**Figure 10-2 Routine management in the cloud**

## 10.2.1 Product Model

A product model, also known as a profile, defines the properties of a device, such as the color, size, collected data, identifiable commands, and reported events. The manufacturer, device type, and device model are used together to uniquely identify a product model. You can easily develop product models on the IoTDA console without writing any code.

A product model is a file that describes what a device is, what it can do, and how to control it. You can build an abstract model of a device by defining a product model on the IoT platform so that the platform can know what services, properties, and commands are supported by the device, such as its color or any on/off switches. After defining a product model, you can use it during device registration.

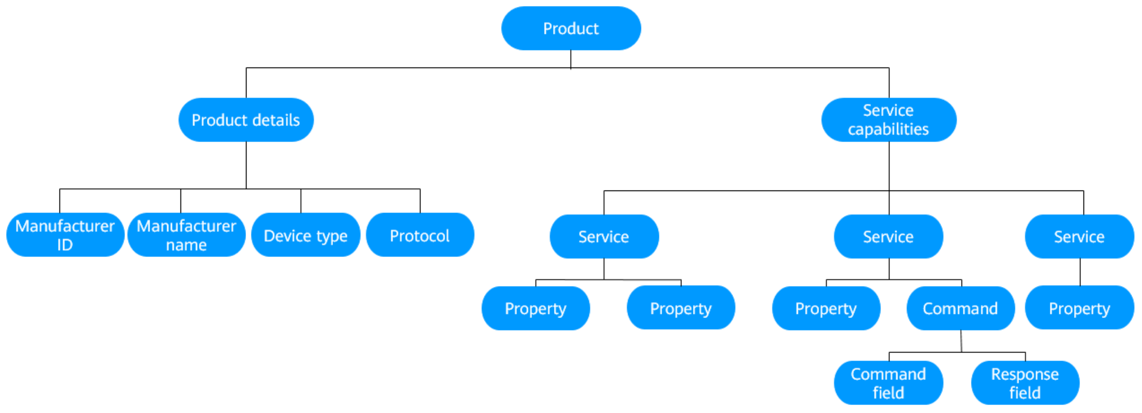


**Figure 10-3 Product model (see the Appendix for a reference image)**

On the IoT platform, the product model is the key to device access. It contains the capabilities and services of a device and the data formats of upstream and downstream device messages. For example, when a device reports data to the IoT platform, the IoT platform matches the product model based on keywords of the reported data and verifies the data format. Only data that is matched is saved on the IoT platform. If the reported data is not matched with the configuration in the product model, the data is considered invalid and dropped.

Product details:

Product details describe basic information about a device, including the manufacturer ID, manufacturer name, device type, and protocol. For example, the manufacturer name of a water meter could be HZYB, the manufacturer ID TestUtf8ManuId, the device type WaterMeter, and the protocol CoAP.

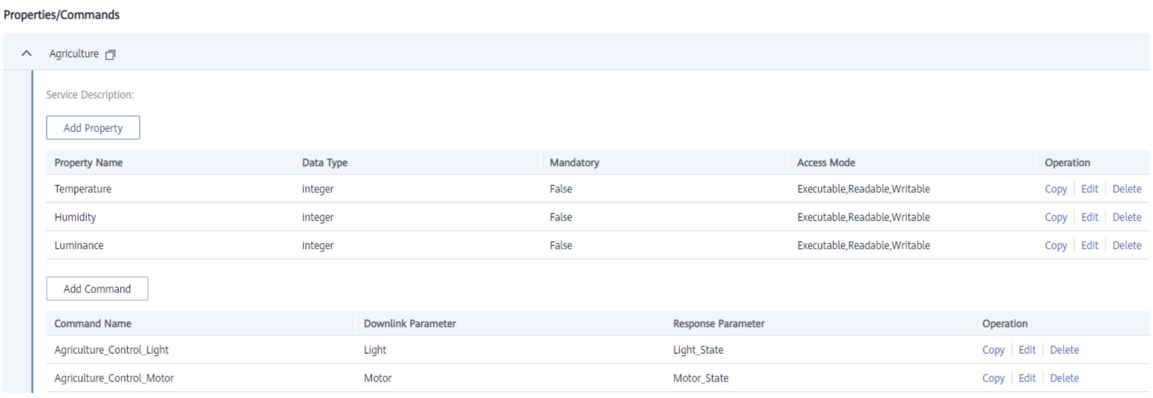
Service capabilities:

Service capabilities are divided into several services. The properties, commands, and command parameters of each service are defined in the product model. Take a water meter as an example. It has multiple capabilities, such as reporting data about the water flow, alarms, power, and connections, and receiving commands from a server. When describing the capabilities of a water meter, the profile includes five services, each of which has its own properties or commands.

#### 10.2.1.2 Product Model Example: Smart Water Meter

**Table 10-2 Water meter service capabilities**

| Service | Description |
|---|---|
| Basics (WaterMeterBasic) | Used to define parameters reported by the water meter, such as the water flow, temperature, and pressure. If these parameters need to be controlled or modified using commands, you also need to define parameters in the commands. |
| Alarm (WaterMeterAlarm) | Used to define data reported by the water meter in various alarm scenarios. Commands need to be defined if necessary. |
| Battery (Battery) | Used to define data including the voltage and current intensity of the water meter. |
| Transmission rule (DeliverySchedule) | Used to define transmission rules for the water meter. Commands need to be defined if necessary. |
| Connectivity (Connectivity) | Used to define connection parameters of the water meter. |



Properties/Commands

Agriculture

Service Description:

| Property Name | Data Type | Mandatory | Access Mode | Operation |
|---|---|---|---|---|
| Temperature | integer | False | Executable,Readable,Writable | Copy \| Edit \| Delete |
| Humidity | integer | False | Executable,Readable,Writable | Copy \| Edit \| Delete |
| Luminance | integer | False | Executable,Readable,Writable | Copy \| Edit \| Delete |

| Command Name | Downlink Parameter | Response Parameter | Operation |
|---|---|---|---|
| Agriculture_Control_Light | Light | Light_State | Copy \| Edit \| Delete |
| Agriculture_Control_Motor | Motor | Motor_State | Copy \| Edit \| Delete |

**Figure 10-4 Product model example (see the Appendix for a reference image)**

The IoT platform provides multiple methods for developing product models. You can select one that suits to your needs.

- Importing models (preset product models on the platform)
- Uploading a profile (offline development)
- Importing models in an Excel file
- Customizing functions (online development)

## 10.2.2 Codec

### 10.2.2.1 Overview

What Is a Codec?

The codec decodes binary data reported by devices into JSON data that can be read by applications. It also encodes commands in JSON format delivered by applications into binary data that can be executed by devices.

Why Is a Codec Used?

NB-IoT devices use data in binary or TLV format.

CoAP is used for communications between NB-IoT devices and the IoT platform. The payload of CoAP messages carries data at the application layer, at which the data type is defined by the devices. Because NB-IoT devices have high requirements on power consumption, their application layer data is generally not in JSON format.

Applications do not understand data in binary or TLV format.

How Do We Develop a Codec?

The IoT platform provides three methods for developing codecs. Offline codec development is complex and time-consuming. Graphical codec development is recommended.

**Graphical development**: The codec of a product can be quickly developed in a visualized manner on the IoTDA console.

**Offline Development**: A codec is developed through secondary development based on the Java codec demo to implement encoding, decoding, packaging, and quality inspection.

**Script-based development**: JavaScript scripts are used to implement encoding and decoding.

The IoT platform abstracts and encapsulates the original codec development code. Therefore, developers can develop codecs simply by defining the format of code streams reported by devices and mapping the properties in the code streams and the profiles in a graphical way. When the development is complete, the codec is automatically generated and can be deployed on the IoT platform.

For an upstream message sent by a device, the CoAP packet is parsed to obtain the application layer data, the codec provided by the device manufacturer is invoked to decode the message, and then the message is sent to an application. For a downstream message sent by an application, the codec provided by the device manufacturer is invoked to assemble a CoAP message, and then the message is sent to the device. See Figure 10-5 for details. The codec also encodes commands and responses to reported device data sent by the platform.

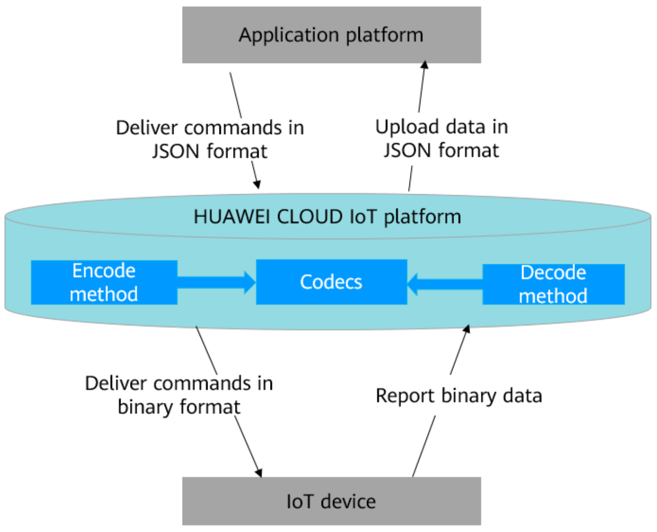

Application platform
Deliver commands in JSON format
Upload data in JSON format
HUAWEI CLOUD IoT platform
Encode method
Codecs
Decode method
Deliver commands in binary format
Report binary data
IoT device

**Figure 10-5 Functions of codec in NB-IoT scenarios**

### 10.2.2.2 Data Reporting

The message processing flow includes data reporting and command delivery, as shown in Figure 10-6.

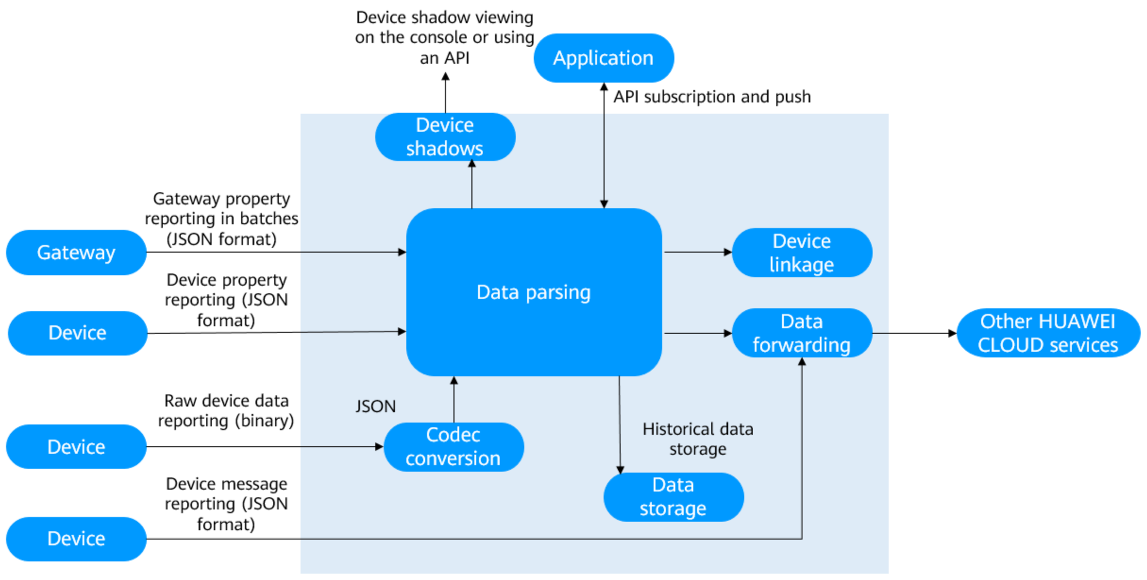

Device shadow viewing on the console or using an API
Application
API subscription and push
Device shadows
Gateway property reporting in batches (JSON format)
Gateway
Data parsing
Device linkage
Device property reporting (JSON format)
Device
Data forwarding
Other HUAWEI CLOUD services
Raw device data reporting (binary)
JSON
Device
Codec conversion
Historical data storage
Data storage
Device message reporting (JSON format)
Device

**Figure 10-6 Data reporting process (see the Appendix for a reference image)**

After being registered with the IoT platform and powered on, a device can collect and report data based on the service logic. Data collection and reporting can be triggered by a schedule or by specific events. Data can be sent to the platform through the following APIs:

Reporting device messages: A device reports custom data to the platform through message reporting APIs. The platform does not parse or store the messages reported. Instead, it forwards the messages to other HUAWEI CLOUD services for storage and processing based on data forwarding rules. Then, the data is further processed through the consoles or APIs of the other services.

Raw device data (binary) reporting: A device reports raw code streams through binary reporting APIs. The platform uses codecs to parse the raw data into data in JSON format defined by the product model and reports the parsed data to IoTDA for service processing.

Device property reporting: A device reports property data defined in the product model through property reporting APIs. The platform parses the data and reports the data to IoTDA for service processing.

Batch property reporting: A gateway reports data of a batch of devices to the platform at a time. The platform parses the data and reports the data to IoTDA for service processing.

In the data reporting process, the codec is used in the following scenarios:

1. Decoding binary data reported by a device into JSON data and sending the data to an application.
2. Encoding the JSON data returned by an application into binary data and sending the data to a device.

### 10.2.2.3 Command Delivery

The IoT platform delivers a command to a device, and the device responds to and executes the command. In this way, the IoT platform can remotely control the device.

The following figures show immediate delivery and delayed delivery of LwM2M/CoAP and MQTT commands.

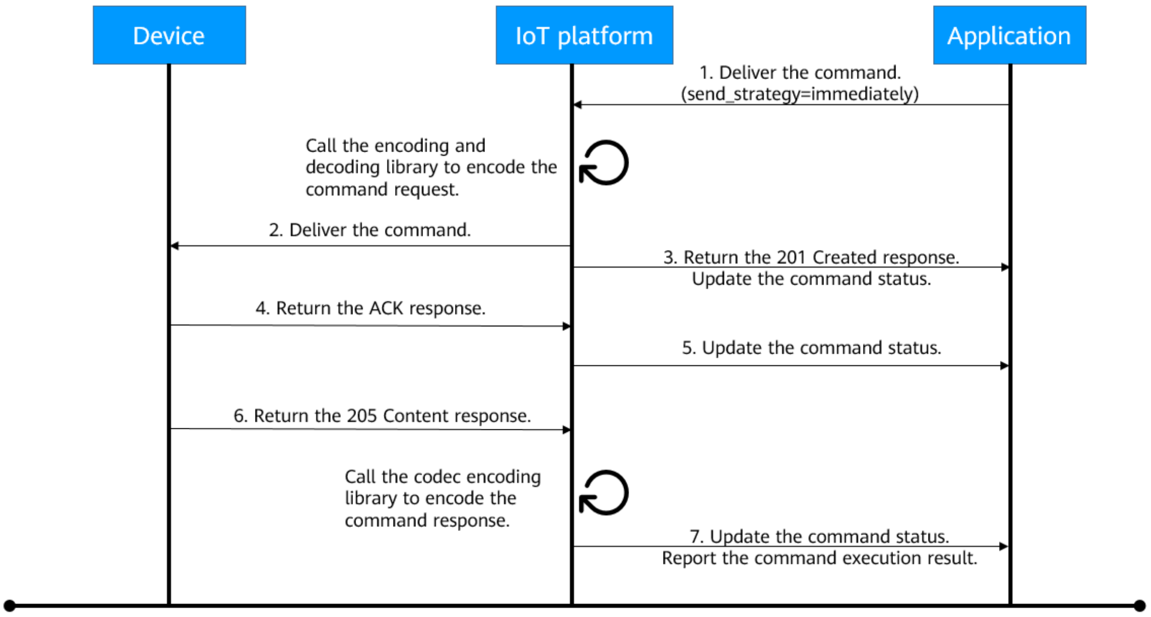

Device
IoT platform
Application
1. Deliver the command. (send_strategy=immediately)
Call the encoding and decoding library to encode the command request.
2. Deliver the command.
3. Return the 201 Created response. Update the command status.
4. Return the ACK response.
5. Update the command status.
6. Return the 205 Content response.
Call the codec encoding library to encode the command response.
7. Update the command status. Report the command execution result.

**Figure 10-7 Immediate delivery of LwM2M/CoAP device commands (see the Appendix for a reference image)**

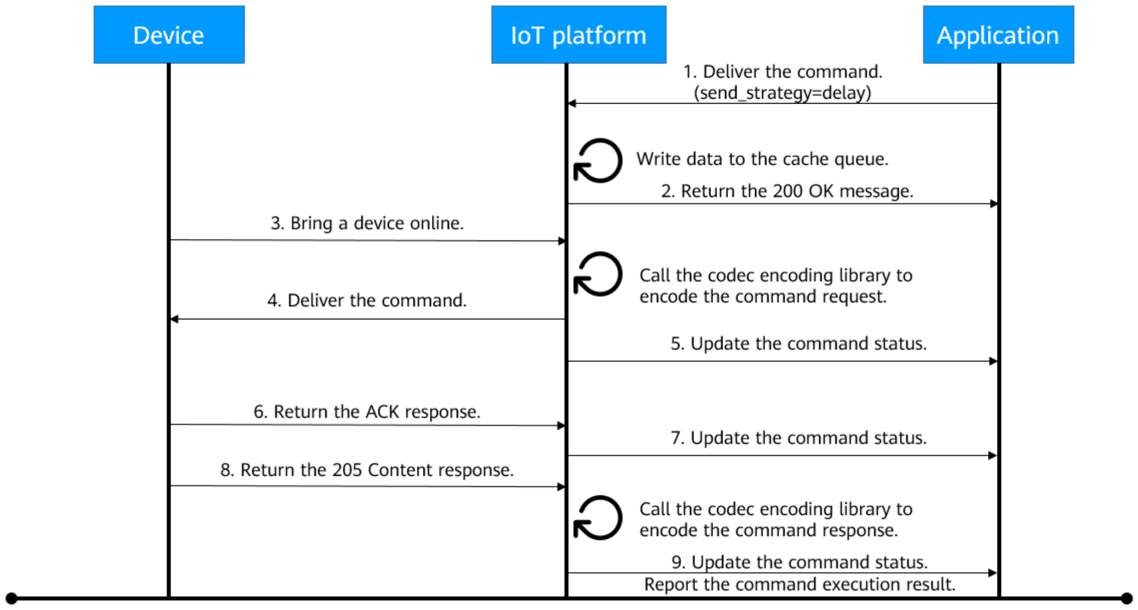

Device
IoT platform
Application
1. Deliver the command. (send_strategy=delay)
Write data to the cache queue.
2. Return the 200 OK message.
3. Bring a device online.
Call the codec encoding library to encode the command request.
4. Deliver the command.
5. Update the command status.
6. Return the ACK response.
7. Update the command status.
8. Return the 205 Content response.
Call the codec encoding library to encode the command response.
9. Update the command status. Report the command execution result.

**Figure 10-8 Delayed delivery of LwM2M/CoAP device commands (see the Appendix for a reference image)**

An application delivers a command to the IoT platform. The IoT platform finds the corresponding codec based on the product information, encodes the command request, and delivers the command to the device. At the same time, the IoT platform sends a command status to the application. When the device responds to the command, the IoT platform changes the command status to Successful or Failed.

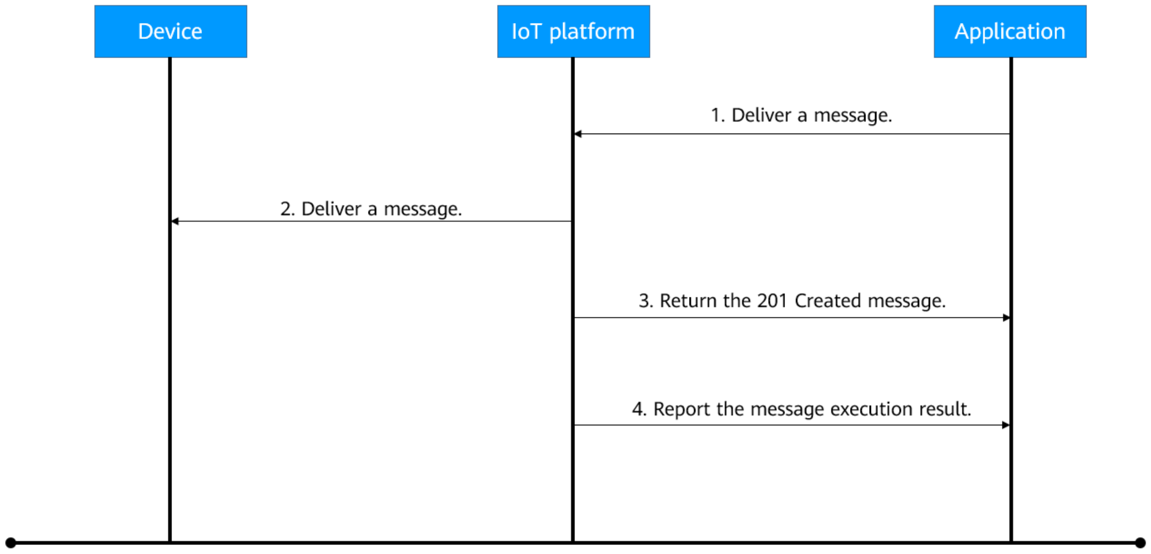

Device
IoT platform
Application
1. Deliver a message.
2. Deliver a message.
3. Return the 201 Created message.
4. Report the message execution result.

**Figure 10-9 Immediate delivery of MQTT device commands (see the Appendix for a reference image)**

Codec is not required and device commands are transparently transmitted. An application delivers a command to the IoT platform. The IoT platform delivers the command to the device and returns the execution result.

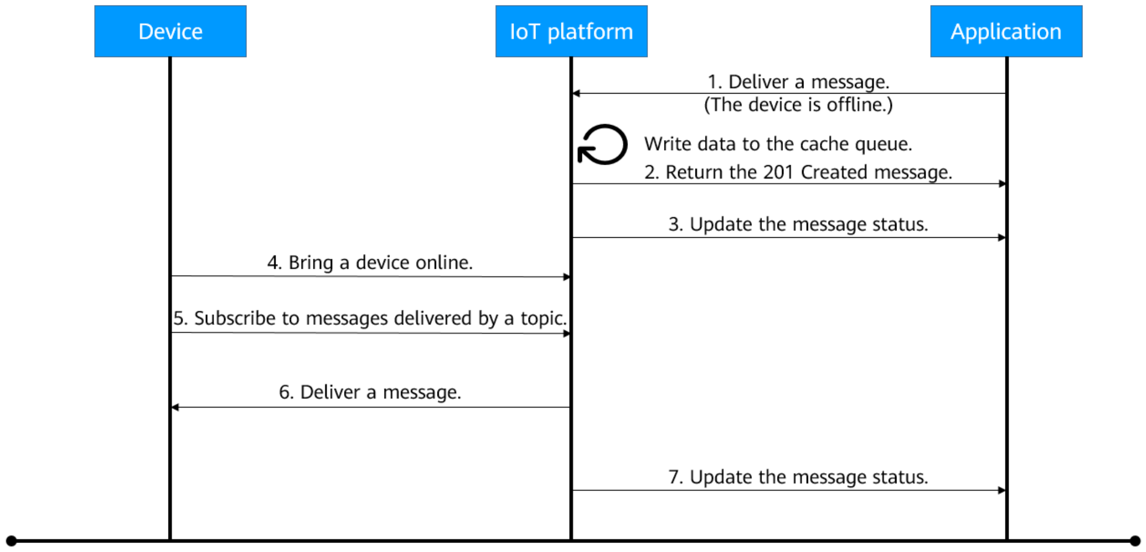
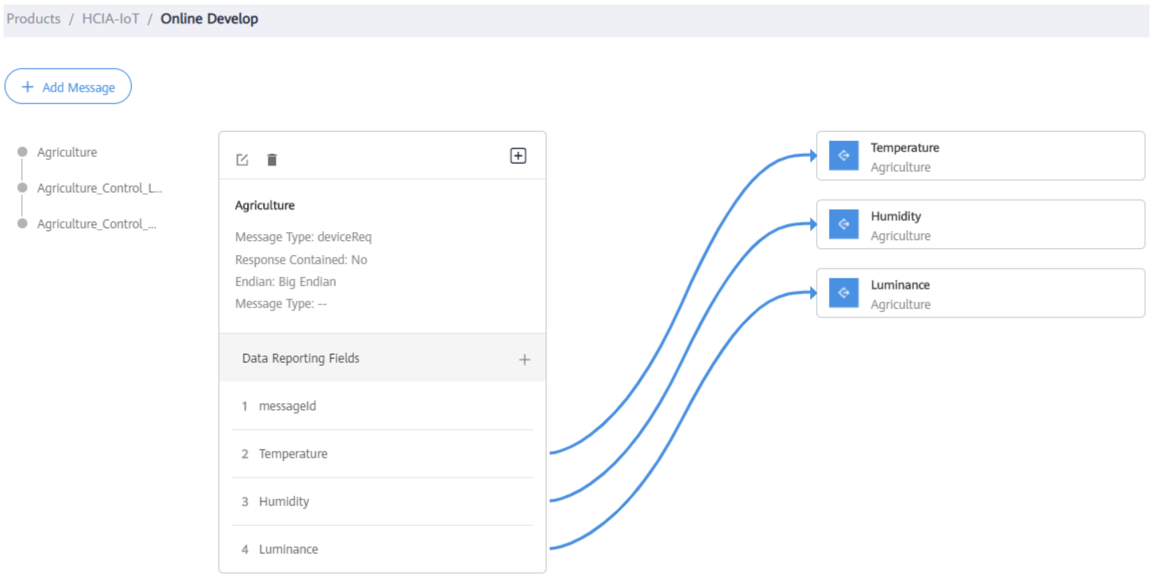

Device
IoT platform
Application
1. Deliver a message.
(The device is offline.)
Write data to the cache queue.
2. Return the 201 Created message.
3. Update the message status.
4. Bring a device online.
5. Subscribe to messages delivered by a topic.
6. Deliver a message.
7. Update the message status.

**Figure 10-10 Delayed delivery of MQTT device commands (see the Appendix for a reference image)**

An application delivers a command, and the IoT platform writes data to the cache queue and updates the message status when the device is offline. When the device goes online and subscribes to messages delivered by a topic, the platform delivers the message and updates the message status.

Codec example



Agriculture
Message Type: deviceReq
Response Contained: No
Endian: Big Endian
Message Type: --

Data Reporting Fields
1 messageId
2 Temperature
3 Humidity
4 Luminance

**Figure 10-11 Codec example (see the Appendix for a reference image)**

## 10.3 Development on the Application Side

The IoT platform provides APIs to make application development easier and more efficient. You can call these open APIs to quickly integrate platform functions, such as product, device, subscription, and rule management, as well as command delivery.

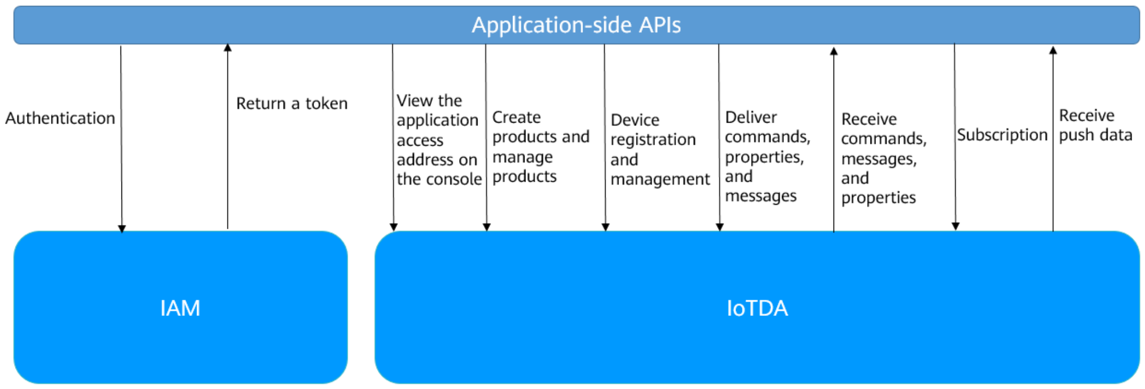

Application-side APIs
Authentication
Return a token
View the application access address on the console
Create products and manage products
Device registration and management
Deliver commands, properties, and messages
Receive commands, messages, and properties
Subscription
Receive push data
IAM
IoTDA

**Figure 10-12 Development on the application side**

Applications must be authenticated by the IAM service and obtain tokens to integrate functions such as product and device management.

## 10.3.1 Calling APIs

The HUAWEI CLOUD IoT platform provides a variety of RESTful APIs for application developers to quickly develop IoT applications based on the capabilities provided by the platform.

### 10.3.1.1 Making an API Request

A request URI is in the following format:

{URI-scheme} :// {Endpoint} / {resource-path} ? {query-string}

Although a request URI is included in the request header, most programming languages or frameworks require the request URI to be transmitted separately.

**URI-scheme**: protocol used to transmit the request. All APIs use HTTPS.

**Endpoint**: domain name or IP address of the server bearing the REST service endpoint. Obtain the value from Regions and Endpoints. For example, the endpoint of the IoT platform in **CN North-Beijing4** is **iotda.cn-north-4.myhuaweicloud.com**.

**resource-path**: access path of an API for performing a specific operation. Obtain the path from the URI of an API. For example, **resource-path** of the API for querying products is **/v5/iot/{project_id}/products/{product_id}**.

**query-string**: query parameter, which is optional. Ensure that a question mark (?) is included before each query parameter that is in the format of "Parameter name=Parameter value". For example, **? limit=10** indicates that a maximum of 10 data records will be displayed.

For example, to obtain information about a specific product from the IoT platform in **CNNorth-Beijing4**, combine the endpoint (**iotda.cn-north-4.myhuaweicloud.com**) of the **CN North-Beijing4** region and **resource-path** (**/v5/iot/{project_id}/products/{product_id}**) in the URI of the API for Querying a Product. The following is an example:

```
https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
```

Request Methods

The HTTP protocol defines the following request methods that can be used to send a request to the server:

**GET**: requests the server to return specific resources.

**PUT**: requests the server to update specific resources.

**POST**: requests the server to add resources or perform special operations.

**DELETE**: requests the server to delete specific resources, for example, objects.

**HEAD**: same as GET except that the server must return only the response header.

**PATCH**: requests the server to update partial content of a specific resource. If the resource does not exist, the PATCH method creates a resource.

For example, in the URI of the API for Querying a Product, the request method is GET. The request is as follows:

```
GET https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
```

Request Message Header

You can add header fields, for example, fields required by a specific URI or HTTP method, to a request header. For example, to request authentication information, add Content-Type, which specifies the request body type.

Common request header fields are as follows:

**Content-Type**: request body type or format. This field is mandatory and its default value is **application/json**. Other values of this field will be provided for specific APIs if any.

**X-Auth-Token**: user token. This field is mandatory when token authentication is used. You can call the API for Obtaining a User Token Through Password Authentication to obtain the value. In the response message header returned by the API, **X-Subject-Token** is the desired user token.

The API for Querying a Product requires authentication. Therefore, the **Content-Type** and **X-Auth-Token** fields need to be added to the header of the request for calling the API. An example of such a request is as follows:

```
GET https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
```

```
Content-Type: application/json
X-Auth-Token: ******
```

Request Message Body

A request body is generally sent in a structured format. It corresponds to **Content-Type** in the request header and transfers data other than the request header. If the request body supports Chinese characters, the Chinese characters must be encoded in UTF-8 mode.

The message body in API requests varies according to APIs. Certain APIs, such as the GET and DELETE APIs, do not require a message body.

For the API for Creating a Product, you can obtain the required request parameters and parameter description from the API request. The following is an example request with the body included. Replace the italic fields in bold with the actual values. For example, ***name*** indicates the product name, ***device_type*** indicates the device type, and ***protocol_type*** indicates the protocol type used by the device.

```
POST https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/abab***cdcd/products
Content-Type: application/json
X-Auth-Token: ********

{
   "name" : "Thermometer",
   "device_type" : "Thermometer",
   "protocol_type" : "MQTT",
   "data_format" : "binary",
   "manufacturer_name" : "ABC",
   "industry" : "smartCity",
   "description" : "this is a thermometer produced by Huawei",
   "service_capabilities" : [ {
      "service_type" : "temperature",
      "service_id" : "temperature",
      "description" : "temperature",
      "properties" : [ {
         "unit" : "centigrade",
         "min" : "1",
         "method" : "R",
         "max" : "100",
         "data_type" : "decimal",
         "description" : "force",
         "step" : 0.1,
         "enum_list" : [ "string" ],
         "required" : true,
         "property_name" : "temperature",
         "max_length" : 100
      } ],
      "commands" : [ {
         "command_name" : "reboot",
         "responses" : [ {
            "response_name" : "ACK",
            "paras" : [ {
               "unit" : "km/h",
               "min" : "1",
               "max" : "100",
               "para_name" : "force",
               "data_type" : "string",
               "description" : "force",
               "step" : 0.1,
               "enum_list" : [ "string" ],
               "required" : false,
               "max_length" : 100
            } ]
         } ],
         "paras" : [ {
            "unit" : "km/h",
            "min" : "1",
            "max" : "100",
            "para_name" : "force",
            "data_type" : "string",
            "description" : "force",
            "step" : 0.1,
            "enum_list" : [ "string" ],
            "required" : false,
            "max_length" : 100
         } ]
      } ],
      "option" : "Mandatory"
   } ],
   "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
```

```
}
```

### 10.3.1.2 Authentication

Requests for calling an API can be authenticated in either of the following methods:

Using tokens: General requests are authenticated using tokens.

Using Access Key/Secret Access Key (AK/SK): Requests are authenticated by encrypting the request body using an AK/SK.

Token authentication

A token is a character string generated by the server and is used for a client to send a request. After the first login, the server generates a token and returns the token to the client. In subsequent requests, the client needs to carry the token, but not the username and password. The validity period of a token is 24 hours, which starts from the time when the client obtains the token. If the same token needs to be used for authentication, it is recommended that the token be cached to avoid frequent calling. Before the token expires, you must obtain a new token. Otherwise, the authentication on the server will fail after the token expires.

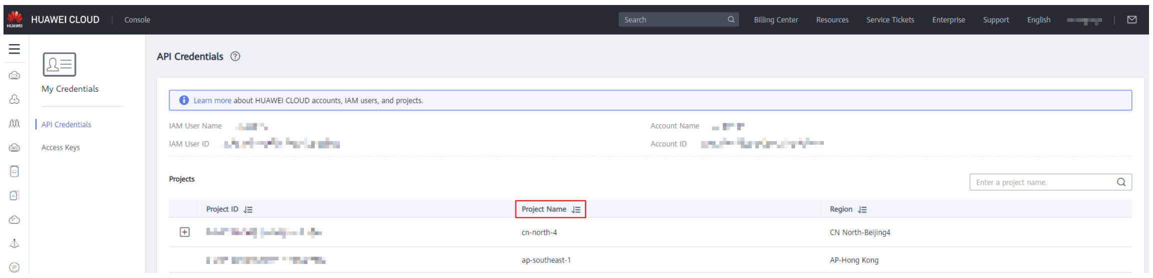
If you obtain the token for multiple times, the latest token is used. The previous token will be overwritten and become invalid.

A token specifies temporary permissions in a computer system. During API authentication using a token, the token is added to requests to get permissions for calling the API.

To obtain a token, call the API for Obtaining a User Token Through Password Authentication. The following is an example:

```
POST https://iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
        "auth": {
                "identity": {
                        "methods": [
                                "password"
                        ],
                        "password": {
                                "user": {
                                        "name": "username",
                                        "password": "********",
                                        "domain": {
                                                "name": "domainname"
                                        }
                                }
                        }
                },
                "scope": {
                        "project": {
                                "name": "xxxxxxxx"
                        }
                }
        }
}
```

Note: **username** indicates the IAM user name, **password** indicates the password for logging in to HUAWEI CLOUD, **domainname** indicates the account name, and **projectname** indicates the project name.



API Credentials

Learn more about HUAWEI CLOUD accounts, IAM users, and projects.

IAM User Name [redacted]
IAM User ID [redacted]
Account Name [redacted]
Account ID [redacted]

Projects

Enter a project name.

| | Project ID | Project Name | Region |
|---|---|---|---|
| ⊞ | [redacted] | cn-north-4 | CN North-Beijing4 |
| | [redacted] | ap-southeast-1 | AP-Hong Kong |

**Figure 10-13 API credential (see the Appendix for a reference image)**

In the response to the API used to obtain a token, **X-Subject-Token** is the token obtained.

The X-Auth-Token header field must be included to carry the token when calling other APIs. For example, if the token is **ABCDEFJ....**, **X-Auth-Token: ABCDEFJ....** can be added to a request as follows:

```
GET https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

AK/SK-based Authentication

Authentication using AK/SK supports API requests with a body not larger than 12 MB. For API requests with a larger body, authentication using tokens is recommended.

AK/SK is used to sign requests and the signature is then added to the request headers for authentication.

AK: access key ID, the unique ID associated with a secret access key. The AppKey and AppSecret are used together to obtain an encrypted signature for a request.

SK: secret access key used together with an AK to sign requests cryptographically. It identifies a request sender and prevents the request from being tampered with.

In AK/SK-based authentication, you can use AK/SK to sign requests based on the signature algorithm or use a dedicated signature SDK to sign requests.

Unlike the SDKs provided by services, the signature SDK supports only signature.

### 10.3.1.3 Returning a Response

Status Code

After sending a request, you will receive a response, including the status code, response header, and response body.

A status code is a group of digits, ranging from 1xx to 5xx. It indicates the status of a request. For details, see Status Code.

If **201** is returned for calling the API for Creating a Product, the request is successful.

Response Header

Similar to a request, a response also has a header, for example, **Content-Type**.

For the API for Creating a Product, the platform will return a response containing message headers, for example, **Content-Type** and **Date**.

Response Body

The body of a response is often returned in structured format as specified in the **Content-Type** header field. The response body transfers content except the response header.

The following is part of the response body for the API for Creating a Product:

```
{
   "product_id" : "5ba24f5ebbe8f56f5a14f605",
   "name" : "Thermometer",
   "device_type" : "Thermometer",
   "protocol_type" : "LWM2M",
   "data_format" : "binary",
   "manufacturer_name" : "ABC",
   "industry" : "smartCity",
   "description" : "this is a thermometer produced by Huawei",
     ......
}
```

If an error occurs during API calling, an error code and error description will be displayed. The following shows an error response body:

```
{
       "error_msg": "The format of message is error",
       "error_code": "IOTDA.013005"
}
```

In the preceding information, **error_code** is an error code, and **error_msg** describes the error.

## 10.3.2 API

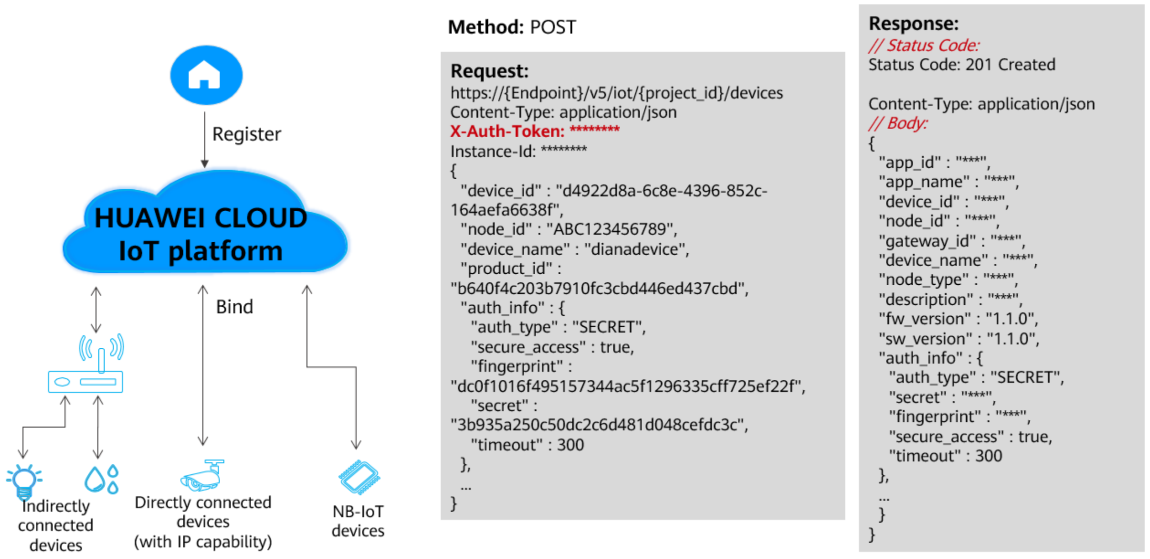
### 10.3.2.1 Creating a Device

This API is used by an application to create a device on the IoT platform. The device can access the IoT platform only after being created.

This API can use the **gateway_id** parameter to create a child device under a parent device. Currently, a maximum of two levels of child devices are supported.

This API also supports initial device configuration. The API reads the product details provided in the **product_id** parameter in the device creation request. If the default value of a product property is defined, the default value is written to the device shadow.

You can also use the **shadow** field in the device creation request to specify the initial configuration for the device. After the initial configuration is specified, the system compares the property values set based on **service_id** and **desired** with the default values of the corresponding properties in the product. If the property values are different, the property values set in the **shadow** field are written to the device shadow.


Register
HUAWEI CLOUD
IoT platform
Bind
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** POST

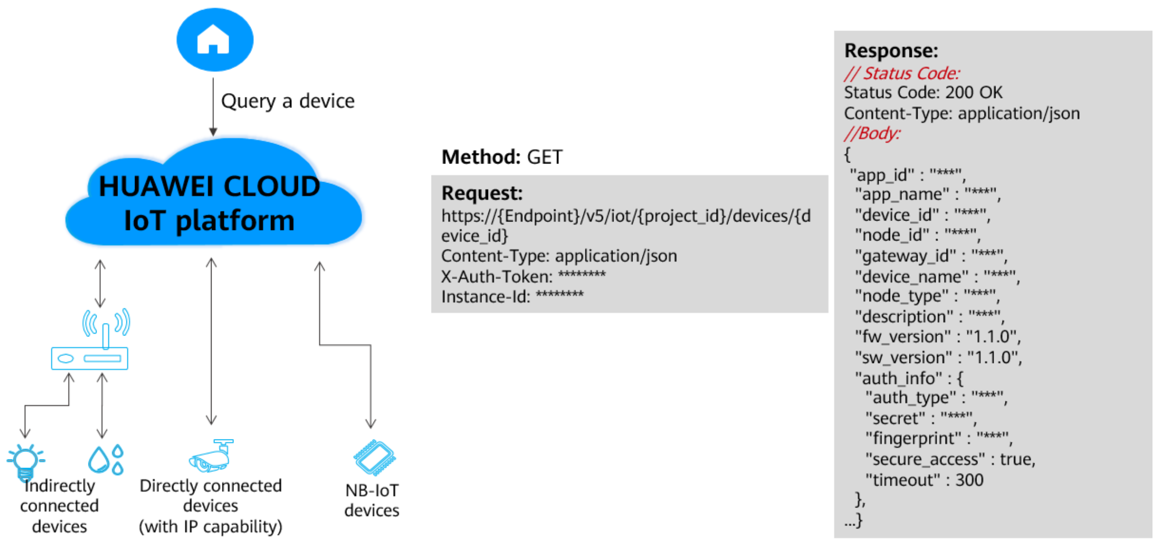
```
Request:
https://{Endpoint}/v5/iot/{project_id}/devices
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
{
  "device_id" : "d4922d8a-6c8e-4396-852c-
164aefa6638f",
  "node_id" : "ABC123456789",
  "device_name" : "dianadevice",
  "product_id" :
"b640f4c203b7910fc3cbd446ed437cbd",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secure_access" : true,
    "fingerprint" :
"dc0f1016f495157344ac5f1296335cff725ef22f",
    "secret" :
"3b935a250c50dc2c6d481d048cefdc3c",
    "timeout" : 300
  },
  ...
}
```

```
Response:
// Status Code:
Status Code: 201 Created

Content-Type: application/json
// Body:
{
  "app_id" : "***",
  "app_name" : "***",
  "device_id" : "***",
  "node_id" : "***",
  "gateway_id" : "***",
  "device_name" : "***",
  "node_type" : "***",
  "description" : "***",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "***",
    "fingerprint" : "***",
    "secure_access" : true,
    "timeout" : 300
  },
  ...
  }
}
```

**Figure 10-14 API for creating a device (see the Appendix for a reference image)**

### 10.3.2.2 Querying a Device

This API is used by an application to query the details of a device on the IoT platform.


Query a device
HUAWEI CLOUD
IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** GET

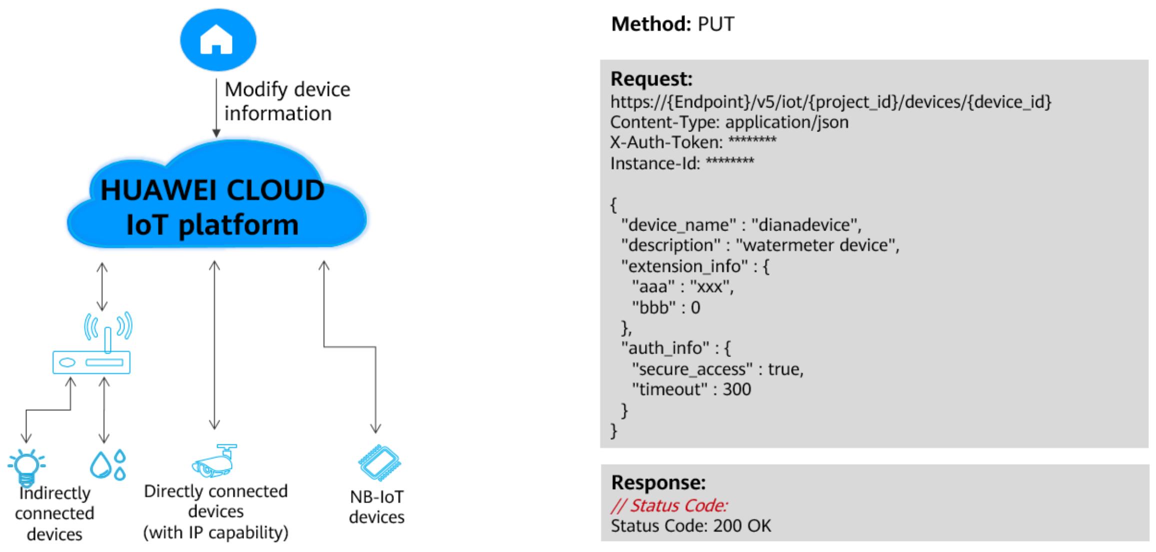
```
Request:
https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

```
Response:
// Status Code:
Status Code: 200 OK
Content-Type: application/json
//Body:
{
 "app_id" : "***",
  "app_name" : "***",
  "device_id" : "***",
  "node_id" : "***",
  "gateway_id" : "***",
  "device_name" : "***",
  "node_type" : "***",
  "description" : "***",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "***",
    "secret" : "***",
    "fingerprint" : "***",
    "secure_access" : true,
    "timeout" : 300
  },
...}
```

**Figure 10-15 API for querying a device (see the Appendix for a reference image)**

### 10.3.2.3 Modifying Device Information

This API is used by an application to modify the basic information of a device on the IoT platform.


Modify device information
HUAWEI CLOUD
IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** PUT

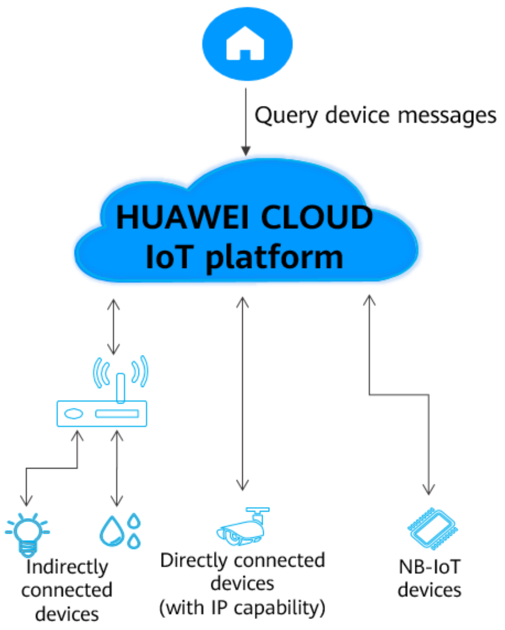
```
Request:
https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********

{
  "device_name" : "dianadevice",
  "description" : "watermeter device",
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  },
  "auth_info" : {
    "secure_access" : true,
    "timeout" : 300
  }
}
```

```
Response:
// Status Code:
Status Code: 200 OK
```

**Figure 10-16 API for modifying device information (see the Appendix for a reference image)**

### 10.3.2.4 Querying Device Messages

This API is used by an application to query messages of a device. By default, the platform stores a maximum of 20 messages for each device. If the number of messages exceeds 20, the earliest messages will be overwritten by subsequent messages.


Query device messages
HUAWEI CLOUD IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** GET

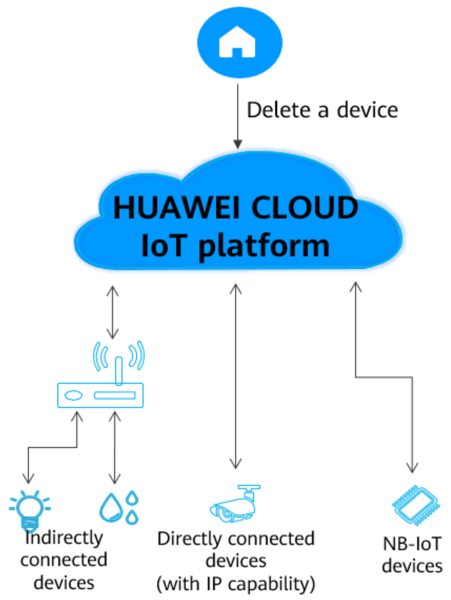
**Request:**
```
https://{Endpoint}/v5/iot/{project_id}/devices/{device_
id}/messages
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

**Response:**
```
// Status Code:
Status Code: 200 OK
Content-Type: application/json

{
  "device_id" : "d4922d8a-6c8e-4396-852c-
164aefa6638f",
  "messages" : [ {
    "message_id" : "b1224afb-e9f0-4916-8220-
b6bab568e888",
    "name" : "message_name",
    "message" : "string",
    "topic" : "string",
    "status" : "PENDING",
    "created_time" : "20151212T121212Z",
    "finished_time" : "20151212T121212Z"
  } ]
}
```

**Figure 10-17 API for querying device messages (see the Appendix for a reference image)**

### 10.3.2.5 Deleting a Device

This API is used by an application to delete a device from the platform. If the device is connected to an indirectly connected device, you must delete the indirectly connected device first.


Delete a device
HUAWEI CLOUD IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** DELETE

**Request:**
```
https://{Endpoint}/v5/iot/{project_id}/devices/
{device_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

**Response:**
```
Status Code: 204 No Content
```

**Figure 10-18 API for deleting a device (see the Appendix for a reference image)**

To view more APIs, scan the QR code in Figure 10-19 to view the API list.

**Figure 10-19 API document**

## 10.4 Development on the Device Side

You can use IoT Device SDKs to quickly connect devices to the IoT platform. After being integrated with the IoT Device SDK, devices that support the TCP/IP protocol stack can directly communicate with the IoT platform. Devices that do not support the TCP/IP protocol stack (such as Bluetooth and Zigbee devices) need to use the gateway to forward device data to the IoT platform. In this case, the gateway must be integrated with the IoT Device SDK in advance. MQTT is the first choice for IoT communications. The HUAWEI CLOUD IoT platform allows devices to connect to the IoT platform using MQTT. You are advised to use the IoT Device SDK to connect devices to the platform over MQTTS.

Development mode on the device side:

- Certified MCU development

  The IoT Device SDK Tiny has been pre-integrated into the main control unit (MCU) and can call methods to connect to the platform. This development mode is applicable to the scenario where devices need to be quickly put into commercial use, with low R&D costs. Devices are connected to the platform directly, without using gateways.

- Certificated module development

  The IoT Device SDK Tiny has been pre-integrated into the module and can invoke AT commands to connect to the platform. This development mode is applicable to the scenario where there are few MCU resources. Devices are connected to the platform directly, without using gateways.

- LiteOS development

  Devices run LiteOS that manages MCU resources. In addition, LiteOS has a built-in IoT Device SDK Tiny that can call functions to connect to the platform. This development mode shortens the device development duration and reduces the development difficulty. It applies to directly connected devices that do not have OSs and do not need to manage child devices.

- Common development

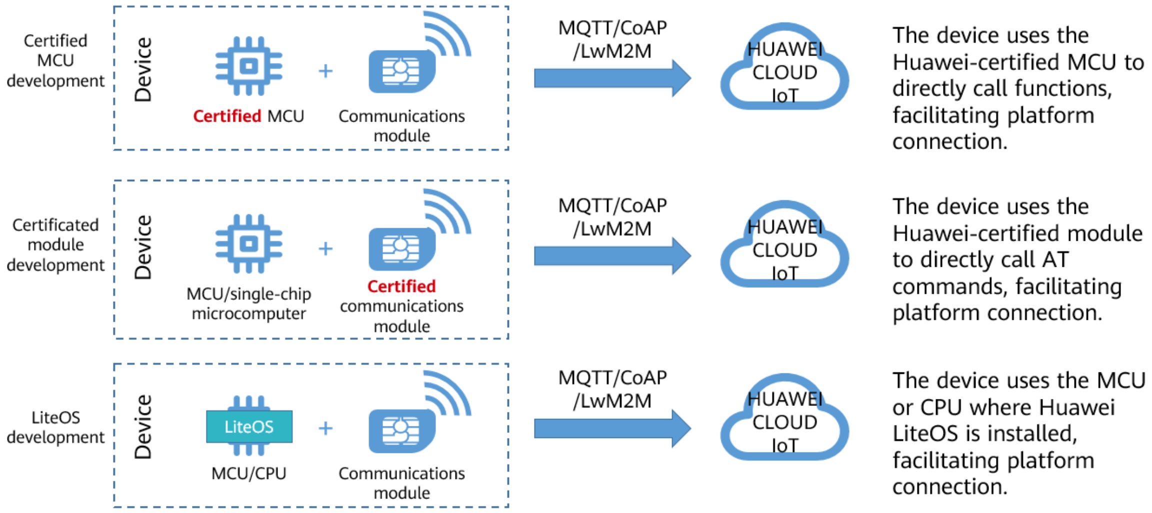
  The IoT Device SDK Tiny is integrated into the MCU and calls the SDK functions to connect to the platform. This type of call is more convenient than API access. This development mode is applicable to the scenario where there is sufficient time for devices to put into commercial use, and the flash and RAM resources of the MCU meet the conditions for integrating the IoT Device SDK Tiny.

- OpenCPU module development

  The MCU capability in the common module is used, and device applications are compiled and run on the OpenCPU. This development mode is applicable to devices that have high security requirements, are small in size, and need to be quickly put into commercial use.

- Gateway development

  The IoT Device SDK is pre-integrated into the CPU or MPU and can call functions to connect to the platform. This development mode is applicable to gateways that manage child devices.


Certified MCU development
Device
Certified MCU
+
Communications module
MQTT/CoAP /LwM2M
HUAWEI CLOUD IoT
The device uses the Huawei-certified MCU to directly call functions, facilitating platform connection.
Certificated module development
Device
MCU/single-chip microcomputer
+
Certified communications module
MQTT/CoAP /LwM2M
HUAWEI CLOUD IoT
The device uses the Huawei-certified module to directly call AT commands, facilitating platform connection.
LiteOS development
Device
LiteOS
MCU/CPU
+
Communications module
MQTT/CoAP /LwM2M
HUAWEI CLOUD IoT
The device uses the MCU or CPU where Huawei LiteOS is installed, facilitating platform connection.

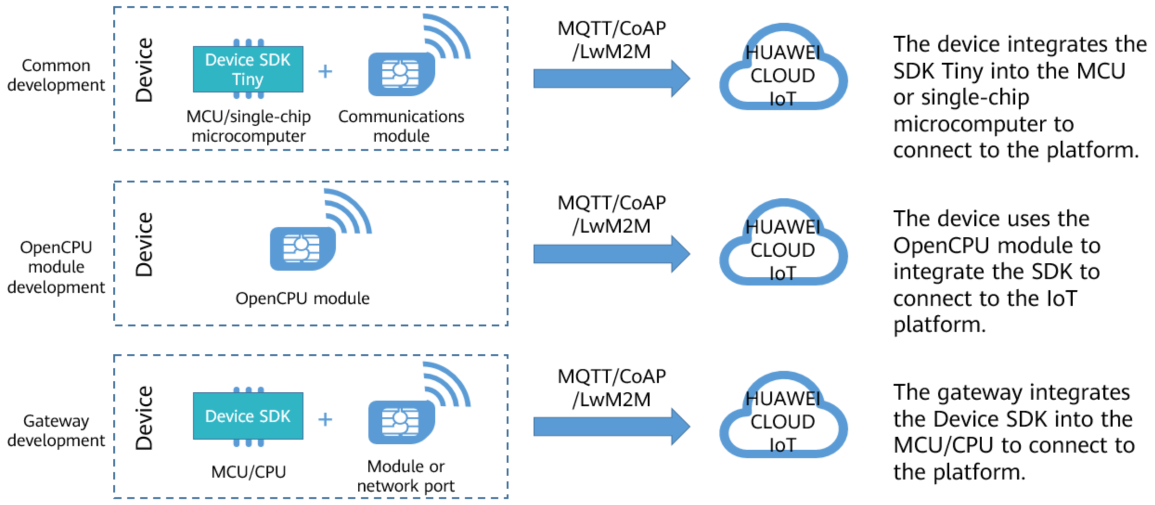

Common development
Device
Device SDK Tiny
+
MCU/single-chip microcomputer
Communications module
MQTT/CoAP /LwM2M
HUAWEI CLOUD IoT
The device integrates the SDK Tiny into the MCU or single-chip microcomputer to connect to the platform.
OpenCPU module development
Device
OpenCPU module
MQTT/CoAP /LwM2M
HUAWEI CLOUD IoT
The device uses the OpenCPU module to integrate the SDK to connect to the IoT platform.
Gateway development
Device
Device SDK
+
MCU/CPU
Module or network port
MQTT/CoAP /LwM2M
HUAWEI CLOUD IoT
The gateway integrates the Device SDK into the MCU/CPU to connect to the platform.

**Figure 10-20 Development on the device side**

Using IoT Device SDKs for access

You can use IoT Device SDKs to quickly connect devices to the IoT platform. After being integrated with the IoT Device SDK, devices that support the TCP/IP protocol stack can directly communicate with the IoT platform. Devices that do not support the TCP/IP protocol stack, such as Bluetooth and Zigbee devices, need to use a gateway integrated with the IoT Device SDK to communicate with the platform.

1. Create a product on the IoTDA console or by calling the API **Creating a Product**.
2. Register the device on the IoTDA console or by calling the API **Creating a Device**.
3. Implement the functions demonstrated in Figure 10-21, including reporting messages/properties, receiving commands/properties/messages, OTA upgrades, topic customization, and generic-protocol access.

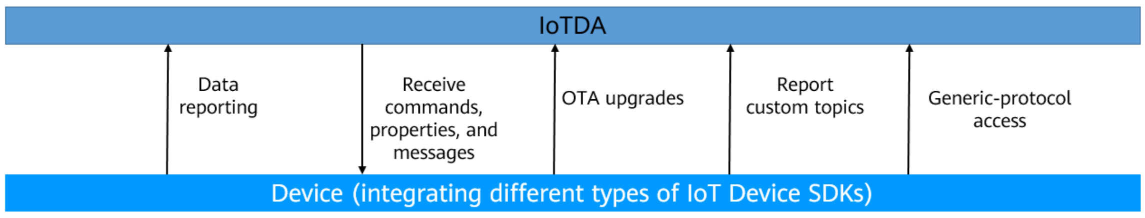

IoTDA
Data reporting
Receive commands, properties, and messages
OTA upgrades
Report custom topics
Generic-protocol access
Device (integrating different types of IoT Device SDKs)

**Figure 10-21 Using IoT Device SDKs for access**

Using Huawei-certified modules for access

Certified modules are pre-integrated with the IoT Device SDK Tiny. They have passed Huawei tests, and comply with Huawei's AT command specifications. The following benefits are available for Huawei-certified modules:

Device manufacturers do not need to worry about how to connect to the HUAWEI CLOUD IoT platform on the MCU (for example, how to set the secret encryption algorithm and clientID composition mode during MQTT connection setup). To connect their devices to the platform, they only need to invoke AT commands. This accelerates device interconnection and commissioning.

The MCU does not need to integrate the MQTT protocol stack or IoT Device SDK Tiny, greatly reducing MCU resource consumption.

Huawei releases certified modules on HUAWEI CLOUD Marketplace so that device manufacturers and service providers can purchase these certified modules to quickly connect to HUAWEI CLOUD IoT.

Figure 10-22 shows how a certified module is used to connect a device to the platform.

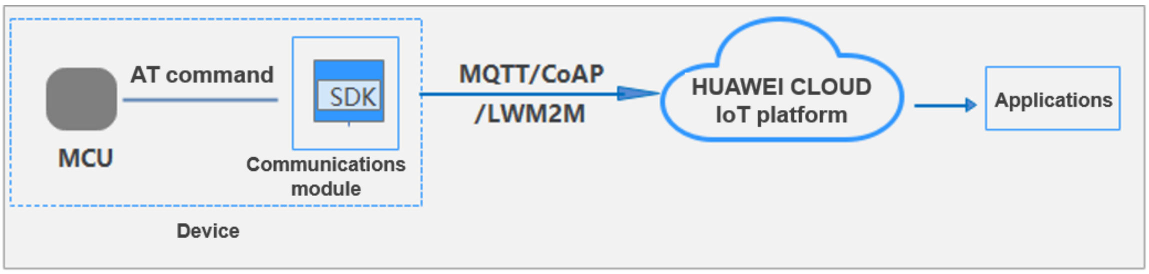


**Figure 10-22 Using certified modules to develop devices**

Device authentication

The platform authenticates a device when the device attempts to access the platform. The authentication process depends on the access method.

**Table 10-3 Device authentication**

| Access Type | Description |
|---|---|
| Device connected using LwM2M over CoAP | Call the API **Creating a Device** or use the IoTDA console to register a device with the platform, and set the node ID (for example, the IMEI) as the verification code. The device can use the node ID to get authenticated and connect to the platform.<br>When Datagram Transport Layer Security (DTLS) or DTLS+ is used, the transmission channel between the device and platform is encrypted by using a PSK. |
| Device using native MQTT or MQTTS | Call the API **Creating a Device** or use the IoTDA console to register a device with the platform, and hardcode the device ID and secret returned by the platform into the device. A CA certificate is preset on MQTTS devices, but not MQTT devices. The device uses the device ID and secret to get authenticated and connect to the platform. |

**Authentication for devices using LwM2M over CoAP**

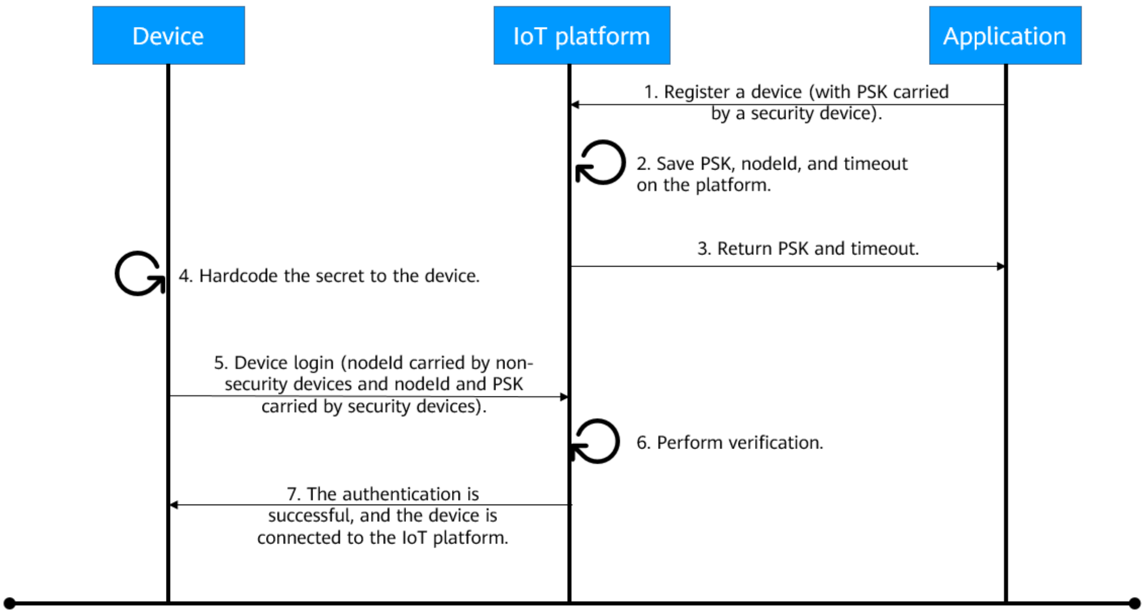

Device
IoT platform
Application
1. Register a device (with PSK carried by a security device).
2. Save PSK, nodeId, and timeout on the platform.
3. Return PSK and timeout.
4. Hardcode the secret to the device.
5. Device login (nodeId carried by non-security devices and nodeId and PSK carried by security devices).
6. Perform verification.
7. The authentication is successful, and the device is connected to the IoT platform.

**Figure 10-23 Authentication for devices using LwM2M over CoAP (see the Appendix for a reference image)**

1. An application calls the API **Creating a Device** to register a device. Alternatively, a user uses the IoTDA console to register a device.

2-3. The platform allocates a globally unique PSK to the device and returns **timeout**.

4. The user hardcodes PSK into the device hardware, software, or firmware.

5. The user powers on the device. The device sends a connection request carrying the node ID (for example, IMEI) and PSK.

6-7. If the authentication is successful, the platform returns a success message, and the device is connected to the platform.

**Authentication for devices using native MQTT or MQTTS**

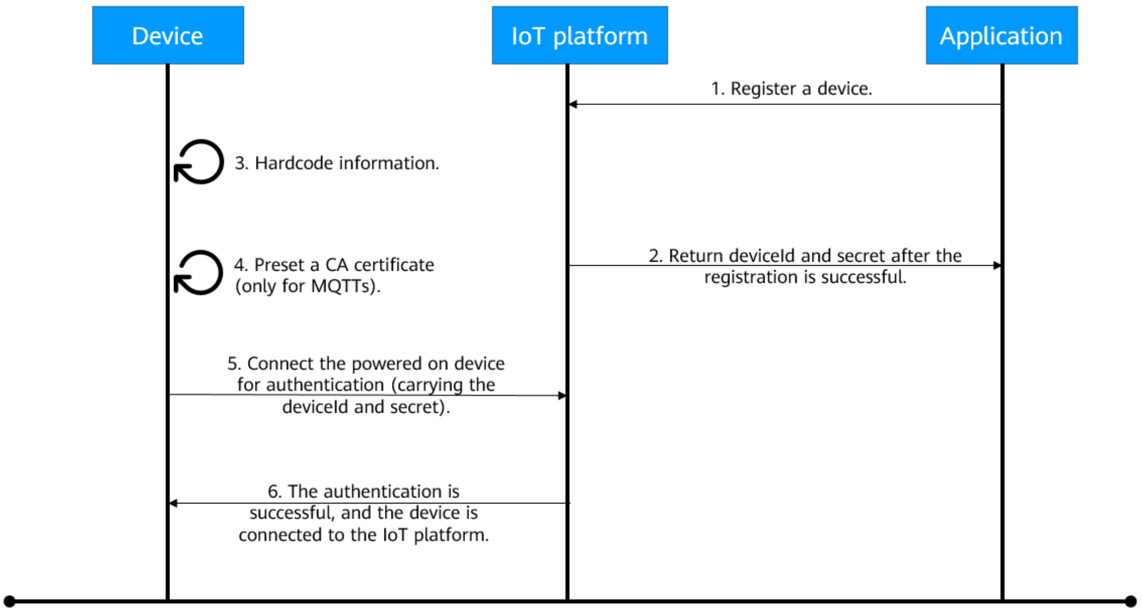

Device
IoT platform
Application
1. Register a device.
3. Hardcode information.
2. Return deviceId and secret after the registration is successful.
4. Preset a CA certificate (only for MQTTs).
5. Connect the powered on device for authentication (carrying the deviceId and secret).
6. The authentication is successful, and the device is connected to the IoT platform.

**Figure 10-24 Authentication for devices using native MQTT or MQTTS (see the Appendix for a reference image)**

1. An application calls the API **Creating a Device** to register a device. Alternatively, a user uses the IoTDA console to register a device.
2. The platform allocates a globally unique device ID and secret to the device.
3. The user hardcodes the device ID and secret to the device hardware, software, or firmware.
4. (Optional) The user presets the CA certificate on the device. This step is required only for devices connected using MQTTS.
5. The user powers on the device. The device sends a connection request carrying the device ID and secret.
6. If the authentication is successful, the platform returns a success message, and the device is connected to the platform.

# 11 IoT OS

From this chapter, you will start to acquire knowledge about the sensing layer. In this chapter, readers will understand what an operating system (OS) is, the OS development process, and the challenges faced by the IoT OS. In addition, this chapter describes the basic concepts and solutions of Huawei's IoT OS, known as LiteOS.

## 11.1 Preliminary Study of the OS

### 11.1.1 Concept

Most people have experienced OSs such as Windows and Linux used on computers and iOS and Android on mobile phones. OSs play a very important role in human-computer interaction. As such, it is important to understand what an operating system is.

This section describes the concept of the OS from the perspective of the modern computer system. First of all, the microcomputer system can be divided into two parts: the hardware system and the software system. In the hardware system, there are two main types: the host and peripherals. The host contains devices such as the CPU and storage device. Peripherals are external devices, such as the mouse, keyboard, disk, and monitor.

The software system is also divided into two parts: system software and application software. The system software includes the OS and other required tools such as development tools and diagnosis programs. The application software includes the software that is commonly used in everyday life, such as Word and PowerPoint.

Modern computer systems consist of one or more processors, main memory, disks, printers, keyboards, mouse devices, monitors, network interfaces, and other input/output devices. A modern computer system is considered a complex system. If application programmers had to master every detail of the system, it would not be possible for them to write code. Moreover, managing and optimizing the use of these components is a daunting task. So, a layer of software called an OS is installed on the computer, which is tasked with providing a better, simpler, and clearer computer model for user programs and managing all the devices just mentioned.

Put simply, the OS is a program that manages and controls all software and hardware resources. Essentially, this program is the basis and core for running other programs on your computer. Only with an OS can the user run other programs on the computer, control the hardware operated by the user (such as the keyboard and mouse), as well as the CPU and memory in the computer host.

The difference between an OS and a normal application is not just their positions. Another major difference is that the OS is larger, more complex, and longer-lived. The source code of the Linux or Windows system has 5 million lines or more. This makes it difficult to write code for the OS, which is why the OS has a long service life. Once the code of the OS is written, the owner of the OS certainly does not want to start again from scratch. Instead, the OS continuously evolves over a long period of time.

### 11.1.2 History

OSs have been around for decades. This section will briefly introduce the history of the OS.

The first generation of OSs began in the 1940s, when engineers from the same group designed, built, transformed, operated, and maintained one machine. At that time, all programming was written in pure machine language, with thousands of cables needing to be connected to the plug-in board to control the basic functions of the machine. There was no programming language, not even assembly language, and no one had ever heard of an OS, because all operations were performed by humans. In general, a programmer used a machine by booking a time on the machine's timetable on the wall, before going to the computer room and connecting the plug-in board to the computer. For the next few hours, the programmer has nothing to do but hope that the more than 20,000 vacuum tubes in operation will not burn out. At that time, all calculations were actually simple mathematical operations, such as making sines, cosines, and logarithmic tables, or calculating cannonball trajectories.

By the early 1950s, significant improvements had been made. For example, punched cards had emerged, allowing programs to be written on the cards and then read by the computer without the plug-in board. But the rest of the process remained the same. In the late 1950s, the invention of transistors dramatically changed the landscape. Manufacturers could then mass produce and sell computers to users, who expected them to run for long periods of time and do all kinds of useful work.

The second-generation OS, also known as the batch-processing OS, was very expensive at the time, making it vital to find ways to reduce the waste of machine time. To this end, the programmer collected the jobs in the input room, read them on a tape using a relatively inexpensive computer, and then did the real calculations with a more expensive computer. After the bulk of jobs were collected, the program in the card was written onto the tape and mounted to the tape drive. The OS then read the first job from the tape and ran it, and its output was written onto the second tape without printing. At the end of each job, the OS automatically read the next job from the tape and ran it. After a batch of jobs were completed, the operator took out the input and output tapes, replaced the input tapes with the next batch of jobs, and then took the output tapes to a relatively cheap computer for printing.

Then, the multi-channel batch processing OS introduced by the use of integrated circuits worked on the principle of multi-channel program design. The principle was to allow multiple programs to be entered and run in the memory at the same time to reduce the CPU idle time. For programmers, however, they missed using the first-generation OS because, at that time, they could exclusively occupy a machine for several hours, allowing them to get real-time feedback on program debugging. However, the second- and third-generation OSs required them to wait for a long time. This situation led to the emergence of time-sharing systems.

A time-sharing system was actually a variant of multiple programs. Each user had an online terminal, which divided the running time of the processor into short time slices and allocated the processor to each online job in the order based on the time slices. If a job could not complete its calculation within the allocated time slice, the job was suspended and the processor was taken over by other jobs. The job would then continue to be run after the next round. Because of the high-speed computing and job rotation, it seemed to each user that they had exclusive use of a computer.

Although the multi-channel batch processing system and the time-sharing system can obtain satisfactory resource utilization and system response time, they cannot meet the requirements for real-time control and information processing. To this end, a real-time system was generated. That is, the system can quickly respond to random external events and process the events within a strict time frame. This means certain urgent tasks can be completed within a certain time limit without waiting in a time slice queue. There are two types of time limits. If an action must occur at a specified time (or within a specified time range), it is called a hard real-time system. An example of this is automatic flight control systems for aircraft, which must provide absolute assurance that a particular action will be completed within a specified time frame. The other type is known as soft real-time systems, such as flight booking systems and bank management systems, which can tolerate occasional time violations that do not cause any permanent damage.

Another key development that began in the mid-1980s was the growth of personal computer networks running network OSs and distributed OSs. On a network OS, a user is aware of the existence of multiple computers, and can log in to a remote machine and copy files from one machine to another. Each computer runs its own local OS and has its own local user.

In essence, there is no difference between a network OS and a single-processor OS. Of course, they need a network interface controller and some underlying software to drive it, as well as programs for remote login and remote file access, but these add-ons do not change the essential structure of the OS.

In contrast, a distributed OS is presented to users as a traditional single-processor OS, despite being made up of multiple processors. Users are not aware of where their programs are running or where their files are stored. These should be handled effectively by the OS itself.

A real distributed OS is more than just the addition of a small piece of code to a standalone OS, as it is essentially different from a centralized system. For example, a distributed system usually allows an application to run on multiple processors at the same time, requiring a more complex processor scheduling algorithm to obtain maximum parallelism optimization.

The communication delay in the network often results in the need for the distributed algorithm to adapt to the environment of incomplete, outdated, and even incorrect information. This is fundamentally different from a standalone system, in which the OS has all the information about the entire system.

### 11.1.3 Development

With the emergence of new technologies, computers have developed from transistors to integrated circuits, large-scale integrated circuits, and micro-computers. In addition, modern society has evolved from the Internet era where computers are used, to the mobile Internet era where mobile computers (mobile phones) are used, and then to the Internet of Things (IoT) era. In the IoT era, IoT devices are extremely diverse, and their architectures and protocols are sophisticated. In this era, building an operating system with unified management is difficult but essential.

So why is it essential to build an IoT operating system? The reason is that in every generation of electronic devices, OSs compete fiercely for prominence. For example, Windows has long been, and still is, the leading OS for personal computers. For mobile devices, Android and iOS take center stage.

It is no different in the IoT era. Currently, all enterprises may be in the development phase. However, there is only room for one or two OSs to lead the entire IoT industry. Although the IoT OS was built for IoT, its applications will extend beyond IoT. But unlike a computer and a mobile phone, an IoT is more like a network that connects all electronic devices together, and can be said to have only an auxiliary role.

So what should we be focusing on in the development of IoT? In fact, we should focus on developing AR and VR. Mark Zuckerberg, the founder of Facebook, once said that "VR is going to be the most social platform". So, we can expect the use of VR and AR devices to increase hugely in the future. But what is this prediction based on? With the emergence of 5G, the network transmission rate is set to greatly increase. This increased rate will make it easier to communicate with our colleagues when we work from home. And with the continuous development of AR and VR technologies, we can communicate face to face across long distances through VR or AR devices. This will revolutionize the way we work and interact with people.

For example, British Airways has developed a service that provides users with an immersive experience of traveling to their destinations before they even board their flights. Another possible application is to help us perform services such as telemedicine. Therefore, AR and VR devices could very well become the new tools for users to interact with computers.

Our goal is to develop new ways for users to interact with machines, such as using AR and VR, in addition to OSs for the IoT. Therefore, it is very important to build such an OS for AR and VR devices.

## 11.2 Challenges to the IoT OS

After understanding the importance and benefits of the IoT operating system, let's take a look at the challenges that we will face when developing such an operating system. This section describes the challenges we faced when developing IoT terminals and the OS.

### 11.2.1 IoT Terminal Development

We are developing an OS running on IoT terminals. However, there are many types of IoT terminals, which all use different chips and architectures. For example, some may use ARM, embedded FPGA, x86, or DSP architectures. When we develop these devices, we need to adapt to different interfaces.

Not only do these terminals use different chips and architectures, they also use different communication protocols. Taking smart home as an example, the devices and communication protocols used are sophisticated. For example, the Bluetooth speakers in our home use the Bluetooth protocol, which enables low power consumption. The protocols ZigBee and Z-Wave are used for smart lamps and thermostats. These protocols can connect to a large number of devices, provide high security performance, and support self-networking.

For cameras and air conditioners, we use the Wi-Fi protocol; for air quality monitors, we use the IPv6-based 6LowPAN protocol. There are so many different possible applications at home, such as lighting, security, and entertainment. Different protocols are used for different devices. Therefore, multiple protocols and communication technologies are developed, and developers need to adapt and interconnect them.

### 11.2.2 IoT OS Development

In the OS itself, developers also face a very big challenge. Take the smart band as an example. It has many sensors, such as the acceleration sensor and heart rate detector, which all need to be managed together. In terms of videos, a high-definition video has a large file size and consumes a lot of power. For example, when a portable camera is used to shoot a high-definition video such as 4K 60 fps, it consumes a lot of power. So enable users to capture high-quality videos with low power consumption is a challenge.

To sum up, we need to make our terminals smarter. There are three different aspects of smartness. The first is intelligent management, which is what computers do. The OS of a computer mainly connects and uses resources in all the parts of the computer. The OS of the IoT is no different. It also needs to manage various IoT terminals in a unified manner and allocate resources. The second is intelligent connectivity. We need to enable devices that use different communication protocols to communicate with each other. The third and final aspect is intelligent networking. We need to use the mesh networking technology to build a smart, self-organized, and self-connected network for all devices.

## 11.3 Huawei LiteOS

### 11.3.1 Introduction to Huawei LiteOS

Based on the requirements for terminals that we have just mentioned, Huawei provides the IoT OS LiteOS to make terminals smarter. LiteOS is an open-source OS, which uses the "1 + N" architecture. 1 indicates a kernel, and N indicates N middleware. All the middleware can be added or removed according to the user's needs, like the different parts of the Iron Man's suit. The purpose of building such a lightweight and flexible OS is to quickly open up the IoT market and enable a large number of developers to participate in the construction of such an IoT ecosystem.

## 11.3.2 Commercial Use of Huawei LiteOS

Currently, many commercial devices in the industry are running on the LiteOS system. As we have just mentioned, such a lightweight operating system is designed to quickly open up the IoT market. In the NB-IoT chip field, LiteOS supports more than 50% of chips. In the MCU joint commercial solution, LiteOS supports 90% of mainstream MCU chips to make it easier to launch the IoT solution. In addition, not only mobile phones, but also wearable devices and other types of IoT terminals are running on our LiteOS, and their delivery volume reaches millions or even tens of millions. Therefore, the market in this field is huge.

At the beginning, Huawei LiteOS was used in mobile phones or wearable devices in the consumer field. Afterwards, the OS was applied to various terminals in the IoT industry. The reason why LiteOS can be applied to the two different fields is that the two seemingly different scenarios have common requirements.

## 11.3.3 Huawei LiteOS Industry Applications

In the consumer mobile phone field, Huawei LiteOS runs on coprocessors loaded with the Kirin series chipsets. It is mainly used to control all sensors in the mobile phone, with the aim of reducing power consumption and improving measurement accuracy.

The same is true with smart bands in the consumer field, for which it is mainly used to reduce power consumption. HUAWEI Band was the first smart device to showcase Huawei LiteOS. It can control all sensors on the band, and Huawei LiteOS smart sensing framework helps solve problems in multi-sensor high-precision sampling and data synchronization. Because the power consumption is reduced, the standby time of the device is prolonged.

In addition, LiteOS can be used on IoT terminals, such as cameras. With the LiteOS system, the camera can be quickly started up. Due to LiteOS's low power consumption, the camera can be battery-powered, meaning it can be conveniently deployed outdoors. Also, features such as smart hibernation and quick wakeup of Huawei LiteOS enable cameras to save power and quickly respond to events. Quick camera wakeup is especially important in the smart home field. For example, if an intruder enters a house and the camera cannot be quickly woken up and respond to the intrusion event, the opportunity to take a picture of the intruder may be missed. As such, the camera fails to fulfill the very task it was designed for. The smart camera wakeup function effectively overcomes this.

Similarly, LiteOS can be used in smart homes for various terminals inside the home. Huawei LiteOS instructs mobile phone sensors (such as the screen) to implement mobile phone sensor identification and interconnection between smart home appliances. In this way, users can perform operations on other home devices when the mobile phone screen is turned off. In addition, Huawei LiteOS can provide more concise and intelligent operation experience based on features such as smart scenario awareness of mobile phones. Huawei LiteOS also optimizes the interconnection protocols at multiple layers, such as the OS layer and network connection protocol layer. One the one hand, Huawei LiteOS implements more real-time communication between devices, achieving "zero" wait for user operations, while on the other, Huawei LiteOS ensures more reliable and smooth connections and reduces frame freezing during user operations.

In the LPWA field, Huawei LiteOS can be used in solutions such as smart water meters and smart parking. In the smart water metering solution, Huawei LiteOS can be directly used on NB-IoT chips to control sensors. Unlike other OSs which control the MCU, use the MCU to control the bottom-layer sensors, and use AT commands to connect NB-IoT chips, Huawei LiteOS can directly control the communication module and control the sensors to directly communicate with the base station.

In the smart parking field, Huawei LiteOS is used in the geomagnetic vehicle detector and runs on the NB-IoT chip. The lightweight kernel and other components in the OS work with Huawei NB-IoT chips to integrate sensing and interconnection. One chip supports both data transmission through NB-IoT and application sensing algorithms. Based on its open APIs, Huawei LiteOS enables parking sensing algorithms and applications to be seamlessly migrated to NB-IoT chips for deployment. In addition, the Huawei LiteOS sensing framework optimizes application algorithms and improves efficiency, eliminating unnecessary MCU, memory, and storage costs. By default, the Huawei LiteOS interconnection protocol stack supports interconnection with the HUAWEI CLOUD IoT platform. In addition to making devices plug-and-play, the Huawei LiteOS interconnection protocol stack collaborates with the HUAWEI CLOUD IoT platform to upgrade firmware, algorithms, and applications, reducing the operation costs of the parking system. Huawei LiteOS uses the lightweight platform architecture, dynamic and distributed loading technology, and "Run-Stop" mechanism to drive the entire software stack to implement ultra-low power consumption in standby state, allowing parking devices to operate with extremely low power consumption.

# 12 Sensing Layer Development

This chapter describes the knowledge required for the IoT sensing layer from the perspective of development.

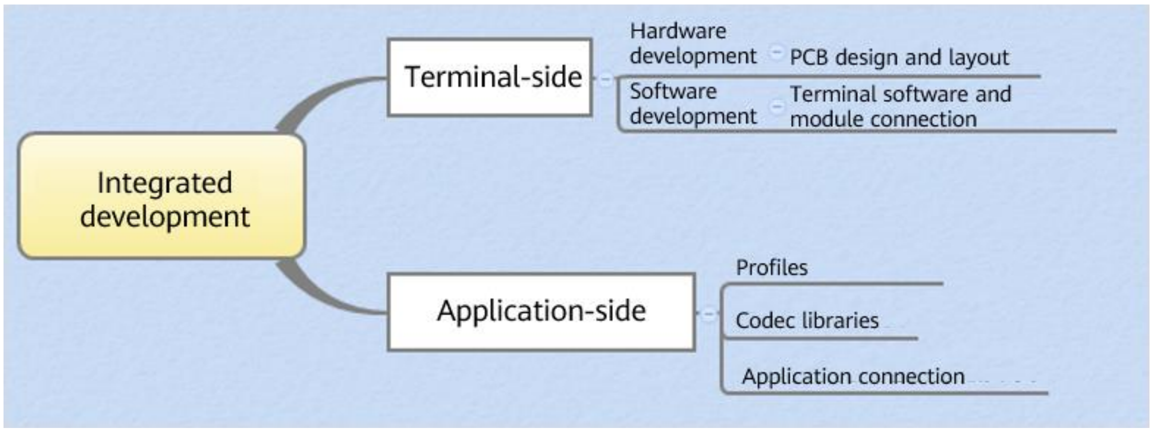


**Figure 12-1 E2E integrated IoT development**

Figure 12-1 shows all the parts required for end-to-end integrated IoT development, which is divided into two sections: development on the terminal side and development on the application (platform) side, which have been described in chapter 10. The development on the terminal side can be divided into two parts. The first is hardware development, which includes the PCB design. As the experiment in this course does not involve hardware development, this chapter only describes some basic knowledge regarding single-chip microcomputers and sensors.

For the second part, software development, we must become familiar with the development of terminal software and module interconnection. This part will introduce the basic architecture of the Huawei LiteOS system to help you complete the software development on the terminal side.

This chapter will be divided into two parts: hardware development at the sensing layer and software development at the sensing layer.

## 12.2 Hardware Development at the Sensing Layer

### 12.2.1 Single-Chip Microcomputer Basics

#### 12.2.1.1 Microcomputer Working Principles

Because in the previous chapters, there is no relevant content about the single-chip microcomputers and readers may know little about the single-chip microcomputers, this section will start from the computer that readers are familiar with to introduce the single-chip microcomputer.

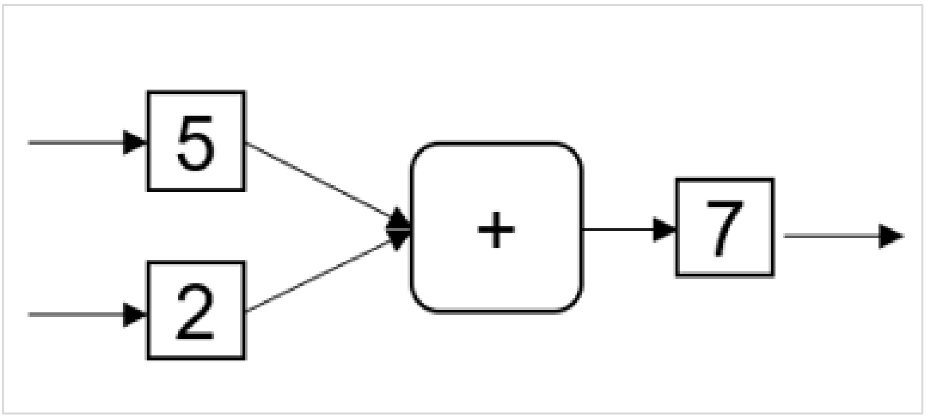


**Figure 12-2 Single-function circuit**

The most important function of a computer is its ability to compute. As such, during hardware development, we will develop and design hardware circuits to implement the computing function. This requires knowledge of digital and analog circuits, which is too complex to expand on here.

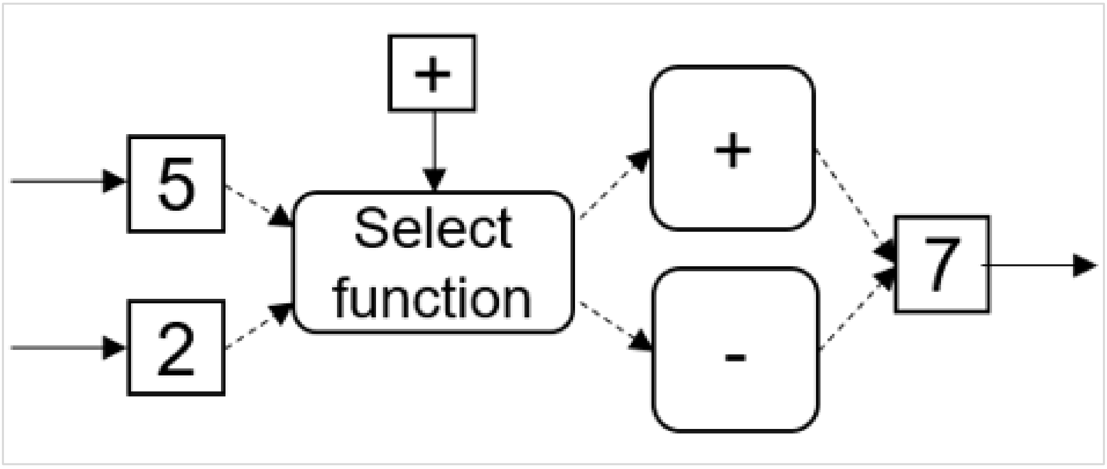


**Figure 12-3 Circuit with the select function**

As shown in Figure 12-2, for example, if developers want a computer to be able to add two numbers, they can design circuits to achieve this function. However, if they want to implement additional functions, such as the subtraction of two numbers, they must design additional circuits that are totally independent of the previous one. This is problematic and often a waste of space and manufacturing costs. Therefore, if we want to implement multiple functions in the circuits, a multiplexer shown in Figure 12-3 must be added to properly select each one.

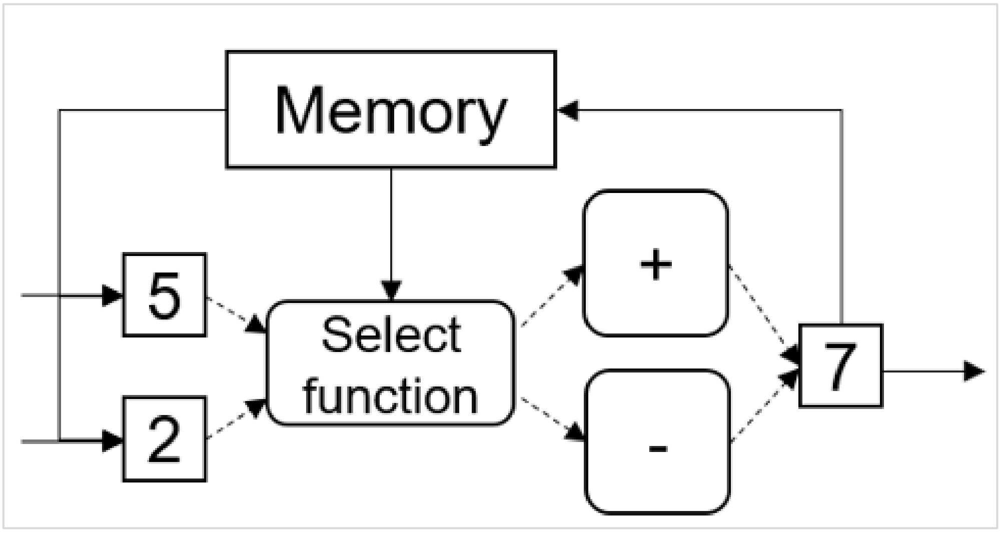


**Figure 12-4 Modern computer model**

But while we may have multiple functions, the most important memory function of the computer has not yet been realized. A circuit with no memory means that it can run only once, and it can be very complicated to repeat this step from scratch after the result is obtained. In this case, the registers shown in Figure 12-4 need to be added to store the previous data or the commands that you want to run. After the circuit contains each of these elements, it can be regarded as a complete computer. All modern computers are designed on the basis of this principle, and only vary based on the complexity of the circuit.

After we understand how a computer works, it becomes clear that implementing different functions is equivalent to executing the content in a program. A program actually runs through a series of simple commands to process computer data step by step. Here comes another important concept in the single-chip microcomputer, that is, the instruction set.

#### 12.2.1.2 Instruction Set

On a computer, an instruction is a command that directs the computer hardware to perform an operation or process a function. An instruction is the smallest functional unit for a computer to run, and hardware implements the functions specified by each instruction. The computer instruction system, also called an instruction set, is the collection of all instructions on that computer and embodies all of the computer's functions. That is, the instruction set must be designed based on the functions, and then implemented on the hardware according to the instruction set. Beyond instructions, the instruction system also includes instruction formats, addressing modes, and data formats for all instructions.

Therefore, the instruction system executed by each computer determines both the capability that the machine requires, as well as its instruction format and the structure.

However, machines that have different structures and instruction formats should have matching instruction systems. Therefore, when designing the instruction system, we should pay attention to the instruction format, type, and operation function. The software is a program of various systems and users written for computers to use. The program consists of a sequence of computer instructions. From this point of view, instructions are a unit of computer language used to design programs.

For example, a programmer wants to design a program that converts degrees Celsius into degrees Fahrenheit, which involves a number of different steps: first **read**ing this Celsius degree data, **subtract**ing 273 from it, **multiply**ing the result by 1.8, and **add**ing 32. Finally, the result is **output**. In this program, each step represents a data operation instructed by the computer. For this reason, the instruction set of a CPU is very important. That is, whether the instruction set is advanced or not affects the performance of the CPU, which means that it is an important indicator of the CPU performance. Indeed, the instruction set is one of the most effective tools to improve the efficiency of microprocessors.

In terms of a current mainstream architecture, an instruction set may be divided into two parts: a complex instruction set computer (CISC) and a reduced instruction set computer (RISC). Typical examples of CISC include Intel and AMD x86 CPUs, while RISC is typified by ARM and IBM Power. Most IoT devices use the ARM architecture, with its CPU using the RISC.

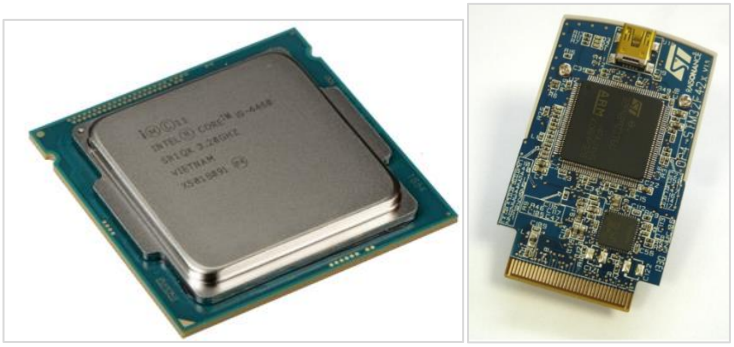
The RISC is designed to reduce the complexity of the CISC CPU by selecting some instructions that can be executed in a single CPU cycle. For example, a multiplication instruction provided by CISC can be invoked to multiply two numbers in memory a and memory b, and save the result in memory a. It may take multiple CPU cycles to complete the multiplication. The RISC does not provide one-stop multiplication instructions. Rather, it needs to invoke four single-CPU cycle instructions to multiply two numbers: memory a is loaded to one register, memory b is loaded to another register, and the numbers in the two registers are multiplied together. The register result is saved to memory a. According to this, the number of instructions in the RISC designed in the early stage is less than that in the CISC. Later on, however, the number of instructions in the RISC exceeds that in the CISC. Therefore, it is the complexity, not the quantity, of instructions that is referenced to distinguish the two types of instruction sets.

Of course, the CISC also needs to complete complex instructions by operating a memory, a register, and an arithmetic unit. During the implementation, complex instructions are converted into a microprogram, which is stored in the microservice memory when the CPU is manufactured. A microprogram contains several microinstructions (also called microcodes). When a complex instruction is executed, it is actually a microprogram that is executed. This also brings about a difference between the two instruction sets. The execution of a microprogram cannot be interrupted, while RISC instructions can be interrupted. Theoretically, RISC can respond to interrupts more quickly.

Therefore, the advantage of RISC is that the RISC system runs 2 to 4 times faster than the CISC system with the same chip running at the same clock speed. Because the RISC processor's instruction set is compact, its memory tubes and floating point units can be designed on the same chip. RISC processors are easier to design than CISC processors, and require shorter time. RISC processors can use more advanced technologies than CISC processors and develop next-generation processors faster.

### 12.2.1.3 Processing Units

After understanding the working principle of a computer and the concept of an instruction set, we next need to examine the integrated circuits and processing units responsible for computing. First of all, the processing unit can be divided into microprocessors and microcontrollers. The microprocessor is what we call the CPU, and this is typically used to handle large-scale, very complex computing. A microprocessor can only be used to process data, and all other necessary components (such as memory) must be connected externally to use.



(a) (b)

**Figure 12-5 (a) Microprocessor (b) Microcontroller**

In contrast, the microcontroller — also known as a single-chip microcomputer - is a very complete device. It contains a CPU, memory, and a range of input and output devices. Because the computing power of the CPU in the microcontroller is much weaker than that of the MPU, the CPU is generally designed for certain embedded devices with low power consumption.

Another big difference between microcontrollers and microprocessors is the clock frequency of the CPU. The clock frequency of the MPU is relatively high because the MPU is usually designed for highly complex task calculations, while the MCU is usually used in some low-power-consumption devices or IoT devices without a complex computing capability. To ensure sustained usage, the CPU clock frequency in the MCU is usually relatively low.

### 12.2.1.4 Architecture of the Single-Chip Microcomputer

The architecture of the single-chip microcomputer can be divided into four different parts: CPU, memory, input and output interfaces, and system bus. This section describes these parts one by one. The first is the CPU, which is mainly used for data processing and computing. The CPU is used to control the entire system in the single-chip microcomputer. It reads and decodes program instructions in serial mode and executes the tasks that require the processor, before finally generating control signals for other tasks. It performs all arithmetic and logical operations, and MCUs with the same processor can execute the same program.

The memory and address decoding circuits are included in the storage system. The memory is classified into two different types. Random access memory, also known as RAM, is used to store data running in programs. All data stored in RAM is temporary, and will disappear after a power failure. The second type of memory is known as read-only memory, or ROM, and is used to store programs that need to be executed by a single-chip microcomputer. Data stored in ROM will not disappear after a power failure, and will always remain available.

The difference between the ROM and RAM is that the latter is similar to the memory of a computer and does not store data. However, the RAM occupies the space for caching data during daily use. ROM, on the other hand, is the computer's hard drive, which is used to store large amounts of data that will not disappear after the computer is shut down.

The subsequent input and output interfaces are used to connect to external devices, such as digital I/O and analog I/O. You can use devices such as data cables to connect them.

Finally, the system bus acts like a wire in a circuit, connecting all parts together for data communication. Generally, buses can be classified into three types: data bus, address bus, and control bus. As the name implies, the data bus is used to transmit data between the CPU, memory, and external devices. The address bus is used by the selection processor to read or write to a specific memory location, and is characterized by its unidirectional data flow from the CPU to address bus, and then memory.

The last control bus is used to transmit control or signaling data.

#### 12.2.1.5 ADC and DAC

After understanding the components of the single-chip microcomputer, we can find that the functions of each component are primarily concerned with data processing and transfer. It is also important to understand that data processed by the single-chip microcomputer is digital (0 and 1). The single-chip microcomputer is composed of a series of complex circuits. However, current, voltage, and other data contained in the circuits exist in the form of simulation. Therefore, it is essential for a computer to use digital signals to represent the analog signals found in nature. To achieve this, the DAC and ADC modules in the single-chip microcomputer are required.

ADC stands for analog-to-digital converter, while DAC stands for digital-to-analog converter. They may have opposite functions, but both are used for conversion between analog and digital signals. For example, if we have a simple 2-bit ADC interface, a 2-bit binary number can represent the following four digits: 00, 01, 10, and 11. Alternatively, we can say that the voltage values of the interface are classified into four levels. In addition, the rated voltage of a single-chip microcomputer is 3.3 V. Consequently, when the voltage is divided into four levels, it is represented by the values 0 V, 1.1 V, 2.2 V, and 3.3 V.

### 12.2.2 Sensing Technology Basics

After learning the relationship between digital and analog signals, let's move on to sensors. As a device that collects analog signals in daily life, some sensors contain an ADC module, enabling them to directly convert analog data in the environment into digital signals for output.

Sensing technology, otherwise known as sensor technology, can sense the surrounding environment and collect analog signals to form a series of data. As mentioned previously, sensing technology resembles the sensory system of our human body, and our five senses can be loosely compared to five sensors of different types. For example, vision is a photosensitive sensor and hearing is a sound sensitive sensor. Meanwhile, our sense of smell is a gas sensor, taste is a chemical sensor, and the sense of touch is a pressure sensor or a temperature sensor.

As the precision of a sensor determines the accuracy of the data collected by the IoT, sensor technology is of vital importance.

Next, let's take a look at some common sensors.

#### 12.2.2.1 Photoelectric Sensor

Photosensitive sensors are widely utilized across various scenarios, and use photosensitive elements to convert optical signals into electrical signals. For example, both outdoor light intensity detection and indoor light monitoring use photosensitive sensors to determine whether people are indoors.

**Figure 12-6 Photoelectric sensor**

#### 12.2.2.2 Temperature Sensor

Temperature sensors are more widely used throughout our daily lives, often employed to measure the working environment of various devices, and even appearing in wristbands to detect body temperature. Temperature sensors can be divided into two types: resistance sensing and thermocouple sensing. The goal of resistance sensing is to design a temperature sensor based on the varying resistance values of metal at different temperatures. Meanwhile, thermocouple sensing involves connecting two different metals. When the temperature changes, the potential difference in the thermocouple circuit can be used to calculate it.

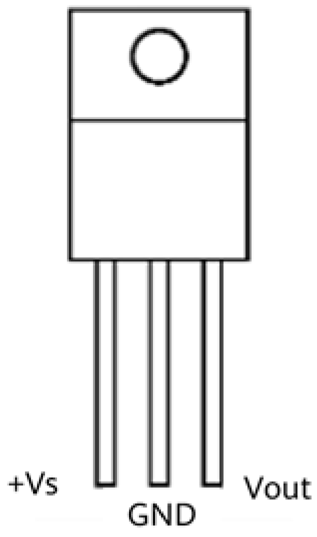


**Figure 12-7 Temperature sensor**

### 12.2.2.3 Acceleration Sensor

Acceleration sensors can be found in devices such as mobile phones, wristbands, and gaming devices, and are used to perform such tasks as step counting, earthquake monitoring, and game control. Typically, acceleration sensors work by leveraging the piezoelectric effect, and mainly perform acceleration calculation using a manner in which a voltage is generated due to crystal deformation caused by acceleration inside the sensor.

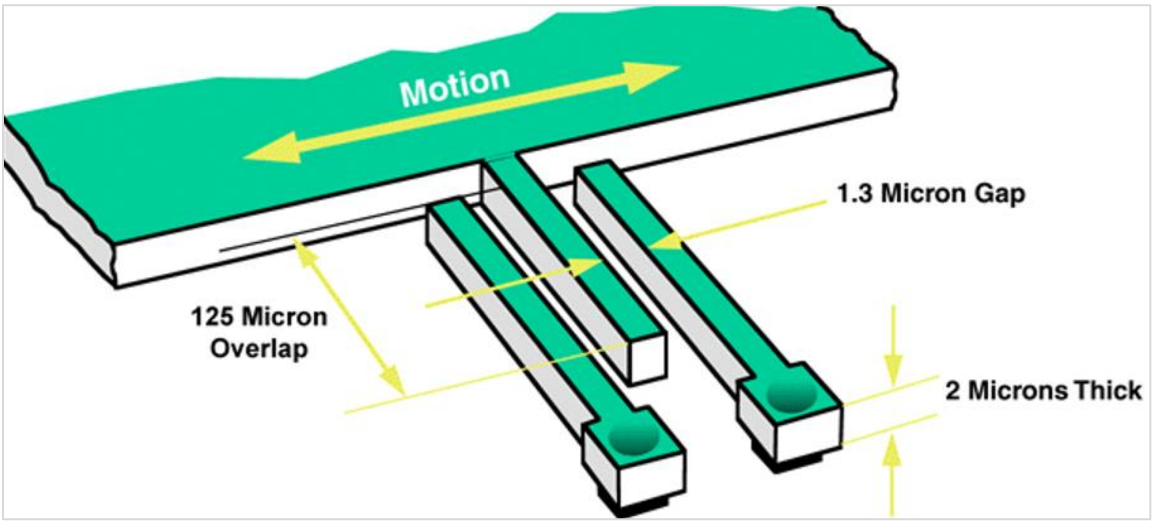


**Figure 12-8 Acceleration sensor**

### 12.2.2.4 Heart Rate Sensor

Last but not least is the pulse sensor, which converts pulse waves into pulse signals. This sensor is typically used in health monitoring equipment, where the pulse sensor can be divided into piezoelectric, piezoresistive, and photoelectric types according to different collection methods. The piezoelectric and piezoresistive types both collect pulse signals through the use of a pulse, while the photoelectric pulse sensor utilizes an infrared detection method to measure changes in light transmittance within a blood vessel during a pulsation process.

## 12.3 Software Development at the Sensing Layer

This section focuses on the content of Huawei LiteOS. With a good understanding of LiteOS, developers can better develop applications on devices. Let's first examine the overall framework of LiteOS. At the bottom layer is the chip with various instruction sets. LiteOS can be carried on mainstream chips such as ARM and x86 CPUs, in addition to Huawei-developed chips. Now we have the LiteOS operating system, on which the applications of different industries are connected.

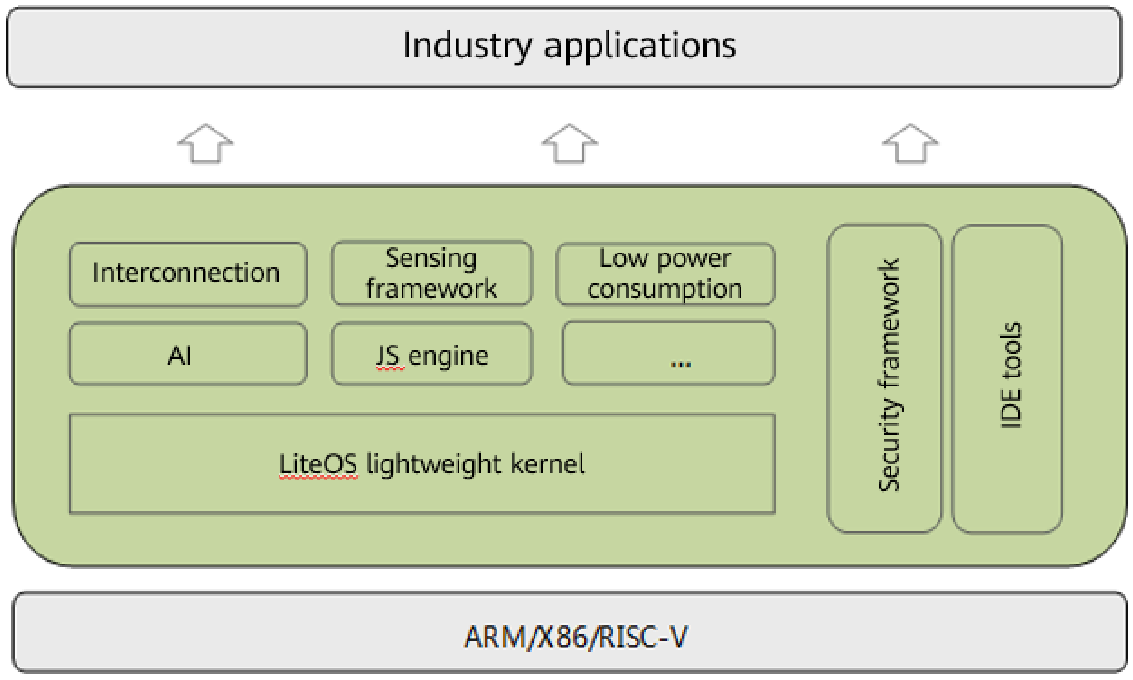


**Figure 12-9 Huawei LiteOS architecture**

## 12.3.2 Huawei LiteOS Kernel

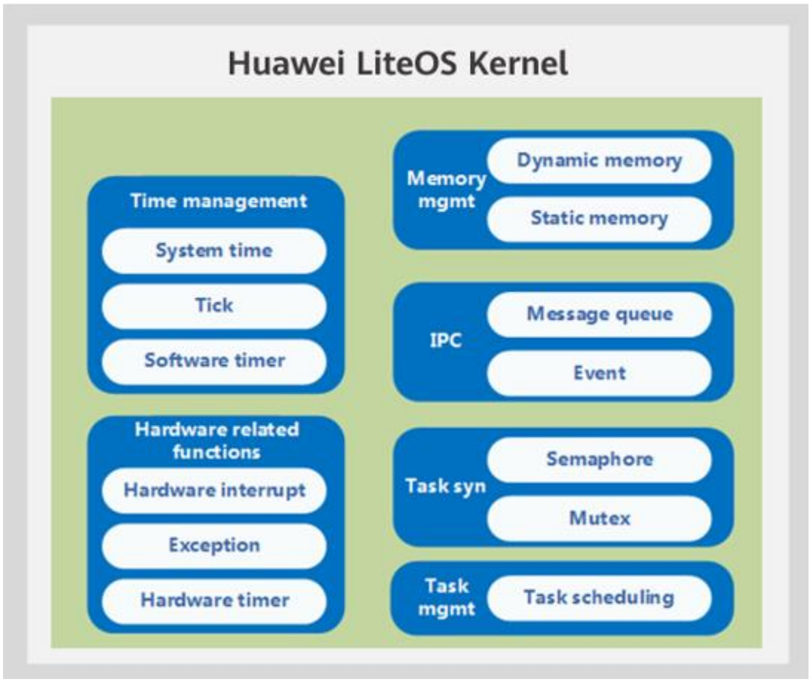


**Figure 12-10 Basic framework of Huawei LiteOS kernel**

The kernel is the most important element of the OS, and serves as the core. While the LiteOS system can have no other middleware, it must possess this kernel. There are many components in the kernel, including task management, memory management, and time management components. The functions of the entire operating system are based on tasks, which are the smallest units processed by the OS. Tasks are processed by managing time and hardware.

LiteOS is most often applicable to the IoT field, which is characterized by low power consumption. Most devices in the IoT field can be used for several years without a battery change, requiring us to meet strict requirements for low power consumption. In addition to network technology and chip design, the operating system must also meet these requirements. As such, the tickless low-power mechanism is supported, enabling the OS to cooperate with the hardware in order to enter a low power consumption mode.

### 12.3.2.2 Task

Among all the modules in the kernel, let's focus on the task module first. Tasks are the smallest running units of an OS. That is, if we want to use an OS to perform a series of operations, these operations are completed by smaller tasks. A task can use system resources such as CPU and memory, and each task is independent. What does this mean? If we divide a problem into several small tasks and require each to calculate different parts of the problem, these tasks can be processed without following a specific sequence. For example, when calculating 1+1+1+1, we can arrive at the correct result regardless of which step of the calculation is performed first.

In addition, it is important to note that the task scheduling mechanism is based on priorities. That is, tasks are marked with different priorities. When tasks have the same priority, time slice rotation scheduling is called on to divide the time segment. The CPU processes one task for a period of time before then processing the next, thereby alternating time segments.

For example, three people (A, B, and C) want to play the same game console. If we use the time slice rotation scheduling mode to handle this problem, we can let A play for 10 minutes first, then B play for 10 minutes, and finally C play for 10 minutes. The cycle will then repeat again and again. This is the easiest and fairest way to deal with the problem.

In LiteOS, there are a total of 32 priorities for tasks, the highest of which is 0. A high-priority task can interrupt a low-priority task at any time, and a low-priority task can only be calculated after the high-priority process ends.

**Table 12-1 Task interface description**

| Function Category | Interface Name | Description |
|---|---|---|
| Task creation and deletion | LOS_TaskCreate | Creates a task, so that the task is ready for scheduling. |
| | LOS_TaskDelete | Deletes a task. |
| Task status control | LOS_TaskResume | Resumes a suspended task. |
| | LOS_TaskSuspend | Suspends a task. |
| | LOS_TaskDelay | Delays a task. |
| Task scheduling control | LOS_TaskLock | Locks a task. |
| | LOS_TaskUnlock | Unlocks a task. |

As described in Table 12-1, many different interfaces are used when we perform different operations with tasks. For example, when we execute the creation and deletion tasks, the LOS_TaskCreate and LOS_TaskDelete interfaces are used. Many other interfaces are also invoked during various operations.

To help readers better understand task management, this document provides the following programming example code for references:

```
#include "math.h"
#include "time.h"
#include "los_task.h"
#include "los_api_task.h"
#include "los_inspect_entry.h"

#ifdef __cplusplus
#if __cplusplus
extern "C" {
#endif /* __cpluscplus */
#endif /* __cpluscplus */

static UINT32 g_uwTskHiID;
static UINT32 g_uwTskLoID;
```

```
#define TSK_PRIOR_HI 4
#define TSK_PRIOR_LO 5

static UINT32 Example_TaskHi(VOID)
{
        UINT32 uwRet = LOS_OK;

        dprintf("Enter TaskHi Handler.\r\n");

        /* task delay 5 ticks, task will be suspended */
        uwRet = LOS_TaskDelay(5);
        if (uwRet != LOS_OK)
        {
                dprintf("Delay Task Failed.\r\n");
                return LOS_NOK;
        }

        /* task resumed */
        dprintf("TaskHi LOS_TaskDelay Done.\r\n");

        /* suspend self */
        uwRet = LOS_TaskSuspend(g_uwTskHiID);
        if (uwRet != LOS_OK)
        {
                dprintf("Suspend TaskHi Failed.\r\n");
                uwRet = LOS_InspectStatusSetByID(LOS_INSPECT_TASK, LOS_INSPECT_STU_ERROR);
                if (LOS_OK != uwRet)
                {
                        dprintf("Set Inspect Status Err.\r\n");
                }
                return LOS_NOK;
        }

        dprintf("TaskHi LOS_TaskResume Success.\r\n");

        uwRet = LOS_InspectStatusSetByID(LOS_INSPECT_TASK, LOS_INSPECT_STU_SUCCESS);
        if (LOS_OK != uwRet)
        {
                dprintf("Set Inspect Status Err.\r\n");
        }

        /* delete self */
        if(LOS_OK != LOS_TaskDelete(g_uwTskHiID))
        {
                dprintf("TaskHi delete failed .\r\n");
                return LOS_NOK;
        }

        return LOS_OK;
}

static UINT32 Example_TaskLo(VOID)
{
        UINT32 uwRet;
        dprintf("Enter TaskLo Handler.\r\n");

        /* task delay 10 ticks, task will be suspended */
        uwRet = LOS_TaskDelay(10);
        if (uwRet != LOS_OK)
        {
                dprintf("Delay TaskLo Failed.\r\n");
                return LOS_NOK;
        }

        dprintf("TaskHi LOS_TaskSuspend Success.\r\n");

        /* resumed task g_uwTskHiID */
        uwRet = LOS_TaskResume(g_uwTskHiID);
        if (uwRet != LOS_OK)
        {
                dprintf("Resume TaskHi Failed.\r\n");
                uwRet = LOS_InspectStatusSetByID(LOS_INSPECT_TASK, LOS_INSPECT_STU_ERROR);
                if (LOS_OK != uwRet)
                {
                        dprintf("Set Inspect Status Err.\r\n");
                }
                return LOS_NOK;
        }

        /* delete self */
        if(LOS_OK != LOS_TaskDelete(g_uwTskLoID))
        {
                dprintf("TaskLo delete failed .\r\n");
                return LOS_NOK;
        }

        return LOS_OK;
}

UINT32 Example_TskCaseEntry(VOID)
{
        UINT32 uwRet;
        TSK_INIT_PARAM_S stInitParam;

        /* lock task schedule */
        LOS_TaskLock();

        dprintf("LOS_TaskLock() Success!\r\n");

        stInitParam.pfnTaskEntry = (TSK_ENTRY_FUNC)Example_TaskHi;
        stInitParam.usTaskPrio = TSK_PRIOR_HI;
        stInitParam.pcName = "HIGH_NAME";
        stInitParam.uwStackSize = LOSCFG_BASE_CORE_TSK_DEFAULT_STACK_SIZE;
        /* create high prio task */
        uwRet = LOS_TaskCreate(&g_uwTskHiID, &stInitParam);
        if (uwRet != LOS_OK)
        {
                LOS_TaskUnlock();
```

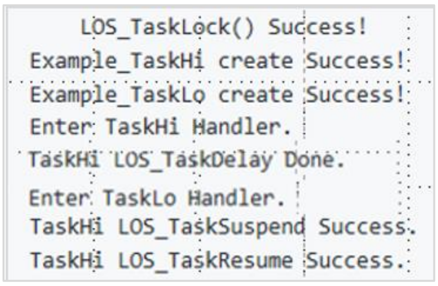
```
            dprintf("Example_TaskHi create Failed!\r\n");
            return LOS_NOK;
        }

        dprintf("Example_TaskHi create Success!\r\n");

        stInitParam.pfnTaskEntry = (TSK_ENTRY_FUNC)Example_TaskLo;
        stInitParam.usTaskPrio = TSK_PRIOR_LO;
        stInitParam.pcName = "LOW_NAME";
        stInitParam.uwStackSize = LOSCFG_BASE_CORE_TSK_DEFAULT_STACK_SIZE;
        /* create low prio task */
        uwRet = LOS_TaskCreate(&g_uwTskLoID, &stInitParam);
        if (uwRet != LOS_OK)
        {
            /* delete high prio task */
            if (LOS_OK != LOS_TaskDelete(g_uwTskHiID))
            {
                dprintf("TaskHi delete failed .\r\n");
            }

            LOS_TaskUnlock();

            dprintf("Example_TaskLo create Failed!\r\n");

            return LOS_NOK;
        }

        dprintf("Example_TaskLo create Success!\r\n");

        /* unlock task schedule */
        LOS_TaskUnlock();

        return uwRet;
}

#ifdef __cplusplus
#if __cplusplus
}
#endif /* __cpluscplus */
#endif /* __cpluscplus */
```

This program simulates how the OS operates in the case of high and low priorities. Two tasks are defined at the beginning of the program. One is named **High**, and the other is **Low**. In addition, two priorities, 4 and 5, are assigned to the two tasks. As mentioned above, a smaller value indicates a higher priority in LiteOS, making 4 the higher priority. At the beginning, the program uses the **UINT32 Example_TskCaseEntry (VOID)** function to create the two tasks. First of all, because the activity of creating a task is itself a task, you need to lock the task to prevent other high-priority tasks from interrupting it. Then, the program displays a message indicating that the task is locked successfully.

Next, the program creates the two tasks. After the tasks are created successfully, the program displays a message indicating that the **TaskHi** and **TaskLo** tasks are created successfully. After that, the program unlocks the locked task. Because no command is printed in this step, the unlock operation is not displayed in the compilation result.

After the two tasks are created, you need to execute them. The first task is **static UINT32 Example_TaskHi(VOID)**, which has a high priority. At the beginning of the function, the system displays "Enter TaskHi Handler.\r\n". The program comments indicate that the program will suspend the task after five ticks. Similarly, the first step of the low-priority **static UINT32 Example_TaskLo(VOID)** function is to display the information of entering the task. While different from the high-priority task, the low-priority task is suspended after 10 ticks. Technically, a task with a higher priority is suspended earlier than a task with a lower priority. The high-priority task function continues after the delay. As it is displayed, after the **TaskHi** delay ends, the task suspends itself.

Because the task with a higher priority is suspended, the OS automatically starts to execute the task with a lower priority. Therefore, the program enters the **TaskLo** task with a lower priority. After the **TaskLo** task is delayed, the **TaskHi** task is suspended, and the program displays a message indicating that the high-priority task has been successfully suspended. After that, the program executes the low-priority task to delete itself. Once this step is complete, because the low-priority task has been completed and deleted, the OS jumps to handle the suspended high-priority task. The system switches to the high-priority task and displays a message stating that the **TaskHi** task continues. The task with a higher priority is executed successfully and is deleted. Figure 12-11 shows the compilation result of the entire process.



```
    LOS_TaskLock() Success!
Example_TaskHi create Success!
Example_TaskLo create Success!
Enter TaskHi Handler.
TaskHi LOS_TaskDelay Done.
Enter TaskLo Handler.
TaskHi LOS_TaskSuspend Success.
TaskHi LOS_TaskResume Success.
```

**Figure 12-11 Compilation result of the task management programming example**

## 12.3.2.3 Memory

Next, let's look at how memory is managed in the kernel module. Memory is an important part of a computer system, and is responsible for data storage and invoking. Data is generated as the CPU operates, and this data cannot be stored or invoked without memory. As a result, memory management is very important, and memory usage must be carefully planned. For example, when allocating memory, we need to consider the optimal usage of both large and small blocks. Otherwise, improper allocation may lead to a waste of memory resources.

The memory management module is responsible for initializing, allocating, and releasing memory. LiteOS memory management is classified into static and dynamic memory management, and Huawei LiteOS provides two types of memory management algorithms: membox for static memory allocation; and bestfit, bestfit_little, and two-level segregated fit (tlsf) for dynamic memory allocation.

Now we'll take a closer look at dynamic memory management. Dynamic memory refers to allocating memory blocks of a specified size in the dynamic memory pool, which is a large memory resource configured in the system. Dynamic memory enables memory resources to be allocated based on user requirements, which is more cost-effective for users who require large memory resources. The main disadvantage of this approach is that resources are allocated on demand, requiring both a large block of memory and a small block of memory. When the small block is allocated, the identifier at the beginning of the block is used to store information about the size of the small block of resources. This leads to a waste of resources, as a large proportion of small block resources becomes occupied. In addition, the allocation of resources on demand results in a high probability that fragments are generated, as smaller parts of the large block of resources cannot be allocated.

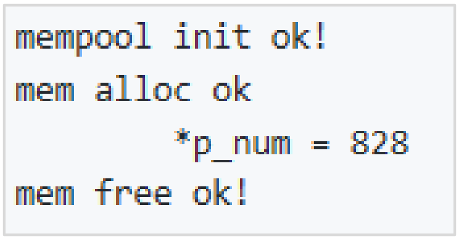
Static memory management uses a different technique, instead allocating memory resources in the resource pool to users in a fixed size. The advantage of this approach is that higher levels of performance and efficiency can be achieved compared with dynamic memory. In addition, as the size of each resource is fixed, the allocated memory has no management structure, leading to higher resource usage than with dynamic memory, and eliminating the fragmentation problem in the process. The main disadvantage, however, is that the size is fixed and cannot be allocated on demand, and is therefore not suitable for users with high memory resource requirements.

This document also provides programming examples about memory management. The example code is as follows:

```
#include <stdlib.h>
#include "los_config.h"
#include "los_memory.h"
#include "los_api_dynamic_mem.h"
#include "los_inspect_entry.h"

#ifdef __cplusplus
#if __cplusplus
extern "C" {
#endif /* __cpluscplus */
#endif /* __cpluscplus */

#define MEM_DYN_SIZE    256
static UINT32 pDynMem[MEM_DYN_SIZE/4];
extern UINT32 LOS_MemInit(VOID *pPool, UINT32 uwSize);

UINT32 Example_Dyn_Mem(VOID)
{
        UINT32 *p_num = NULL;
        UINT32 uwRet;
        uwRet = LOS_MemInit(pDynMem, MEM_DYN_SIZE);
        if (LOS_OK == uwRet)
        {
                dprintf("mempool init ok! \r\n");
        }
        else
        {
                dprintf("mempool init failed! \r\n");
```

```
            return LOS_NOK;
        }

        /* mem alloc */
        p_num = (UINT32 *)LOS_MemAlloc(pDynMem, 4);
        if (NULL == p_num)
        {
            dprintf("mem alloc failed! \r\n");
            return LOS_NOK;
        }
        dprintf("mem alloc ok \r\n");

        /* assignment */
        *p_num = 828;
        dprintf("*p_num = %d \r\n", *p_num);

        /* mem free */
        uwRet = LOS_MemFree(pDynMem, p_num);
        if (LOS_OK == uwRet)
        {
            dprintf("mem free ok!\r\n");
            uwRet = LOS_InspectStatusSetByID(LOS_INSPECT_DMEM, LOS_INSPECT_STU_SUCCESS);
            if (LOS_OK != uwRet)
            {
                dprintf("Set Inspect Status Err \r\n");
            }
        }
        else
        {
            dprintf("mem free failed! \r\n");
            uwRet = LOS_InspectStatusSetByID(LOS_INSPECT_DMEM, LOS_INSPECT_STU_ERROR);
            if (LOS_OK != uwRet)
            {
                dprintf("Set Inspect Status Err \r\n");
            }
            return LOS_NOK;
        }
        return LOS_OK;
}

#ifdef __cplusplus
#if __cplusplus
}
#endif /* __cpluscplus */
#endif /* __cpluscplus */
```

This document takes the allocation of dynamic memory as an example and provides code to explain the process of memory allocation. This program consists of five steps. The first step is to initialize a dynamic memory pool. After the initialization is successful, the program displays a message indicating that the memory is successfully initialized. The second step is to apply for a memory block in the dynamic memory pool, which equates to allocating the memory in the program. After that, the program displays a message indicating that the memory is successfully allocated. In the third step, the program assigns a value to the memory block, and the OS stores the data in the memory block. In the fourth step, the program displays a message indicating that the memory is successfully allocated and displays the values in the resource block. Finally, after the program accomplishes tasks with the memory resource, the program releases the memory resource and displays a message indicating that the memory is successfully released. The compilation result is also displayed according to this process.



```
mempool init ok!
mem alloc ok
        *p_num = 828
mem free ok!
```

**Figure 12-12 Compilation result of the memory management programming example**

### 12.3.2.4 Interrupt

The concept of an interrupt is very important with regard to computers. For example, we have already mentioned that a high-priority task can interrupt a low-priority task in 2.2.1.1, and interrupt technology is used in many specific operations. The best way to understand this concept is with examples from the real world. For instance, if the phone rings when we are working on a computer, we will stop using the computer and answer the phone instead. Once we have ended the call, we will continue with our work.

This example shows how we often interrupt one task in order to deal with another. The interrupt handling method is very similar to our example. The phone's ring tone is equivalent to an interrupt request, and answering the call is equivalent to an interrupt response. An interrupt refers to the process in which the CPU stops to execute a new program when necessary. As such, an interrupt is not an instant trigger, but an entire process which ensures the working efficiency of the CPU. As the CPU only needs to respond to other tasks when an interrupt is generated, it does not need to wait and query the peripheral status.

In addition, it is important to note that interrupts also have priorities, as multiple interrupts may occur at the same time in the actual system. In such cases, it is necessary to differentiate the priorities of these interrupts based on urgency.

### 12.3.2.5 Queue

Now that we are familiar with task management, memory management, and interrupt management, let's learn about the communication between tasks. As we have already seen, task management mainly describes tasks with different priorities. However, the coordination of multiple tasks is also a problem that needs to be considered by the operating system. The inter-task communication of the LiteOS system can be implemented in the queue, event, semaphore, and mutex modes, which are used to implement the synchronization, mutex, and communication of multiple tasks in the OS. The following describes each mechanism in detail.

The queue is essentially a data structure that can receive messages from tasks or interrupts, and can choose to deliver or store them. Such a structure is very much like the box containing table tennis balls or badminton shuttlecocks, with the message represented by one of the small balls. After we receive the message, we can choose to save it, similar to placing a ball in the box. At the same time, the message queue can also function as a buffer, storing messages but not processing them until it is idle. A task can read messages in the message queue, and the task processing mechanism is suspended when there are no messages in the queue. When a new message is received, the task is woken up to process it.

#### 12.3.2.6 Event

Events are used to synchronize tasks, and it is important to note that data transmission does not occur between them. Instead, they are used only for synchronization between tasks. Compared with the aforementioned queue, an event is more like a condition for triggering a task. This mechanism can be triggered by one event or by multiple events at the same time. LiteOS uses a 32-bit variable to indicate an event (each bit indicates one event). The value 0 indicates that the event does not occur, and the value 1 indicates that the event has occurred. A total of 31 time types can be set (because bit 25 is reserved). In addition, an event can be sent to a task multiple times, which is equivalent to sending the event only once, and the operating system allows multiple tasks to read and write the same event.

The following figure shows the event operation mechanism.

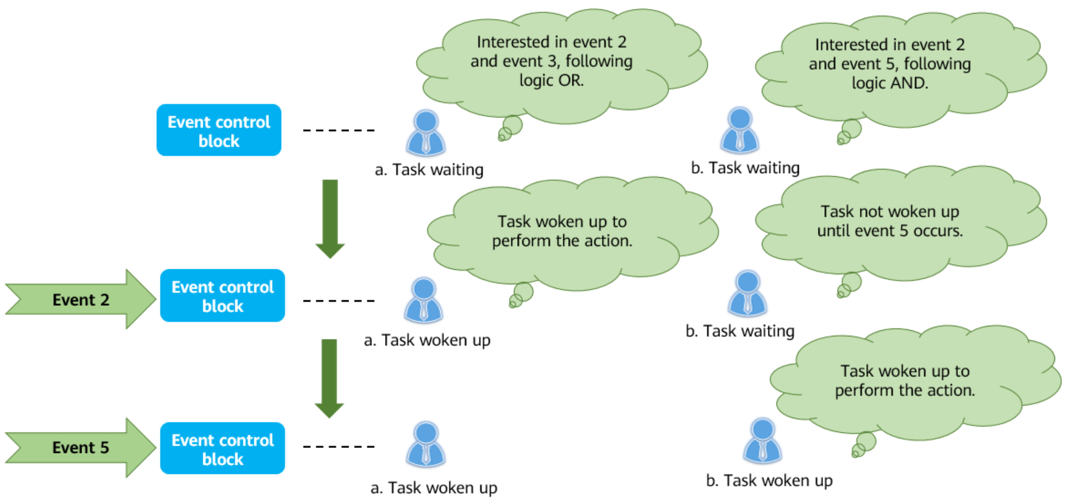

Interested in event 2 and event 3, following logic OR.
Interested in event 2 and event 5, following logic AND.
Event control block
a. Task waiting
b. Task waiting
Task woken up to perform the action.
Task not woken up until event 5 occurs.
Event 2
Event control block
a. Task woken up
b. Task waiting
Task woken up to perform the action.
Event 5
Event control block
a. Task woken up
b. Task woken up

**Figure 12-13 Event operation mechanism**

As shown in Figure 12-13, task A is triggered by event 2 and event 3. The logic is OR, indicating that task A will be woken up when either event 2 or event 3 occurs. In addition, task B is interested in event 2 and event 5, and the logic is AND, which indicates that task B is woken up only when event 2 and event 5 occur at the same time. Therefore, when only event 2 occurs, task A is triggered, while task B is still in the waiting state. When event 5 and event 2 occur together, task A is still in the wakeup state, and task B is also woken up.

### 12.3.2.7 Semaphore

Semaphores are mainly used to implement synchronization between tasks or exclusive access to critical resources. The first semaphore we'll look at is the synchronization semaphore. As the name implies, the synchronization semaphore is used to synchronize tasks, and the specific implementation procedure is as follows:

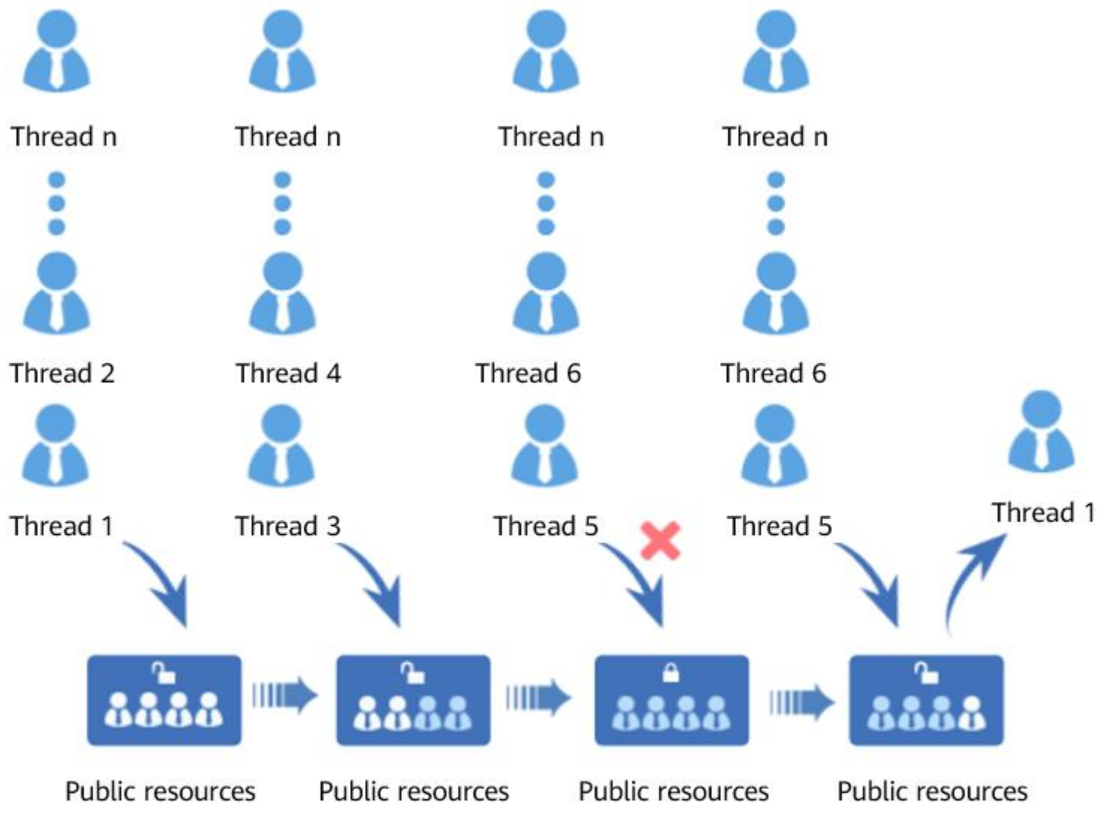


**Figure 12-14 Semaphore operation mechanism**

As shown in Figure 12-14, different threads represent different people, and then our public resources are like a small room that can hold up to four people. When all four enter the room, it is automatically locked so that no one else can enter. In this way, all four individuals in the room are synchronized, just like our four thread characters are synchronized.

### 12.3.2.8 Mutex

The mutex is a special type of semaphore, also known as a binary semaphore, and is used to exclusively occupy shared resources. The mutex has only two states, unlocked or locked.

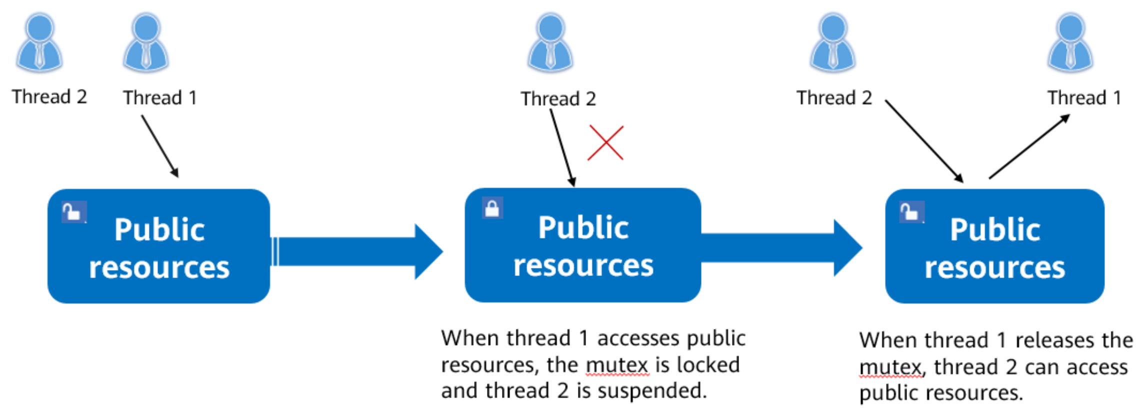

Thread 2
Thread 1
Public resources
Thread 2
Public resources
When thread 1 accesses public resources, the mutex is locked and thread 2 is suspended.
Thread 2
Thread 1
Public resources
When thread 1 releases the mutex, thread 2 can access public resources.

**Figure 12-15 Mutex operation mechanism**

We can use the following example to explain this: imagine a restroom where only one person can enter at a time. The door is locked whenever someone enters, and those outside know that they cannot go in. Only when the first person opens the door and leaves can the next person go inside. This is the principle of the mutex. When the mutex of a task is locked, other tasks cannot use the mutex. When the task is released, the mutex is unlocked and the task loses ownership of the mutex, allowing other tasks to use it instead.

### 12.3.2.9 Time Management

The time management of an operating system is very important because many tasks are directly related to time. For example, we can see here that some programs require us to wait for a period of time (for example, 10 seconds). The waiting time can be set in the program because the operating system manages the time based on the system clock, which is different from the clock we use every day. The system clock uses ticks to count the time, and as the working frequencies of different CPUs are different, their ticks represent different real events. Users can use seconds to count time when performing operations. The time management module is required to convert the generated ticks into user-perceived time.

### 12.3.2.10 Timer

OSs also possess timers, which are primarily used to generate interrupts within a fixed time in order to trigger another step execution. The timer can be classified as either a hardware timer or a software timer, with the number of hardware timers limited by the hardware. As such, the OS also provides a software timer to ensure that user requirements are met.

This approach is similar to virtualization technology on the cloud, where software is also used to reallocate otherwise limited hardware resources. In our case, we use software to simulate the timer, which is then implemented based on the system clock interrupt. For example, the system clock generates an interrupt to achieve the timing effect after counting the number of ticks. In this way, the number of timers is increased, and additional timer-related services can be created.

## 12.3.3 Huawei LiteOS Framework

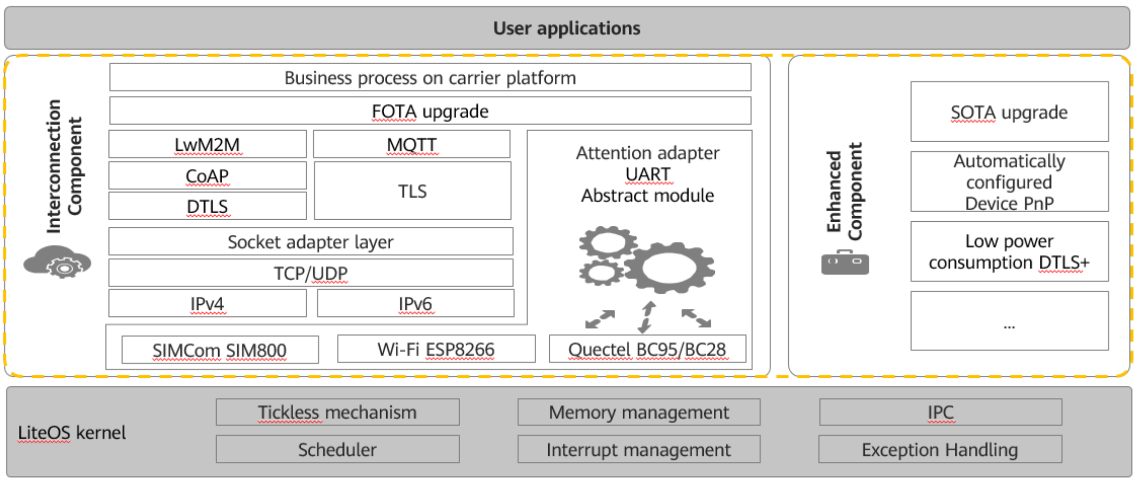


**Figure 12-16 Huawei LiteOS framework (see the Appendix for a reference image)**

Figure 12-16 shows the LiteOS framework, that is, the capabilities provided by different middleware. The framework, which is included in the LiteOS SDK, is the overall architecture of the LiteOS operating system. The aforementioned kernel is at the bottom of the architecture diagram, with the SDK in the middle, and user applications at the top. The SDK contains many components, which can be divided into two parts. The first part is the interconnection component, which is responsible for interworking with the cloud. A key function of IoT is to implement interconnection and interworking between IoT terminals and networks. However, some current IoT terminal systems are still unable to connect to the Internet due to poorly developed interconnection components.

The second part is enhancement components, which are mainly developed for various industry partners. Different capabilities are customized for them based on their requirements. Next, this section will give a brief description of some important components.

First of all, it is about the device-cloud interworking component. One of the advantages of this component is that it can be integrated into various communication modules to enable IoT terminals with limited resources in order to communicate with the cloud. The communication modules include NB-IoT, eMTC, and Wi-Fi modules. In addition, the device-cloud interworking component provides device-cloud collaboration, and integrates a full set of IoT interconnection protocol stacks. As a result, users only need to focus on their own applications, and the implementation of other parts will be completed in our operating system. This allows users to reduce development cycles, focus on the implementation of their own applications, and quickly build products.

In addition to the interconnection framework, we also provide users with complete protocol stacks of IP, TCP/UDP, and CoAP, which include mesh networking capabilities. The mesh network connects applications through the CoAP protocol, whereby an interconnection framework can implement interconnection and interworking between terminals that use different protocols.

While the interconnection framework is mainly used to interconnect sensors that use different protocols, the sensing framework is used to manage different sensors, which also use different data transmission protocols. For example, temperature and humidity sensors may use the SPI, and light sensors may use the IIC. This requires users to compile code by themselves. However, the sensing framework helps users manage data through the use of unified driver interfaces, interaction management, and sensing algorithms. This approach transforms the raw data into various application data, such as that used for step counting, heart rate detection, and environmental monitoring.

Finally, it comes to the security framework. The security framework can be divided into three categories based on things, connections, and networks. IoT terminal security is related to things, and is similar to the security management of mobile phones, including key management and secure storage. This is equivalent to the storage of fingerprints and passwords. Transmission security refers to the security of connections between things and the network layer. And lastly, the IoT networks relate to device-cloud security during connection to the cloud, including authentication and authorization on the cloud as well as device authentication.

### 12.3.4 Huawei LiteOS APIs

Huawei LiteOS APIs are the APIs required for interconnecting the OS with the upper layer. As we have already mentioned, Huawei's vision for the IoT field involves the creation of an ecosystem that is open to all, and within which we can work together to build various IoT applications. In addition, this OS provides open APIs. We aim to enable application developers to focus only on their own application development while we take care of other elements, thereby shortening development cycles. LiteOS is also extremely compatible with Linux, allowing developers who are familiar with such systems to create brand new applications with ease.

# 13 AT Commands of the Communication Module

This section describes AT commands related to experiments. When a developer uses terminals with separate MCUs and communication modules, the MCUs will connect to the communication modules through AT commands first and then connect to the cloud platform.

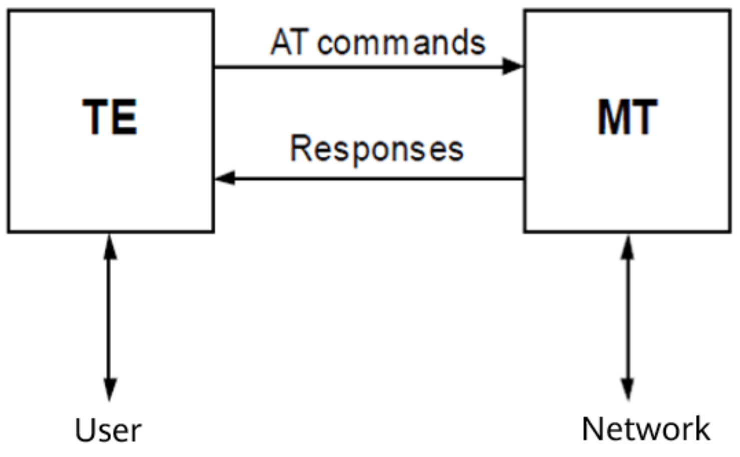


**Figure 13-1 Working mode of AT commands**

The key point is AT commands. An AT command is used for connection and communications between terminal equipment (TE) and PC applications. Generally, an AT command is an instruction set, which is similar to the instruction set used at the sensing layer of a single-chip microcomputer. This instruction set is used to control the communications module. Using AT commands, we can easily control devices in different ways.

## 13.2 Introduction to AT Commands

### 13.2.1 AT Command Types

**Table 13-1 AT command types**

| Category | Syntax | Example |
|---|---|---|
| Set command | AT+<x>=P1 | AT+NNMI=1 |
| Test command | AT+<x>=? | AT+CMEE=? |
| Read command | AT+<x>? | AT+CMEE? |
| Execution command (with parameters) | AT+<x>=<...> | AT+CMEE=0 |
| Execution command (without parameters) | AT+<x> | AT+NRB |

There are four types of AT commands: Set, Test, Read, and Execution. These commands are easy to understand. For example, the AT+CMEE command is used in the NB-IoT module to report errors. This command has two parameters: 0 and 1. The parameter 1 indicates that the module returns a specific error message when an error occurs, and 0 indicates that the module returns ERROR when an error occurs. So if you want to use it as a Test command, use AT+CMEE=? If you want to use it as a Read command, use AT+CMEE? When you run a command with a parameter, it is a Set command. For example, AT+CMEE=0 is used to set the module to report ERROR (with no detailed error information) upon an error.

There are commands without parameters, for example, AT+NRB. This command is used in the NB-IoT module to restart the entire module. The Set command is similar to the Execution command. The difference is that one has parameters and the other does not.

### 13.2.2 NB-IoT AT Commands

**Table 13-2 Common NB-IoT AT commands**

| Purpose | AT Command |
|---|---|
| Disabling a function | AT+CFUN=0 |
| Checking the software version | AT+CGMR |
| Querying the international mobile equipment identity (IMEI) | AT+CGSN=1 |
| Setting the platform address | AT+NCDP=xx.xx.xx.xx |
| Configuring an access point | AT+CGDCONT=1,"IP","xxxx" |

| Purpose | AT Command |
|---|---|
| name (APN) | |
| Restarting the module | AT+NRB |
| Enabling a function | AT+CFUN=1 |
| Querying the IMSI of a SIM card | AT+CIMI |
| Notifying the terminal of connecting to the base station | AT+CSCON=1 |
| Notifying the terminal of connecting to the core network | AT+CEREG=2 |
| Notifying the terminal of downlink data transmission | AT+NNMI=1 |
| Notifying the terminal of successful data transmission | AT+NSMI=1 |
| Attaching to a network | AT+CGATT=1 |
| Querying the terminal status | AT+NUESTATS |
| Querying the IP address assigned by the core network | AT+CGPADDR |
| Transmitting data | AT+NMGS=1,11 |
| Querying the sending buffer | AT+NQMGS |
| Querying the receiving buffer | AT+NQMGR |

The MCU uses AT commands to control the communication module. Terminal manufacturers must develop software that invokes AT commands to control communication modules in addition to developing corresponding service functions.

### 13.2.3 Wi-Fi AT Commands

**Table 13-3 Common Wi-Fi AT commands**

| Purpose | AT Command |
|---|---|
| Restarting the module | AT+RST |
| Querying the version | AT+GMR |
| Scanning nearby access points (APs) | AT+CWLAP |
| Connecting to an AP | AT+CWJAP |

| Purpose | AT Command |
|---|---|
| Disconnecting from an AP | AT+CWQAP |
| Querying connection information | AT+CIPSTATUS |
| Resolving the domain name | AT+CIPDOMAIN |
| Establishing a connection | AT+CIPSTART |
| Starting transparent transmission | AT+CIPMODE |
| Transmitting data | AT+CIPSEND |
| Querying the local IP address | AT+CIFSR |
| Using the ping operation | AT+PING |
| Scanning available APs | AT+CWLAP |
| Restoring factory settings | AT+RESTORE |
| Querying the available memory space of the system | AT+SYSRAM |

Table 13-3 lists the common AT commands for Wi-Fi modules. Different modules have different AT commands, because networks have different usage and functions. Therefore, some commands are the same, and others are different. For example, GPRS and NB-IoT are cellular network technologies defined in 3GPP specifications, and their AT commands have many similarities. However, Wi-Fi is not a communications technology defined in 3GPP specifications, and its AT commands are quite different from those of NB-IoT. Most commands are used to interact with the gateway and access the network through the gateway. The Wi-Fi module does not involve data such as SIM card and core network data. Therefore, its AT commands are different.

### 13.2.4 AT Commands for Huawei-Certified Modules

**Table 13-4 Common AT commands for Huawei-certified modules**

| Purpose | AT Command |
|---|---|
| Obtaining the Huawei SDK version | AT+HMVER |
| Setting MQTT connection parameters | AT+HMCON |
| Disconnecting from the HUAWEI CLOUD IoT platform | AT+HMDIS |
| Sending MQTT data to a topic | AT+HMPUB |

| Purpose | AT Command |
|---|---|
| Transmitting data received by the module to an external MCU | +HMREC |
| Transmitting the module connection or disconnection status to an external MCU | +HMSTS |
| Subscribing to a custom topic | AT+HMSUB |
| Unsubscribing from a custom topic | AT+HMUNS |
| Setting a server or client certificate | AT+HMPKS |

In addition to the AT commands for NB-IoT and Wi-Fi modules, some Huawei-certified modules have AT commands. These modules have passed compatibility certification. The AT commands and format specifications are basically the same as Huawei's general requirements.

## 13.3 Terminal-side Development for Communication Modules

### 13.3.1 NB-IoT Terminal Interconnection Process

The process of connecting an NB-IoT terminal to the IoT platform is as follows:

1. Power on the terminal and run the **AT+NRB** command to reset the terminal. If **OK** is returned, the terminal is running properly.
2. Run the **AT+NTSETID=1,*Device_ID*** command to specify the device ID. The device ID is the terminal IMEI. If the command is executed, **OK** is returned.
3. Run the **AT+NCDP=*IP*,*Port*** command to set the IP address and port for connecting to the IoT platform. The port is 5683. If the command is executed, **OK** is returned.
4. Run the **AT+CFUN=1** command to enable the network access function. If the command is executed, OK is returned.
5. Run the **AT+NBAND=*Frequency_band*** command to specify the frequency band. If the command is executed, **OK** is returned.
6. Run the **AT+CGDCONT=1,"*IP*","*APN*"** command to set the IoT core APN. If the operation is successful, **OK** is returned. To obtain the APN, contact the carrier or OpenLab owner.
7. Run the **AT+CGATT=1** command to connect the terminal to the network. If the command is executed, **OK** is returned.
8. Run the **AT+CGPADDR** command to check whether the terminal has obtained the IP address assigned by the IoT core network. If it has, the terminal has accessed the network.

9. Run the **AT+NMGS=*Data_length*,*Data*** command to enable the terminal to send upstream data. If the upstream data is sent, **OK** is returned.
10. If the IoT platform sends downstream data to the terminal, obtain downstream data by running the **AT+NMGR** command.

NB-IoT is an operator network. Therefore, the connection process of NB-IoT modules or devices is similar to that of mobile phones. The NB-IoT network also uses SIM cards for access.

First, you need to perform initialization, such as terminal reset. Then enter the device ID and platform port, and enable the NB-IoT function. You do not need to set the frequency band and core network when an automatic network is used. Run the **AT+CGATT** command to connect the terminal to the network. So far, the connection between the terminal and platform is established. The subsequent steps are related to data transmission.

### 13.3.2 Wi-Fi Terminal Interconnection Process

The process of connecting a Wi-Fi terminal to the IoT platform is as follows:

1. Power on the terminal, and run the **AT+CWMODE=3** command to reset the terminal. If **OK** is returned, the Wi-Fi mode has been configured on the terminal.
2. Run the **AT+CWJAP=*SSID*,*Password*** command to connect to the router. If the command is executed, **OK** is returned.
3. Run the **AT+CIFSR** command to query the IP address of the ESP8266. If the command is executed, **OK** is returned.
4. Run the **AT+CIPSTART=*TCP*,*IP*,*PORT*** command to set the IP address and port for connecting to the IoT platform. The port is 5683. If the command is executed, **OK** is returned.
5. Run the **AT+CIPSEND=<length>** command to send data. After **>** is returned, input the data. If the command is executed, **SEND OK** is returned.

The connection process for Wi-Fi modules is similar to that for NB-IoT modules, except the Wi-Fi module does not require a SIM card. Therefore, you only need to enter the Wi-Fi name and password as well as the IP address and port of the IoT platform to connect.

# 14 Conclusions

Now you have finished the HCIA-IoT course. Congratulations! This course provides a brief introduction to the basics of the IoT. There are many other IoT technologies and knowledge that are not covered in this course. For example, in addition to the basic knowledge about the four-layer IoT architecture described in this book, there is also knowledge related to development at the sensing, network, and platform layers, such as basic experiments of single-chip microcomputers at the sensing layer, development experiments at the platform layer, and development of communication chips at the network layer. All these will be introduced in HCIP-IoT V2.5. The HCIP-IoT course will also introduce other HUAWEI CLOUD services, such as big data and AI. You are welcome to learn HCIP-IoT.

The following resources are listed, hoping you can learn more from them:

- Huawei Certification: https://e.huawei.com/en/talent/#/cert?navType=authNavKey

- Huawei e-Learning: https://ilearningx.huawei.com/portal/subportal/EBG/51

- Huawei ICT Academy: https://www.huaweiacad.com/webapps/hw-toboo-BBLEARN/portal/exec.do?t=1561101910908

Though we strive for accuracy, information in this Guide may contain inadvertent inaccuracies or even errors. Contact us if you have any questions or suggestions.

# 15 References

Andrew S. Tanenbaum and Herbert Bos. *Modern Operating Systems (4th Edition)*. China Machine Press, 2017.

Huang Yan and Yang Lin. *Huawei Cloud IoT Platform Technology and Practice*. Posts & Telecom Press Co., Ltd., 2020.

Ding Fei, Zhang Dengyin, and Cheng Chunmao. *Introduction to Internet of Things*. Posts & Telecom Press Co., Ltd., 2020.

Kong Linghe, Li Xuefeng, and Chai Fangming. *IoT Operating System Principle (LiteOS)*. Posts & Telecom Press Co., Ltd., 2020.

# 16 Appendix

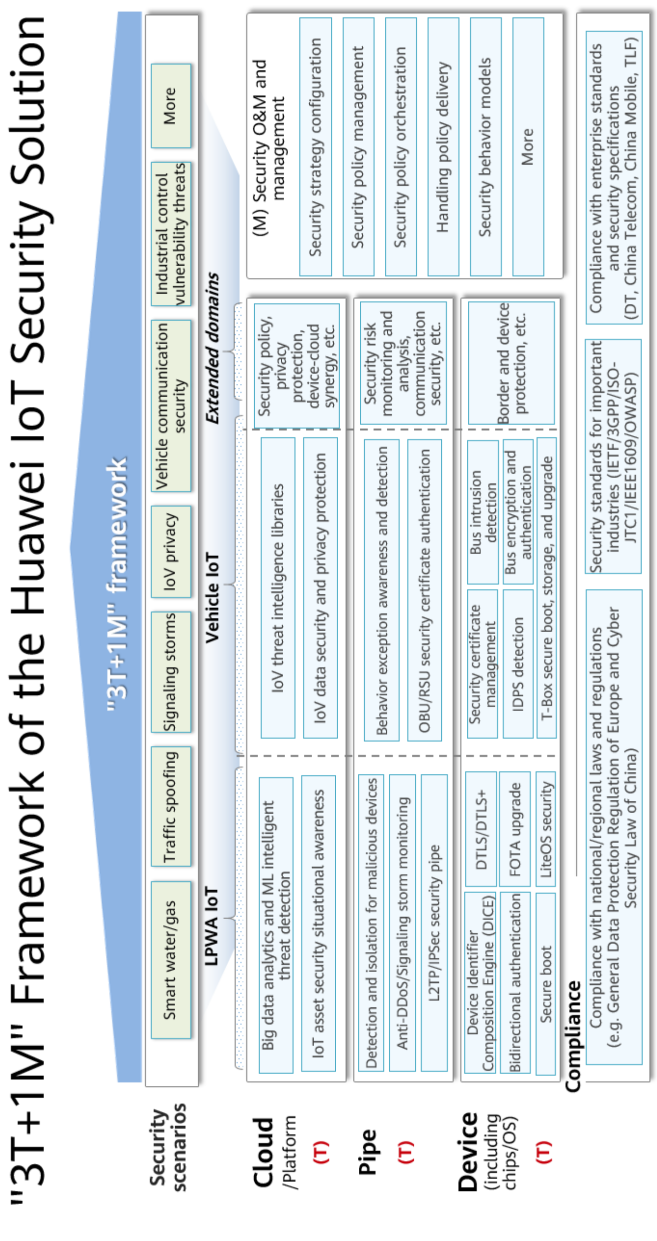
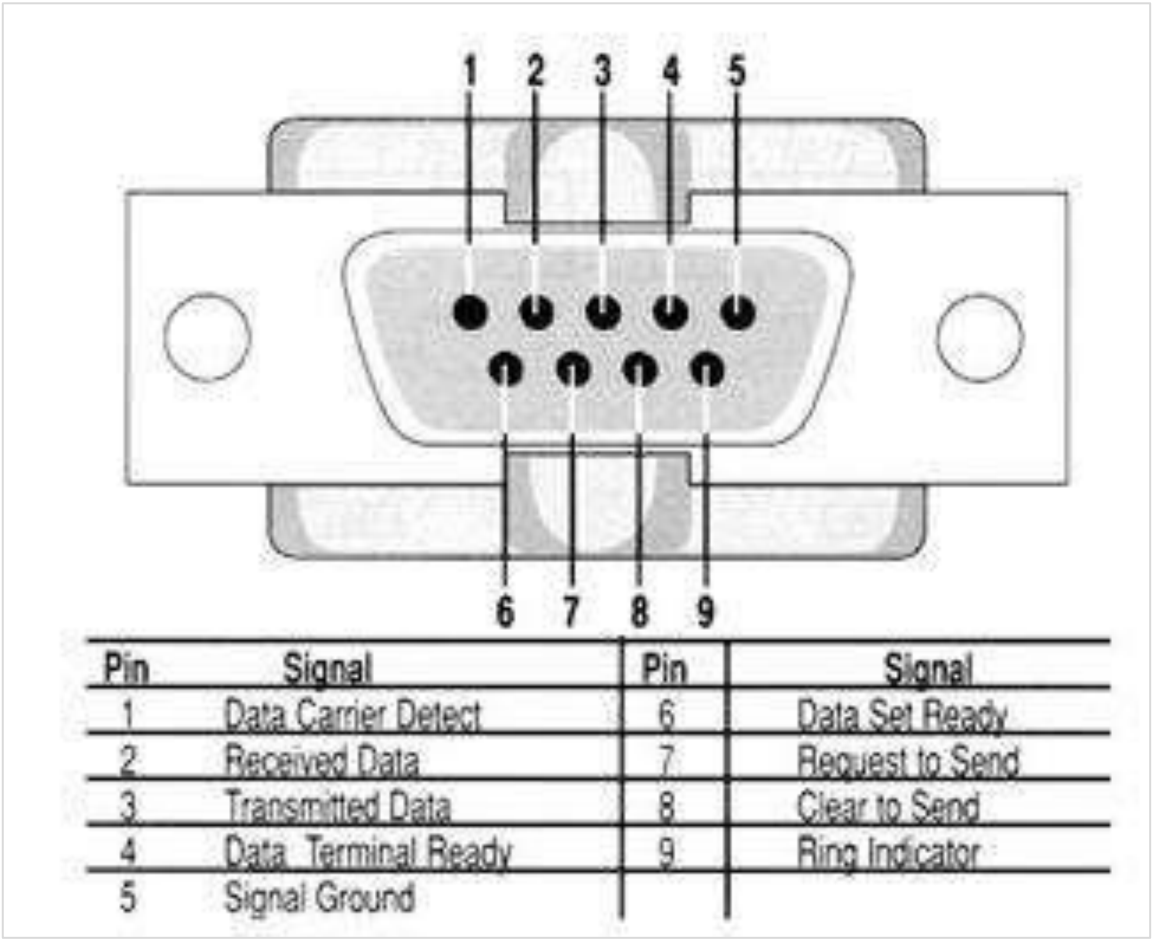

"3T+1M" Framework of the Huawei IoT Security Solution
"3T+1M" framework
Security scenarios
Smart water/gas
Traffic spoofing
Signaling storms
IoV privacy
Vehicle communication security
Industrial control vulnerability threats
More
LPWA IoT
Vehicle IoT
Extended domains
Cloud /Platform (T)
Big data analytics and ML intelligent threat detection
IoT asset security situational awareness
IoV threat intelligence libraries
IoV data security and privacy protection
Security policy, privacy protection, device-cloud synergy, etc.
Pipe (T)
Detection and isolation for malicious devices
Anti-DDoS/Signaling storm monitoring
L2TP/IPSec security pipe
Behavior exception awareness and detection
OBU/RSU security certificate authentication
Security risk monitoring and analysis, communication security, etc.
Device (including chips/OS) (T)
Device Identifier Composition Engine (DICE)
DTLS/DTLS+
Bidirectional authentication
FOTA upgrade
Secure boot
LiteOS security
Security certificate management
Bus intrusion detection
IDPS detection
Bus encryption and authentication
T-Box secure boot, storage, and upgrade
Border and device protection, etc.
(M) Security O&M and management
Security strategy configuration
Security policy management
Security policy orchestration
Handling policy delivery
Security behavior models
More
Compliance
Compliance with national/regional laws and regulations (e.g. General Data Protection Regulation of Europe and Cyber Security Law of China)
Security standards for important industries (IETF/3GPP/ISO-JTC1/IEEE1609/OWASP)
Compliance with enterprise standards and security specifications (DT, China Telecom, China Mobile, TLF)

**Figure 16-1 IoT security solution architecture**



| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 1 | Data Carrier Detect | 6 | Data Set Ready |
| 2 | Received Data | 7 | Request to Send |
| 3 | Transmitted Data | 8 | Clear to Send |
| 4 | Data Terminal Ready | 9 | Ring Indicator |
| 5 | Signal Ground | | |

**Figure 16-2 RS-232 port diagram**

**Table 16-1 RS-232 port**

| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 1 | Data Carrier Defect | 6 | Data Set Ready |
| 2 | Received Data | 7 | Request to Send |
| 3 | Transmitted Data | 8 | Clear to Send |
| 4 | Data Terminal Ready | 9 | Ring Indicator |
| 5 | Signal Ground | | |

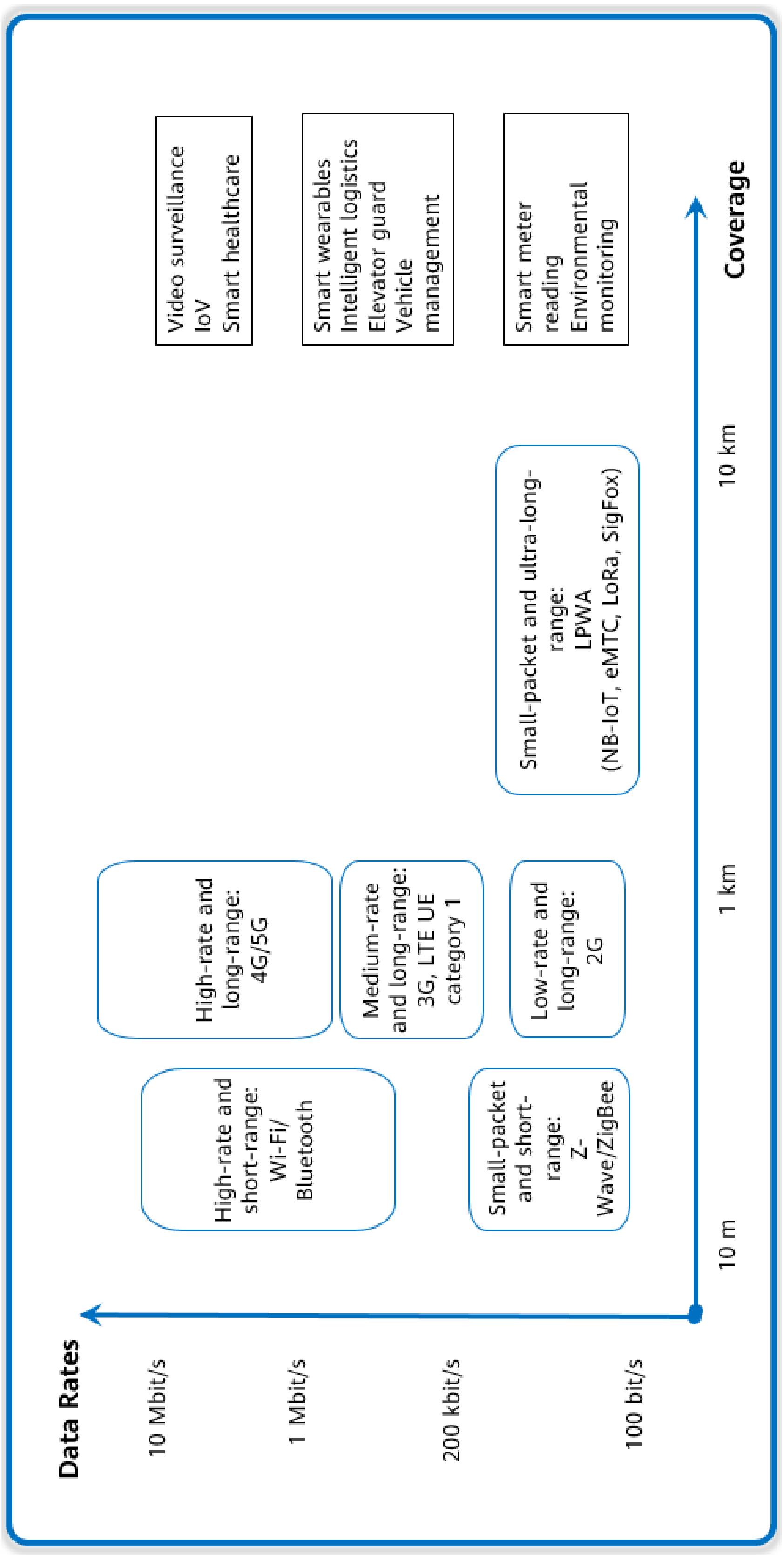

Data Rates
10 Mbit/s
1 Mbit/s
200 kbit/s
100 bit/s
Coverage
10 m
1 km
10 km
High-rate and short-range: Wi-Fi/ Bluetooth
Small-packet and short-range: Z-Wave/ZigBee
High-rate and long-range: 4G/5G
Medium-rate and long-range: 3G, LTE UE category 1
Low-rate and long-range: 2G
Small-packet and ultra-long-range: LPWA (NB-IoT, eMTC, LoRa, SigFox)
Video surveillance
IoV
Smart healthcare
Smart wearables
Intelligent logistics
Elevator guard
Vehicle management
Smart meter reading
Environmental monitoring

**Figure 16-3 Comparison of wireless communications technologies**

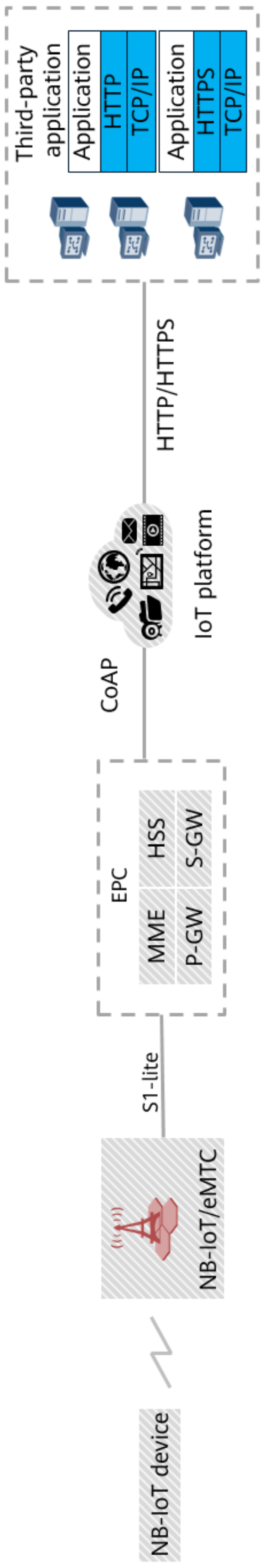
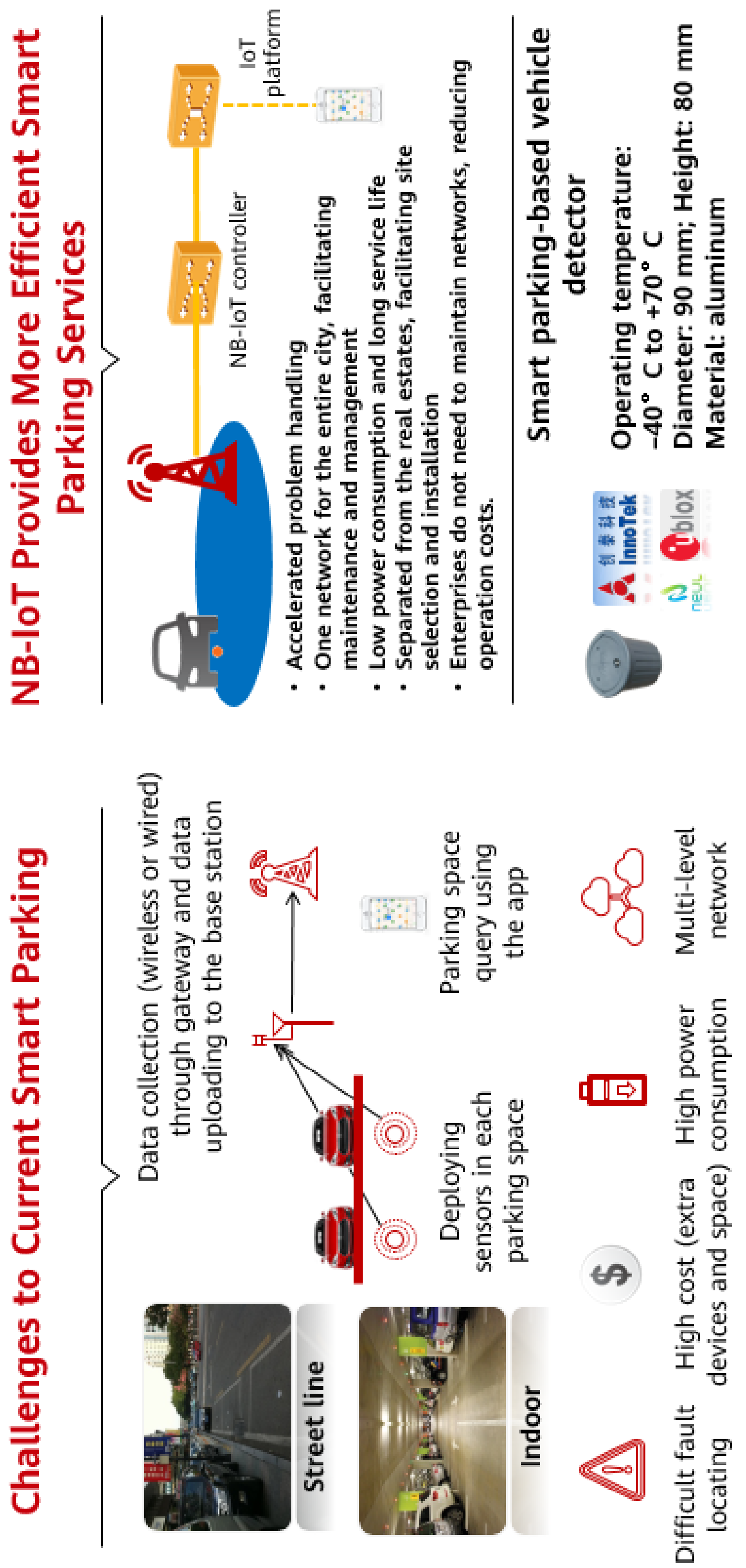


**Figure 16-4 NB-IoT solution architecture**



Challenges to Current Smart Parking

Street line

Indoor

Data collection (wireless or wired) through gateway and data uploading to the base station

Deploying sensors in each parking space

Parking space query using the app

Difficult fault locating

High cost (extra devices and space)

High power consumption

Multi-level network

NB-IoT Provides More Efficient Smart Parking Services

NB-IoT controller

IoT platform

- Accelerated problem handling
- One network for the entire city, facilitating maintenance and management
- Low power consumption and long service life
- Separated from the real estates, facilitating site selection and installation
- Enterprises do not need to maintain networks, reducing operation costs.

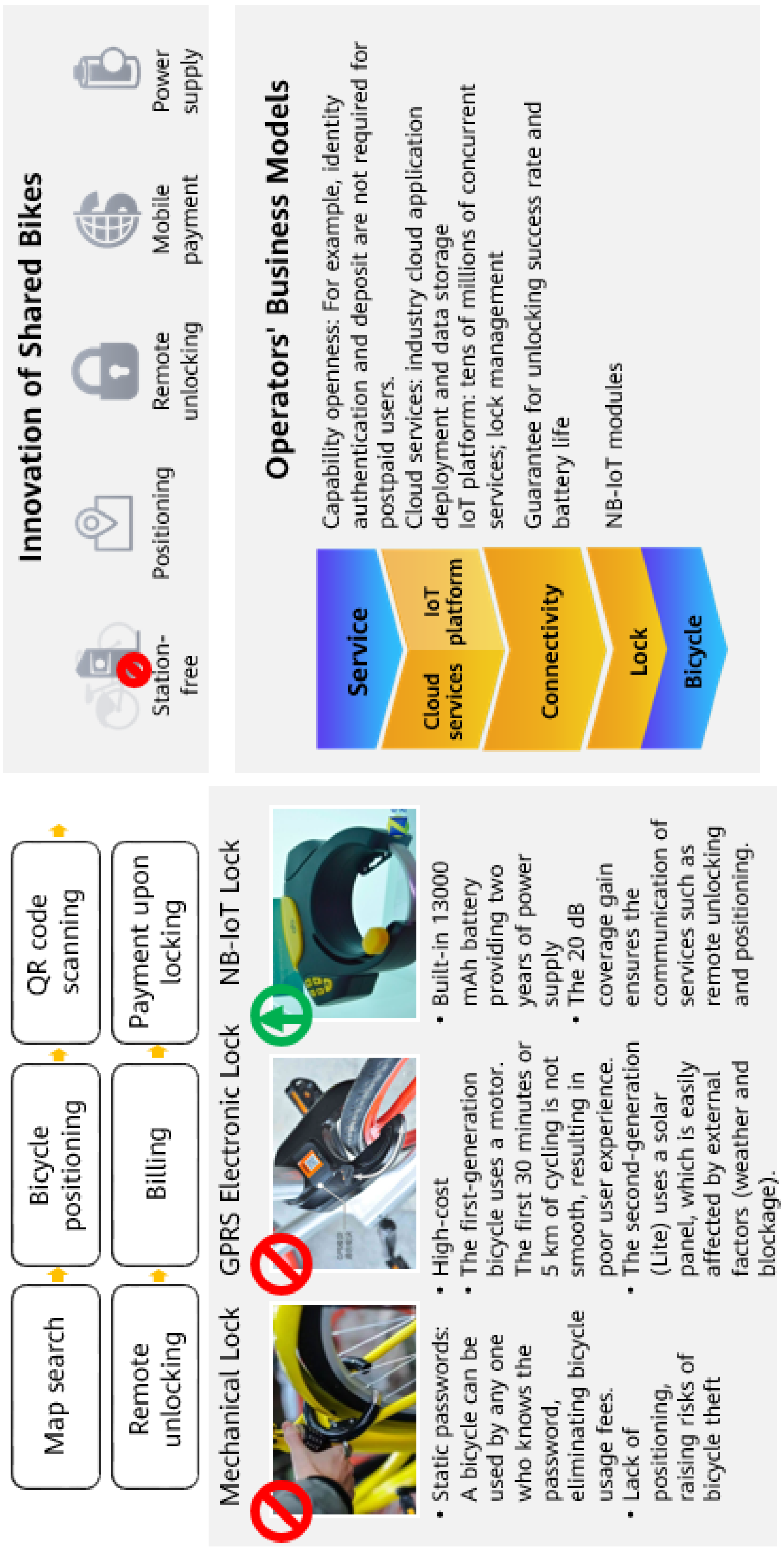
Smart parking-based vehicle detector

Operating temperature: -40° C to +70° C

Diameter: 90 mm; Height: 80 mm

Material: aluminum

Figure 16-5 Smart parking



Mechanical Lock

- Static passwords: A bicycle can be used by any one who knows the password, eliminating bicycle usage fees.
- Lack of positioning, raising risks of bicycle theft

GPRS Electronic Lock

- High-cost
- The first-generation bicycle uses a motor. The first 30 minutes or 5 km of cycling is not smooth, resulting in poor user experience.
- The second-generation (Lite) uses a solar panel, which is easily affected by external factors (weather and blockage).

NB-IoT Lock

- Built-in 13000 mAh battery providing two years of power supply
- The 20 dB coverage gain ensures the communication of services such as remote unlocking and positioning.

Operators' Business Models

Service: Capability openness: For example, identity authentication and deposit are not required for postpaid users.

Cloud services / IoT platform: Cloud services: industry cloud application deployment and data storage. IoT platform: tens of millions of concurrent services; lock management

Connectivity: Guarantee for unlocking success rate and battery life

Lock: NB-IoT modules

Figure 16-6 Shared bikes

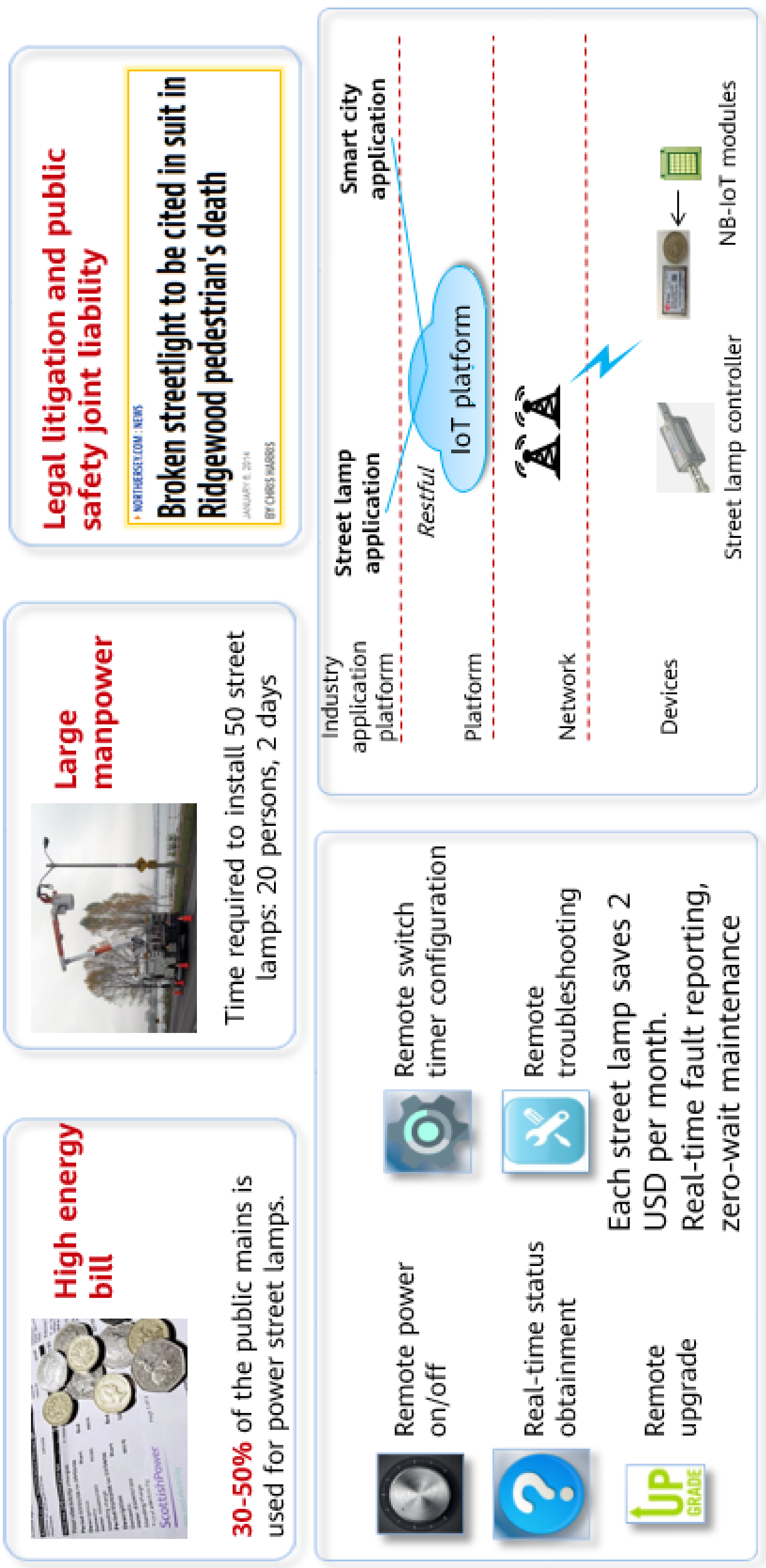

High energy bill
30-50% of the public mains is used for power street lamps.
Large manpower
Time required to install 50 street lamps: 20 persons, 2 days
Legal litigation and public safety joint liability
NORTHJERSEY.COM : NEWS
Broken streetlight to be cited in suit in Ridgewood pedestrian's death
JANUARY 6, 2014
BY CHRIS HARRIS
Remote power on/off
Remote switch timer configuration
Real-time status obtainment
Remote troubleshooting
Remote upgrade
Each street lamp saves 2 USD per month.
Real-time fault reporting, zero-wait maintenance
Industry application platform
Street lamp application
Smart city application
Restful
IoT platform
Platform
Network
Devices
Street lamp controller
NB-IoT modules

**Figure 16-7 Smart street lamps**

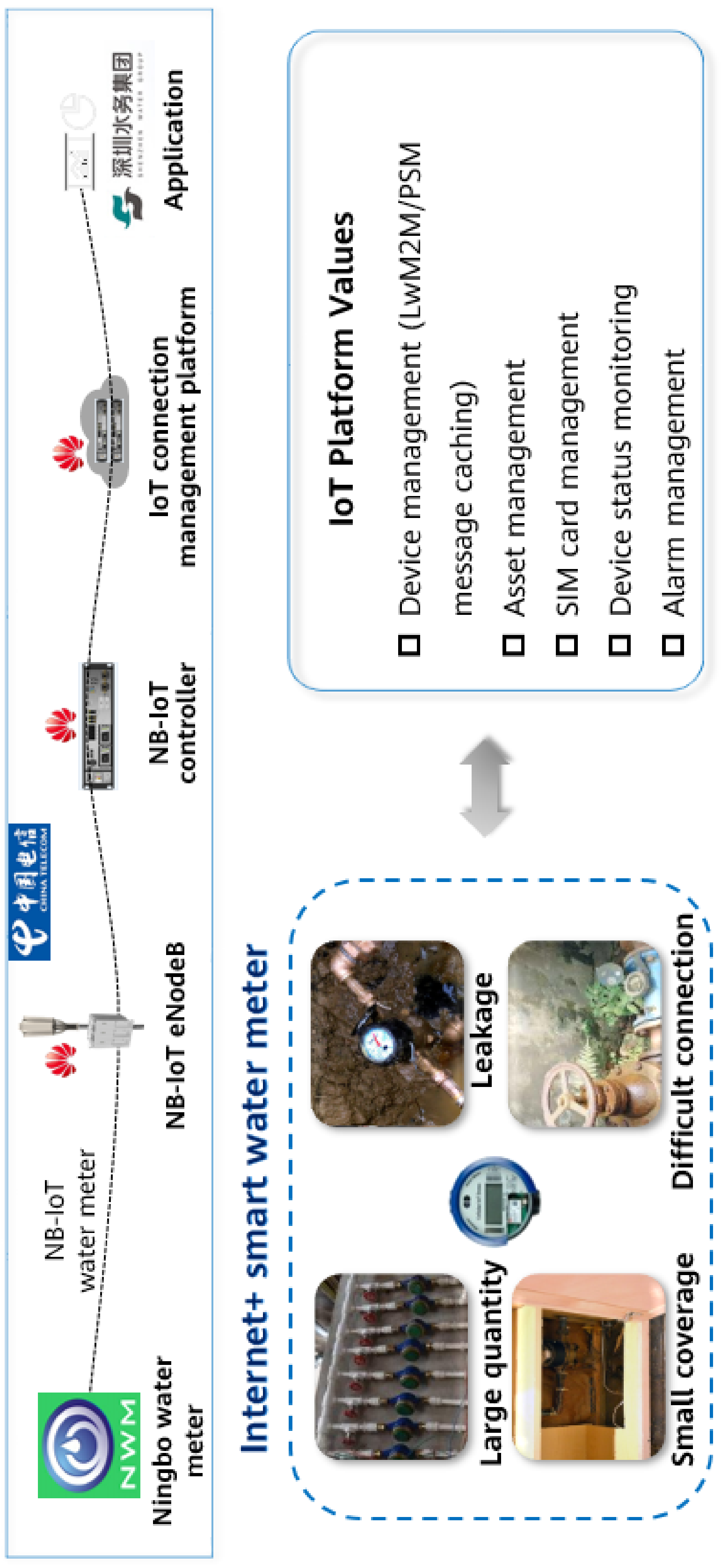


**Figure 16-8 Smart meter reading**

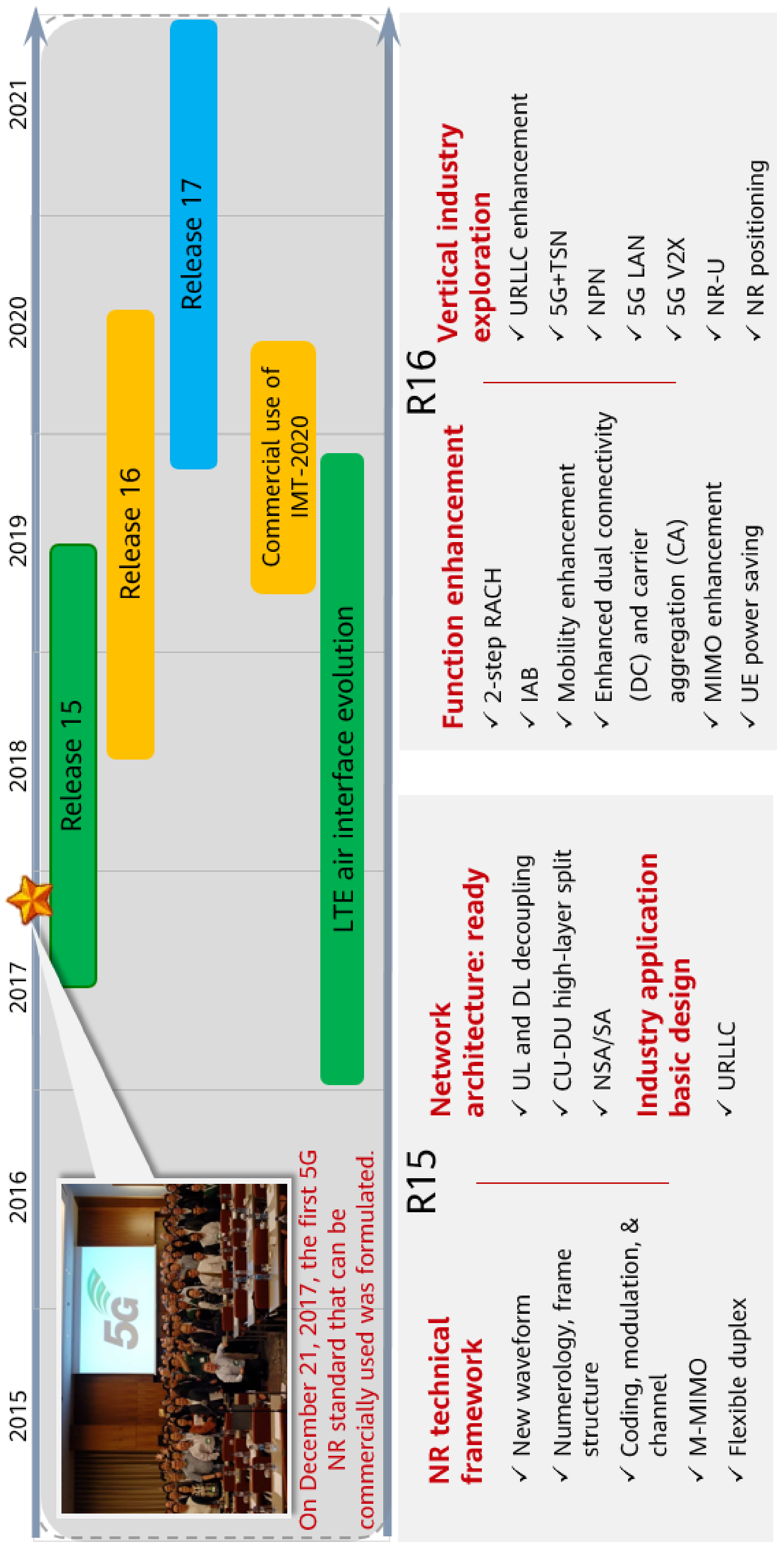

2015
2016
2017
2018
2019
2020
2021
Release 15
Release 16
Release 17
Commercial use of IMT-2020
LTE air interface evolution
On December 21, 2017, the first 5G NR standard that can be commercially used was formulated.
R15
NR technical framework
✓ New waveform
✓ Numerology, frame structure
✓ Coding, modulation, & channel
✓ M-MIMO
✓ Flexible duplex
Network architecture: ready
✓ UL and DL decoupling
✓ CU-DU high-layer split
✓ NSA/SA
Industry application basic design
✓ URLLC
R16
Function enhancement
✓ 2-step RACH
✓ IAB
✓ Mobility enhancement
✓ Enhanced dual connectivity (DC) and carrier aggregation (CA)
✓ MIMO enhancement
✓ UE power saving
Vertical industry exploration
✓ URLLC enhancement
✓ 5G+TSN
✓ NPN
✓ 5G LAN
✓ 5G V2X
✓ NR-U
✓ NR positioning

**Figure 16-9 Evolution of 5G communications technical standards**

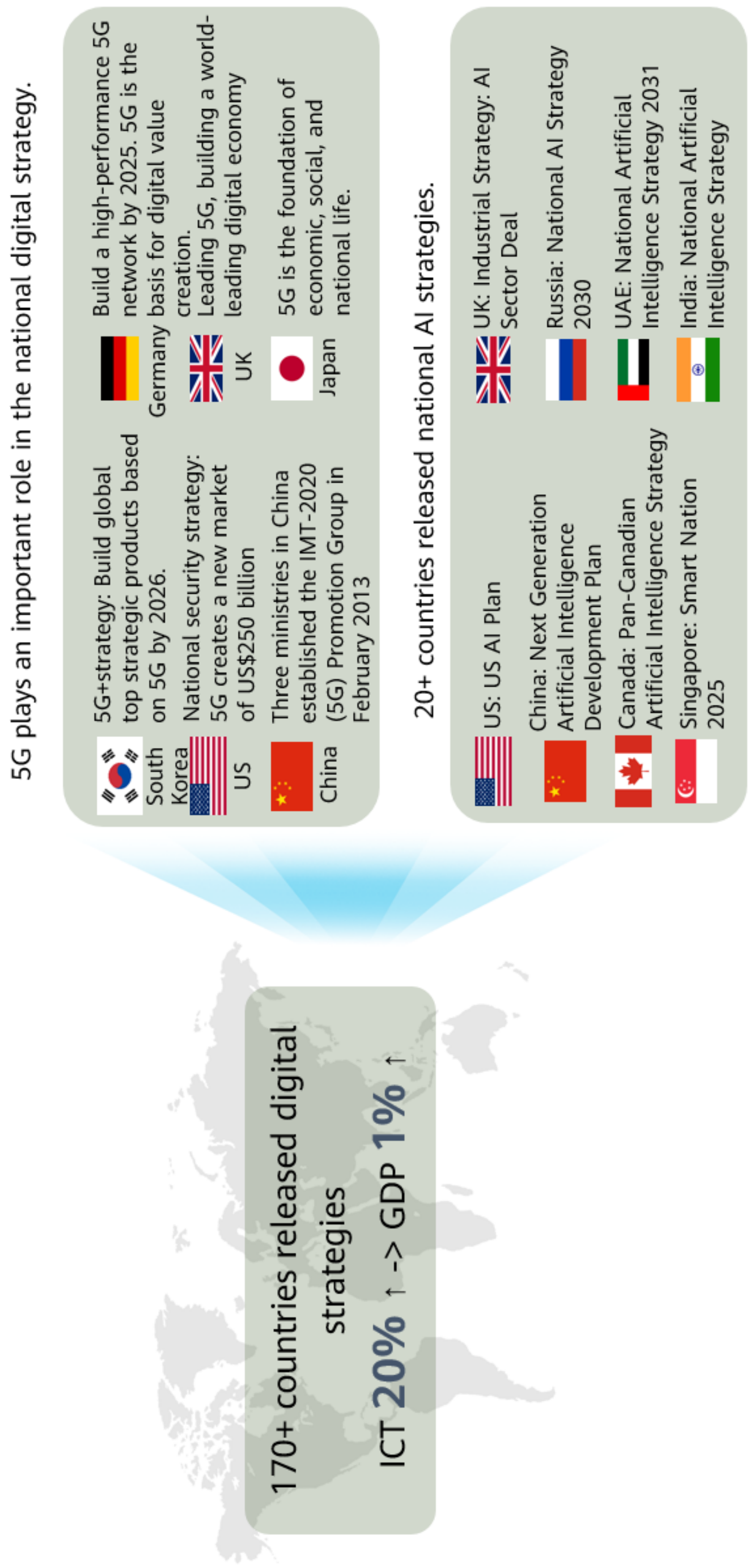

170+ countries released digital strategies
ICT 20% ↑ -> GDP 1% ↑
5G plays an important role in the national digital strategy.
South Korea
5G+strategy: Build global top strategic products based on 5G by 2026.
US
National security strategy: 5G creates a new market of US$250 billion
China
Three ministries in China established the IMT-2020 (5G) Promotion Group in February 2013
Germany
Build a high-performance 5G network by 2025. 5G is the basis for digital value creation.
UK
Leading 5G, building a world-leading digital economy
Japan
5G is the foundation of economic, social, and national life.
20+ countries released national AI strategies.
US: US AI Plan
China: Next Generation Artificial Intelligence Development Plan
Canada: Pan-Canadian Artificial Intelligence Strategy
Singapore: Smart Nation 2025
UK: Industrial Strategy: AI Sector Deal
Russia: National AI Strategy 2030
UAE: National Artificial Intelligence Strategy 2031
India: National Artificial Intelligence Strategy

**Figure 16-10 More countries value 5G & AI as key to national digitization**

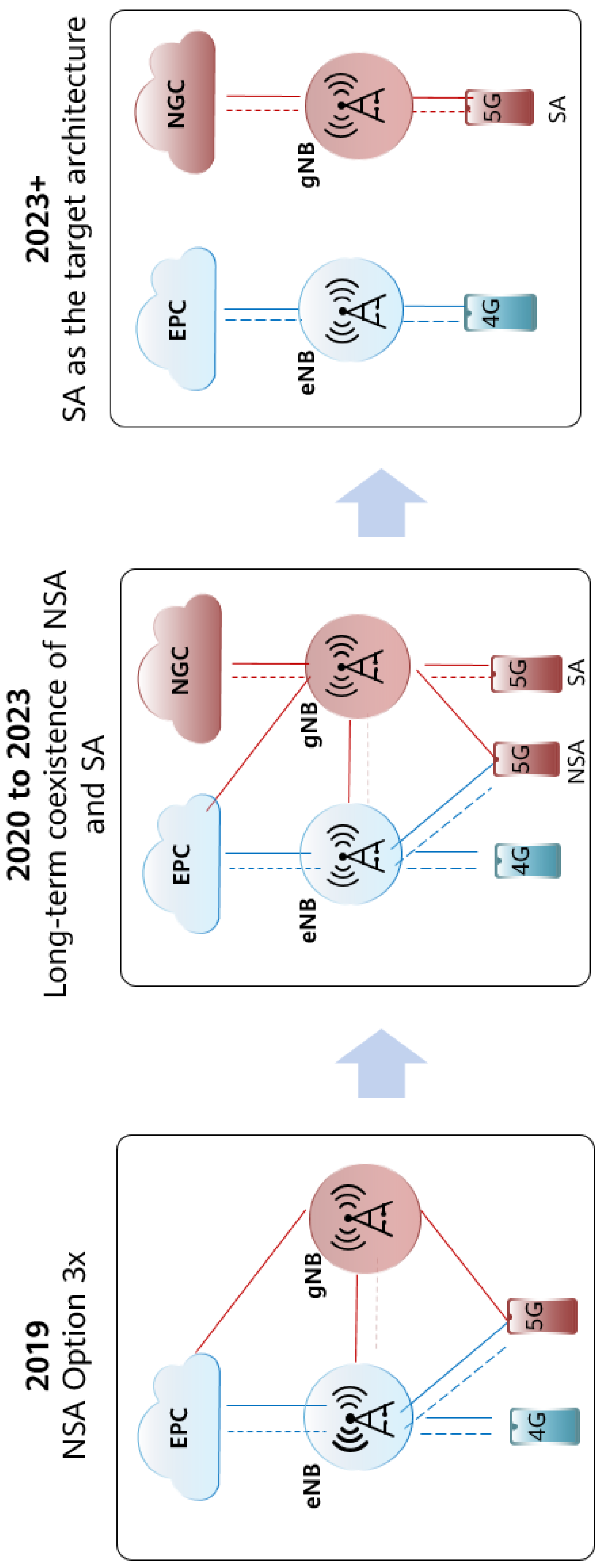


**Figure 16-11 5G networking mode**

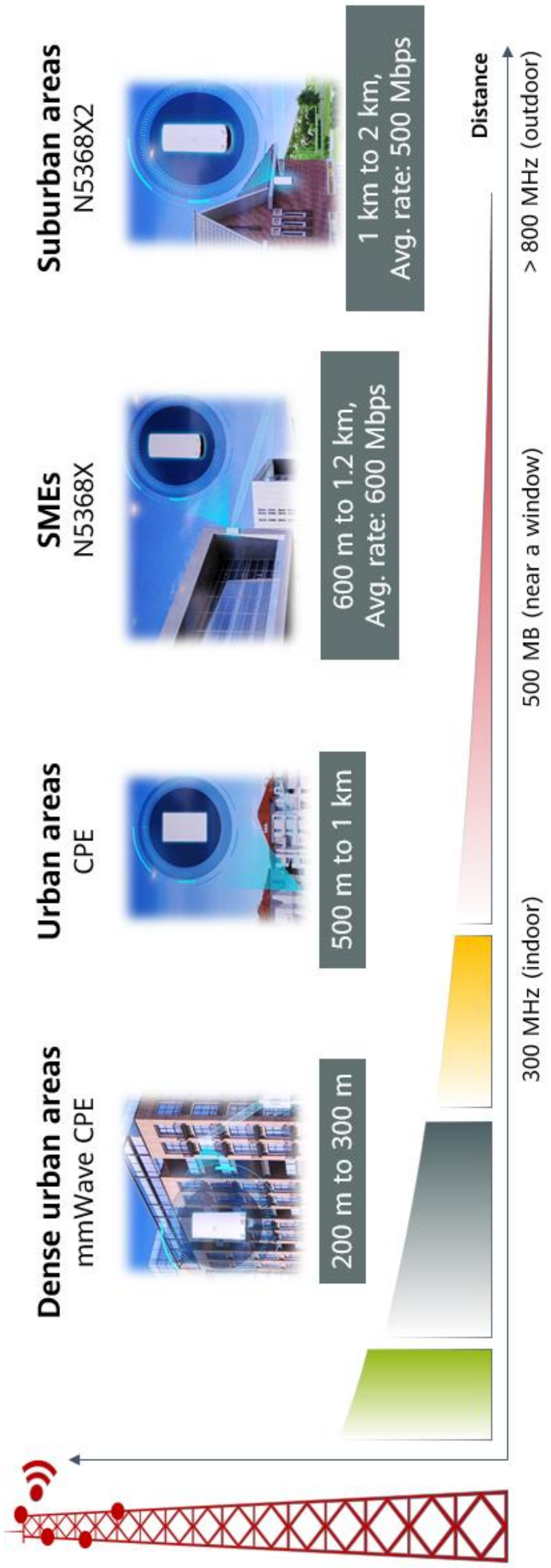


Figure 16-12 FWA solution

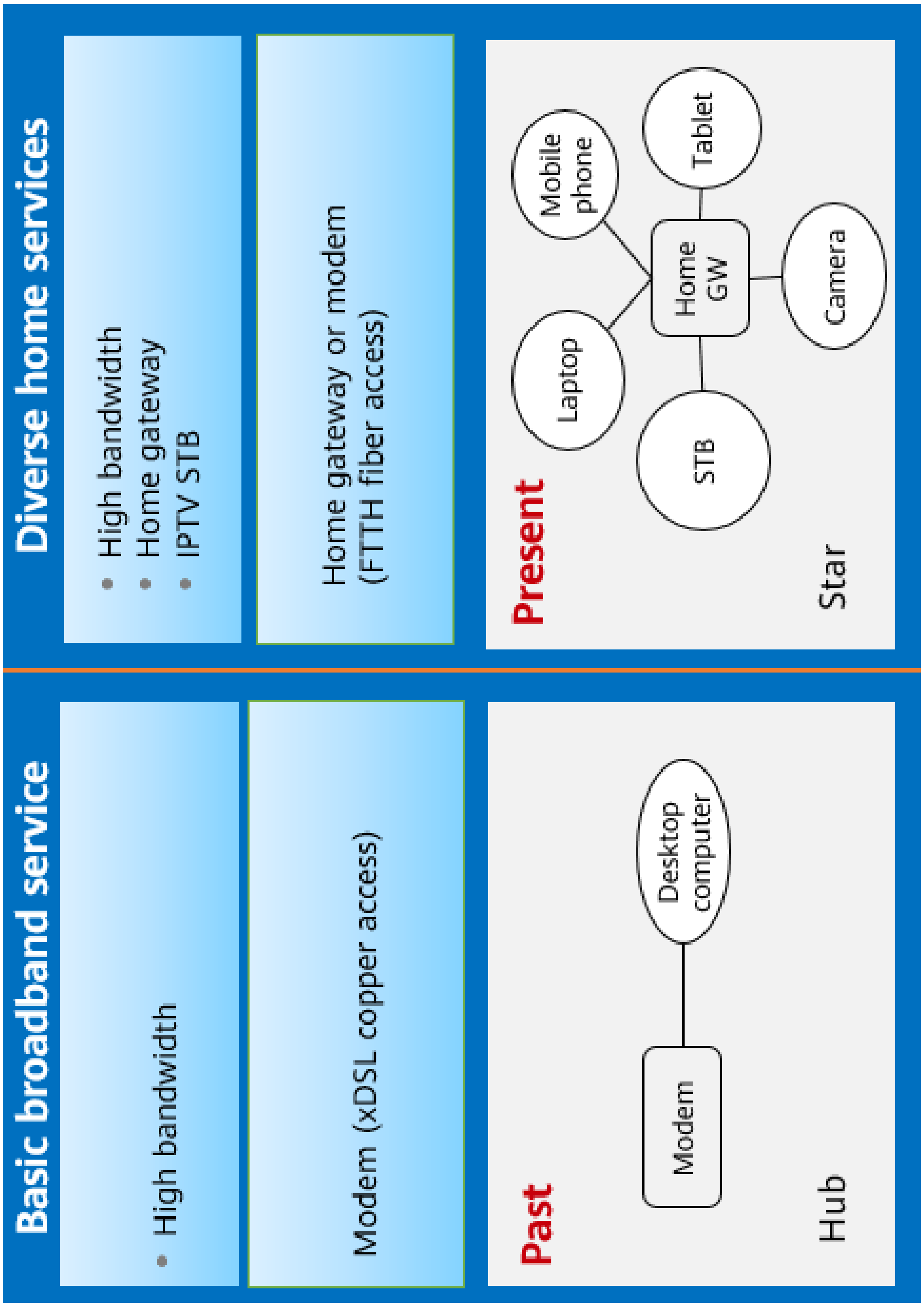


**Figure 16-13 Home network development**

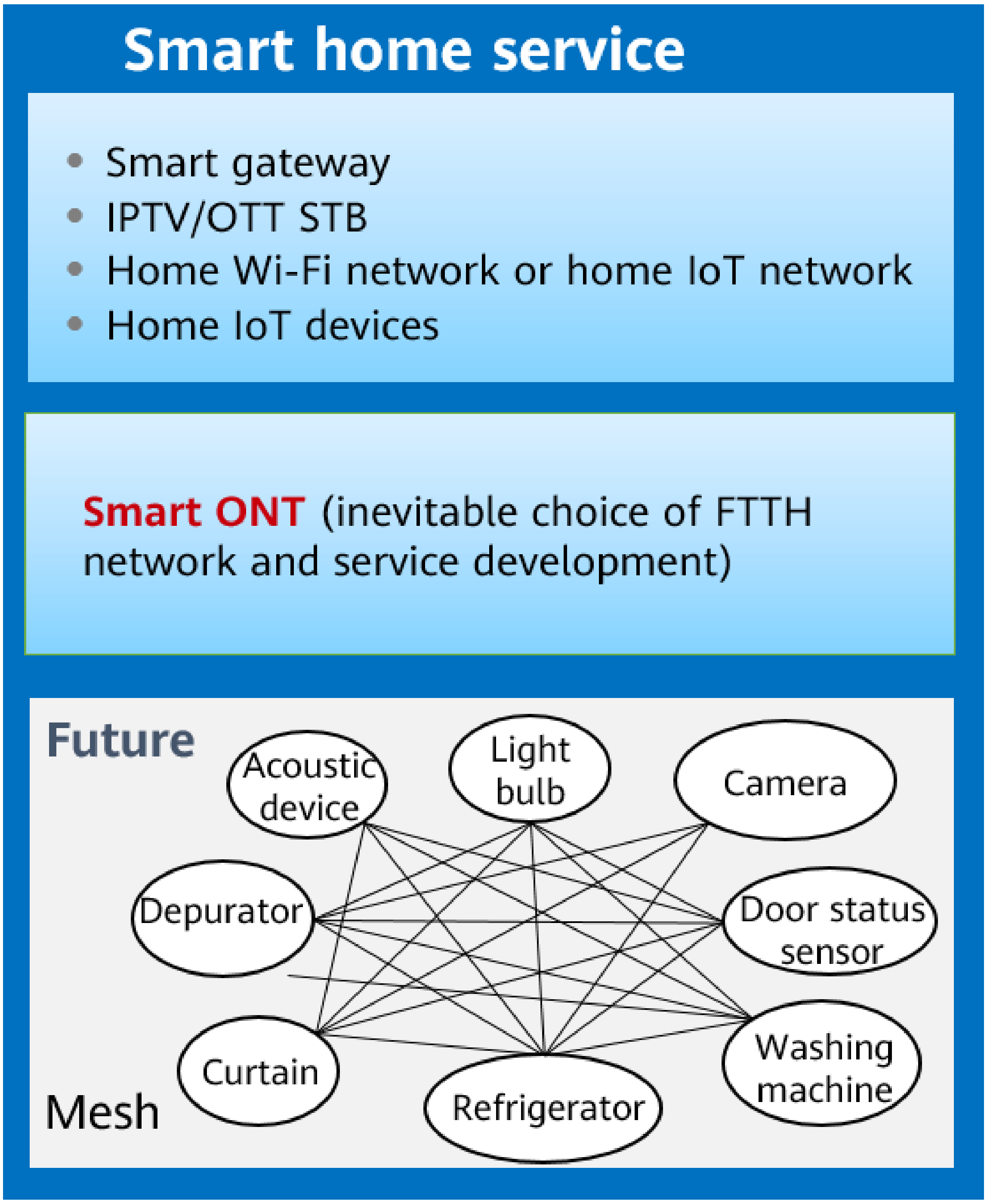


**Figure 16-14 Smart home**

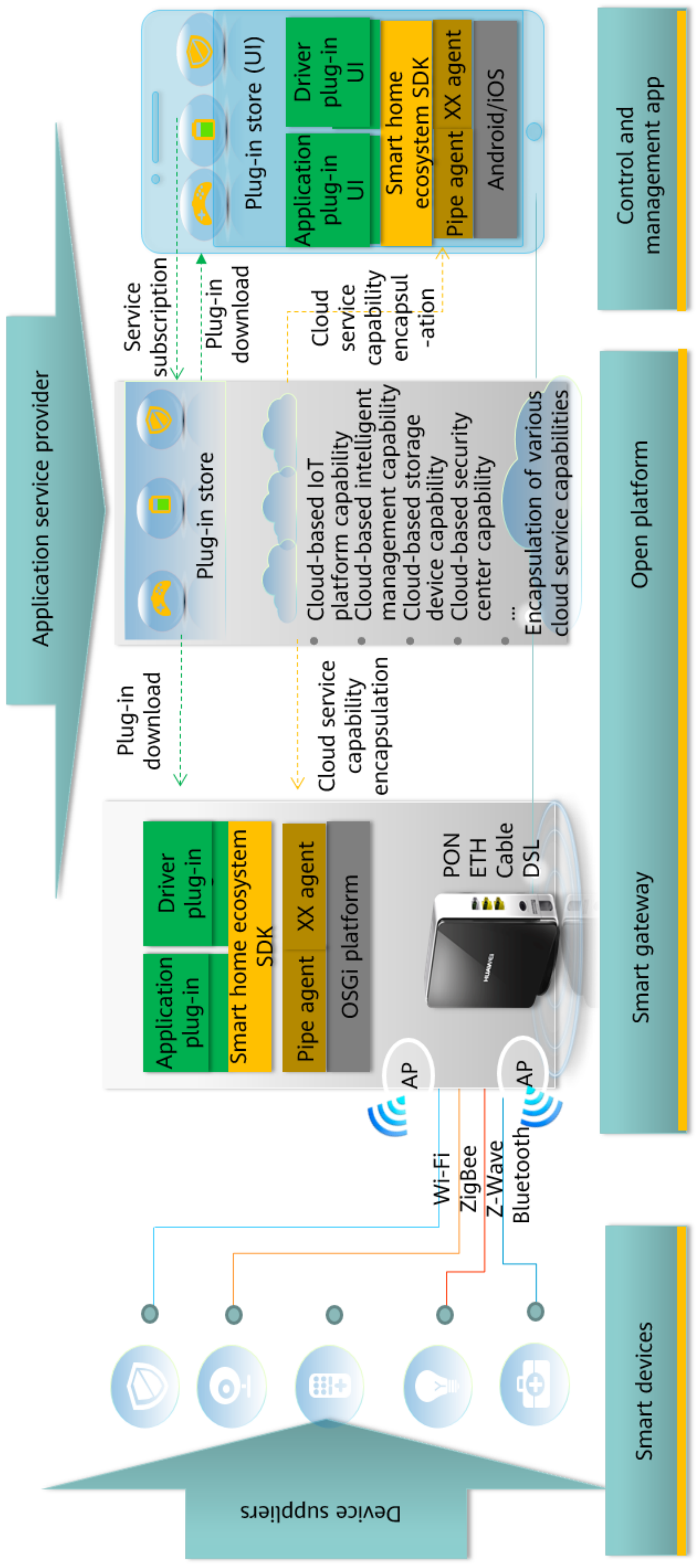


**Figure 16-15 Huawei Smart Home solution**

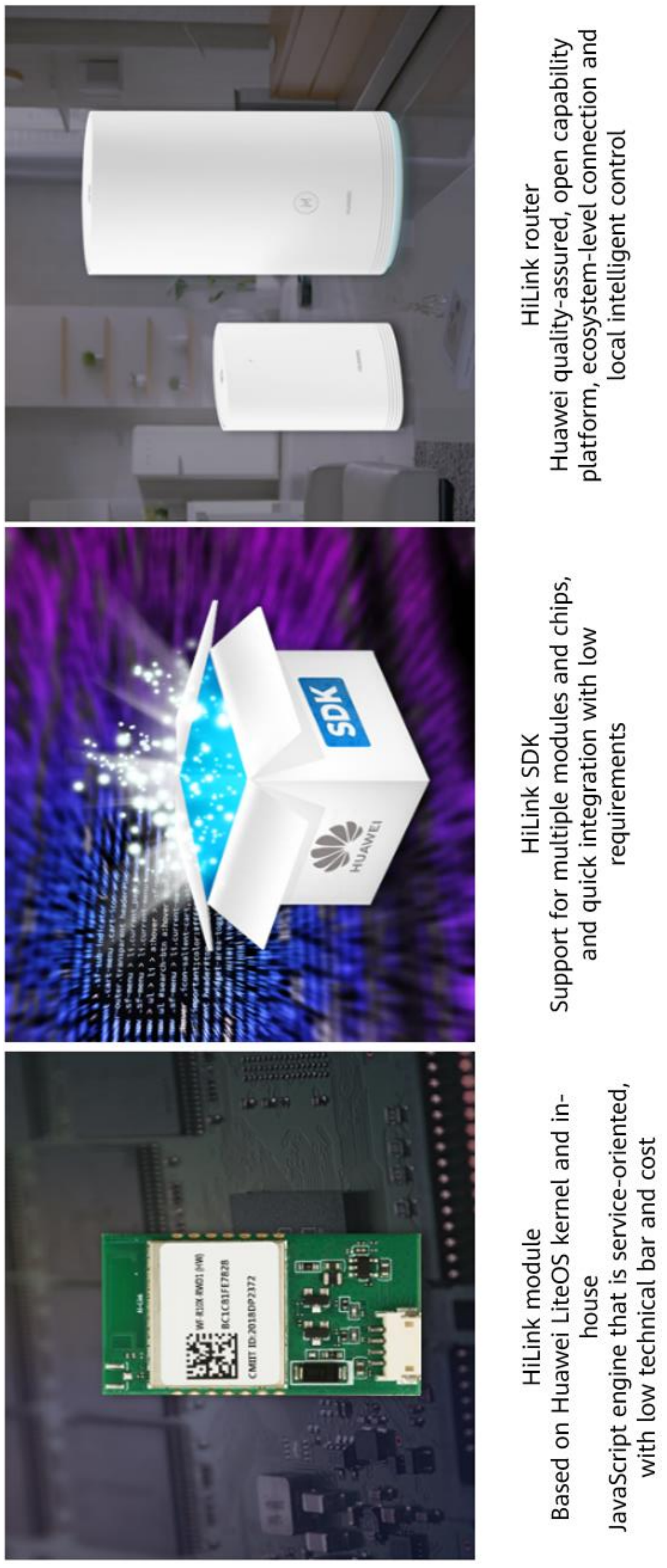

HiLink router
Huawei quality-assured, open capability platform, ecosystem-level connection and local intelligent control
HiLink SDK
Support for multiple modules and chips, and quick integration with low requirements
HiLink module
Based on Huawei LiteOS kernel and in-house JavaScript engine that is service-oriented, with low technical bar and cost

**Figure 16-16 Huawei HiLink access (ecosystem connection)**

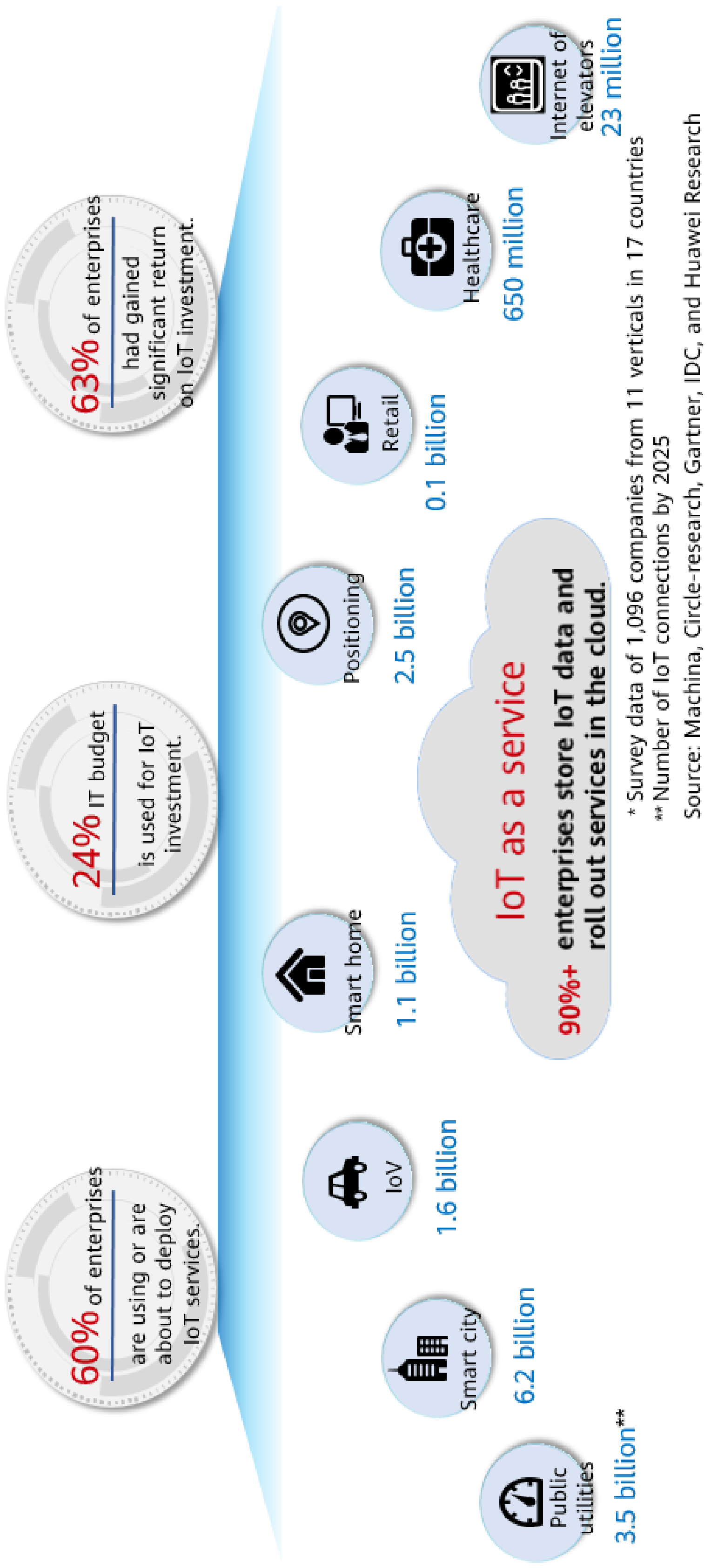

60% of enterprises are using or are about to deploy IoT services.
24% IT budget is used for IoT investment.
63% of enterprises had gained significant return on IoT investment.
Public utilities 3.5 billion**
Smart city 6.2 billion
IoV 1.6 billion
Smart home 1.1 billion
Positioning 2.5 billion
Retail 0.1 billion
Healthcare 650 million
Internet of elevators 23 million
IoT as a service
90%+ enterprises store IoT data and roll out services in the cloud.
* Survey data of 1,096 companies from 11 verticals in 17 countries
** Number of IoT connections by 2025
Source: Machina, Circle-research, Gartner, IDC, and Huawei Research

**Figure 16-17 IoT Is ushering in industry innovation and transformation**

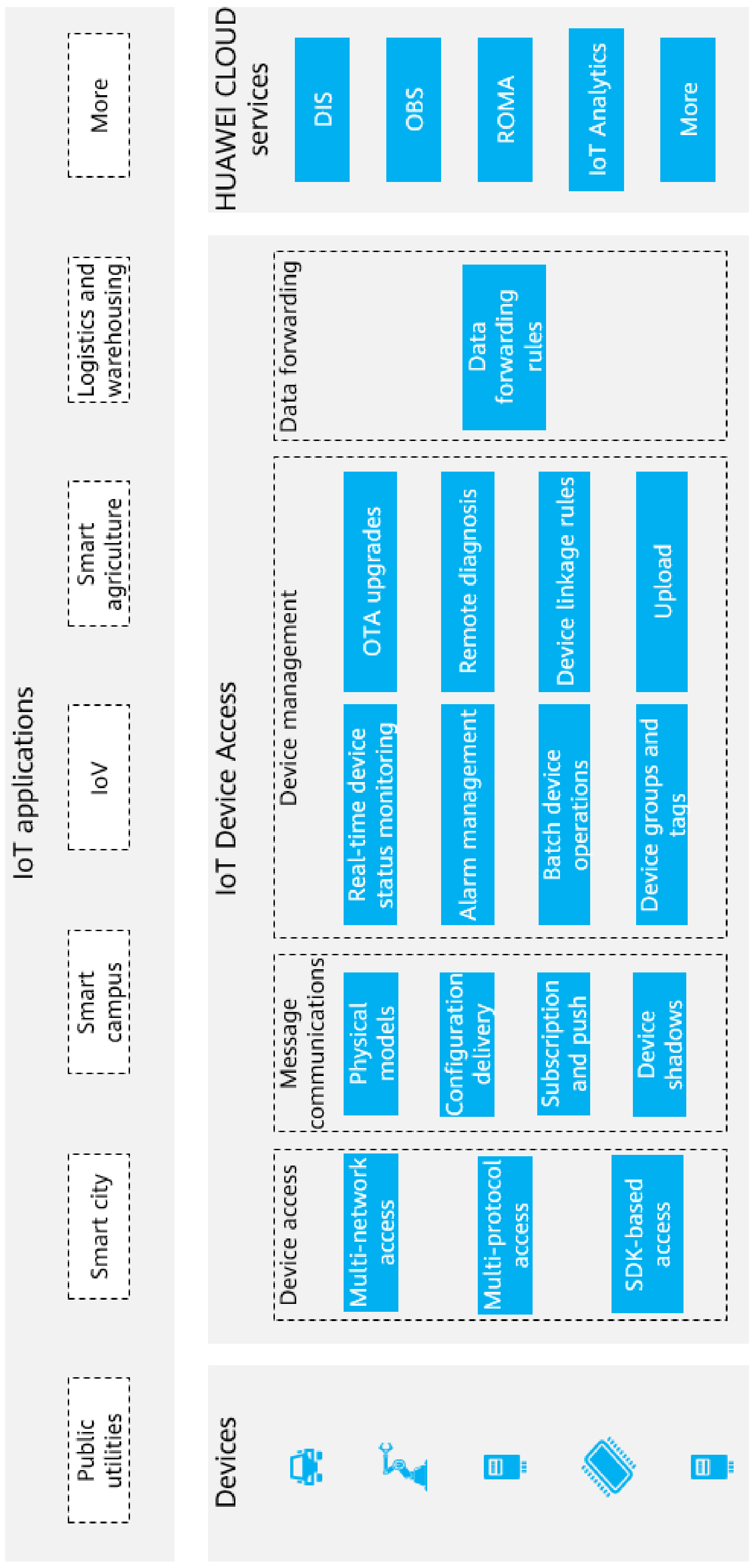
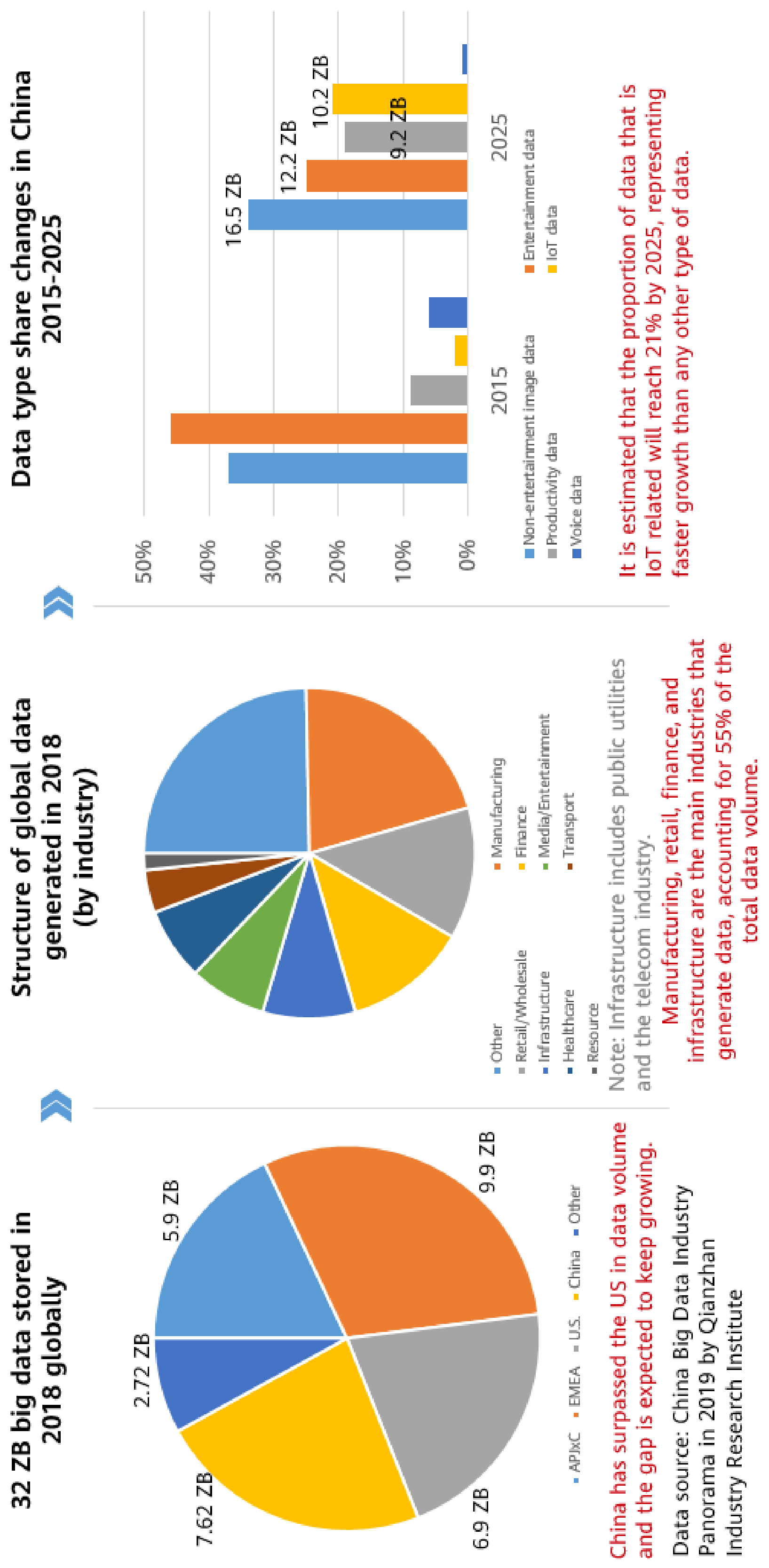


**Figure 16-18 HUAWEI CLOUD IoT platform**



China has surpassed the US in data volume and the gap is expected to keep growing.

Data source: China Big Data Industry Panorama in 2019 by Qianzhan Industry Research Institute

Note: Infrastructure includes public utilities and the telecom industry.

Manufacturing, retail, finance, and infrastructure are the main industries that generate data, accounting for 55% of the total data volume.

It is estimated that the proportion of data that is IoT related will reach 21% by 2025, representing faster growth than any other type of data.

**Figure 16-19 Why is data analysis required?**

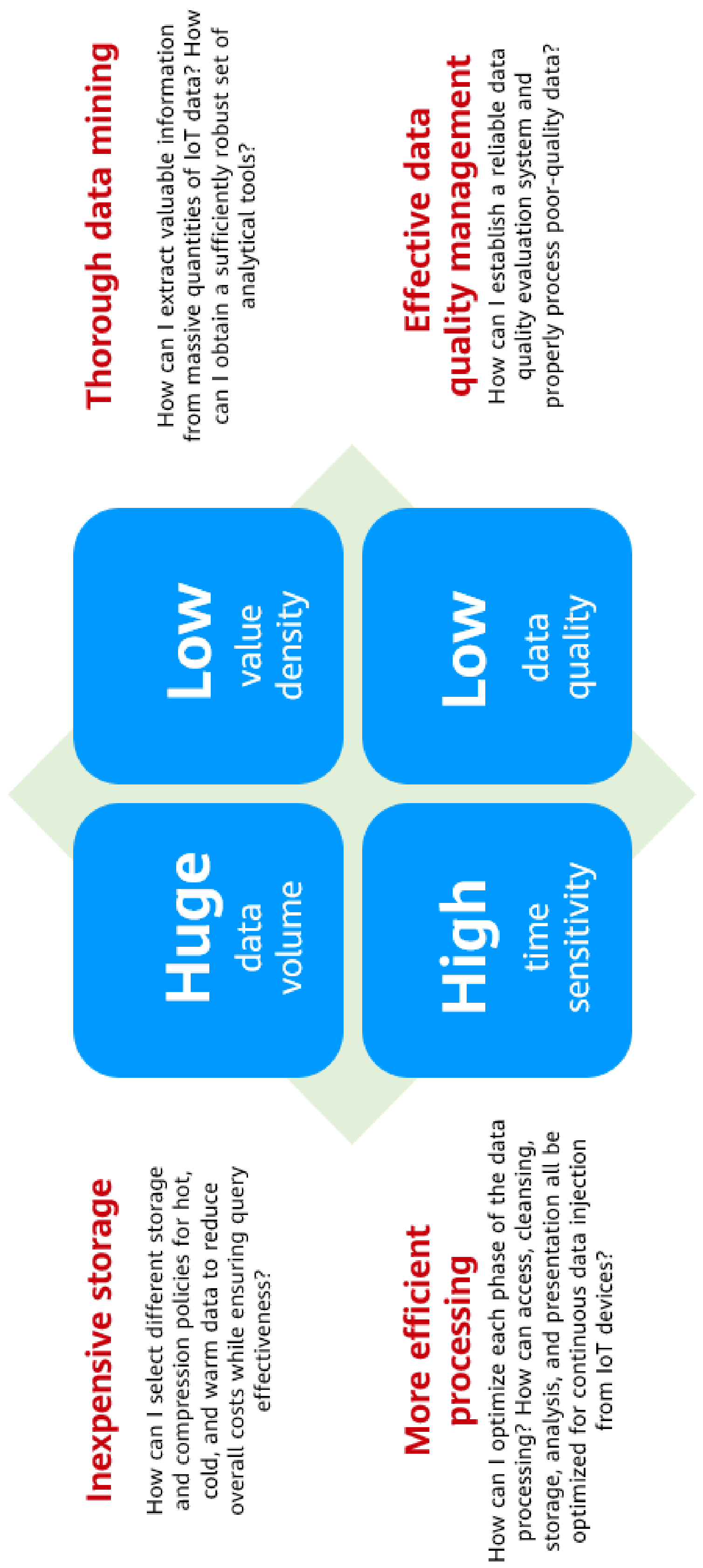

Inexpensive storage
How can I select different storage and compression policies for hot, cold, and warm data to reduce overall costs while ensuring query effectiveness?
Thorough data mining
How can I extract valuable information from massive quantities of IoT data? How can I obtain a sufficiently robust set of analytical tools?
Huge data volume
Low value density
High time sensitivity
Low data quality
More efficient processing
How can I optimize each phase of the data processing? How can access, cleansing, storage, analysis, and presentation all be optimized for continuous data injection from IoT devices?
Effective data quality management
How can I establish a reliable data quality evaluation system and properly process poor-quality data?

**Figure 16-20 IoT data characteristics**

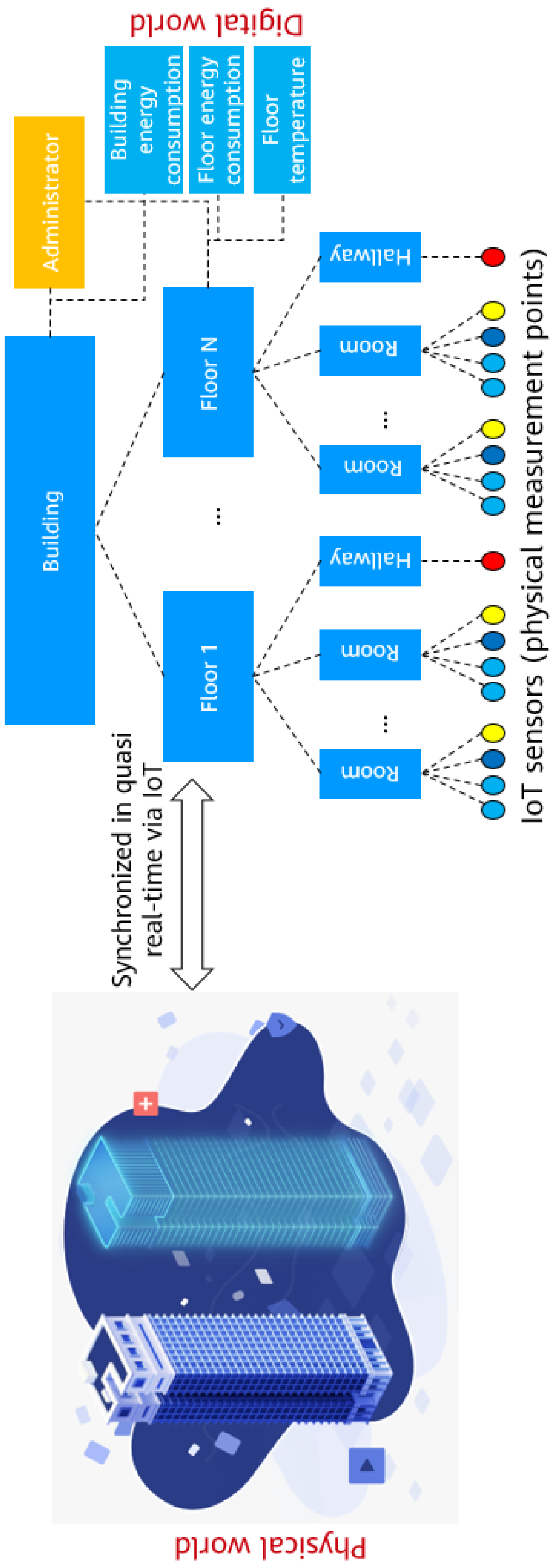


Figure 16-21 Asset model

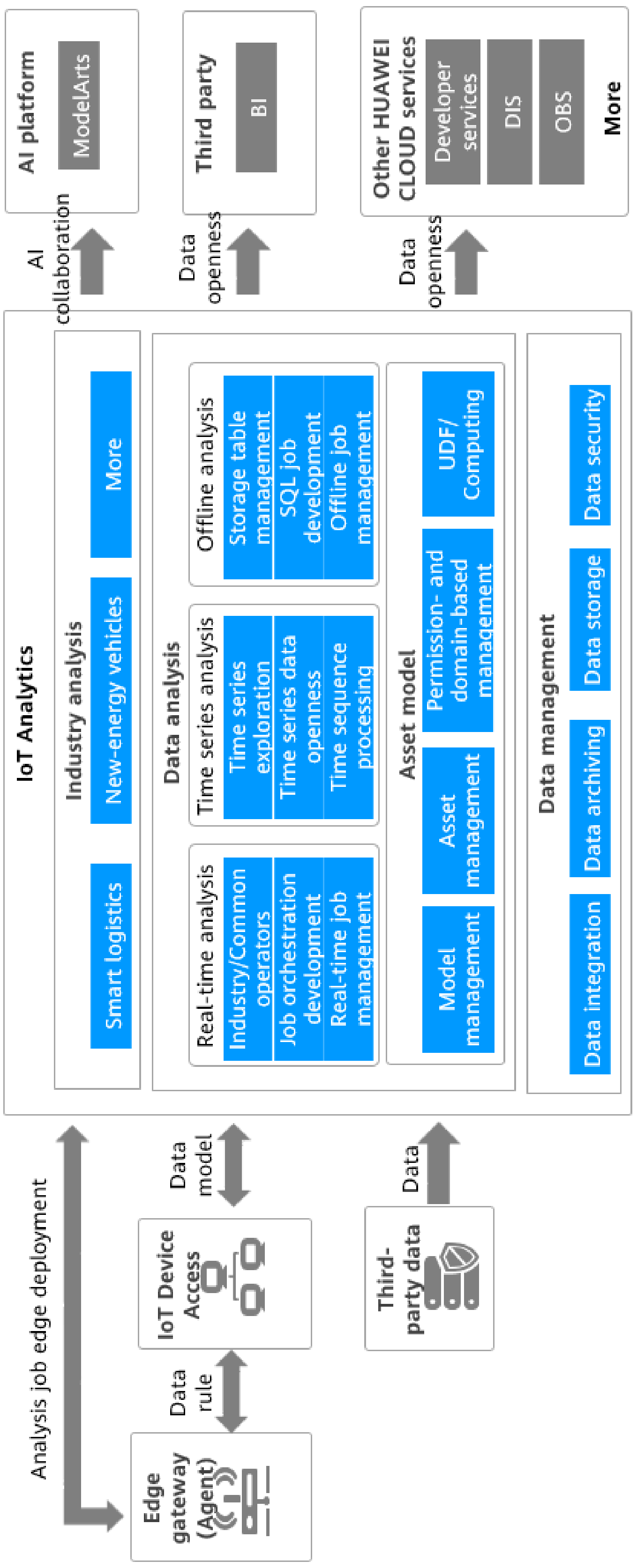


**Figure 16-22 Data analysis architecture**

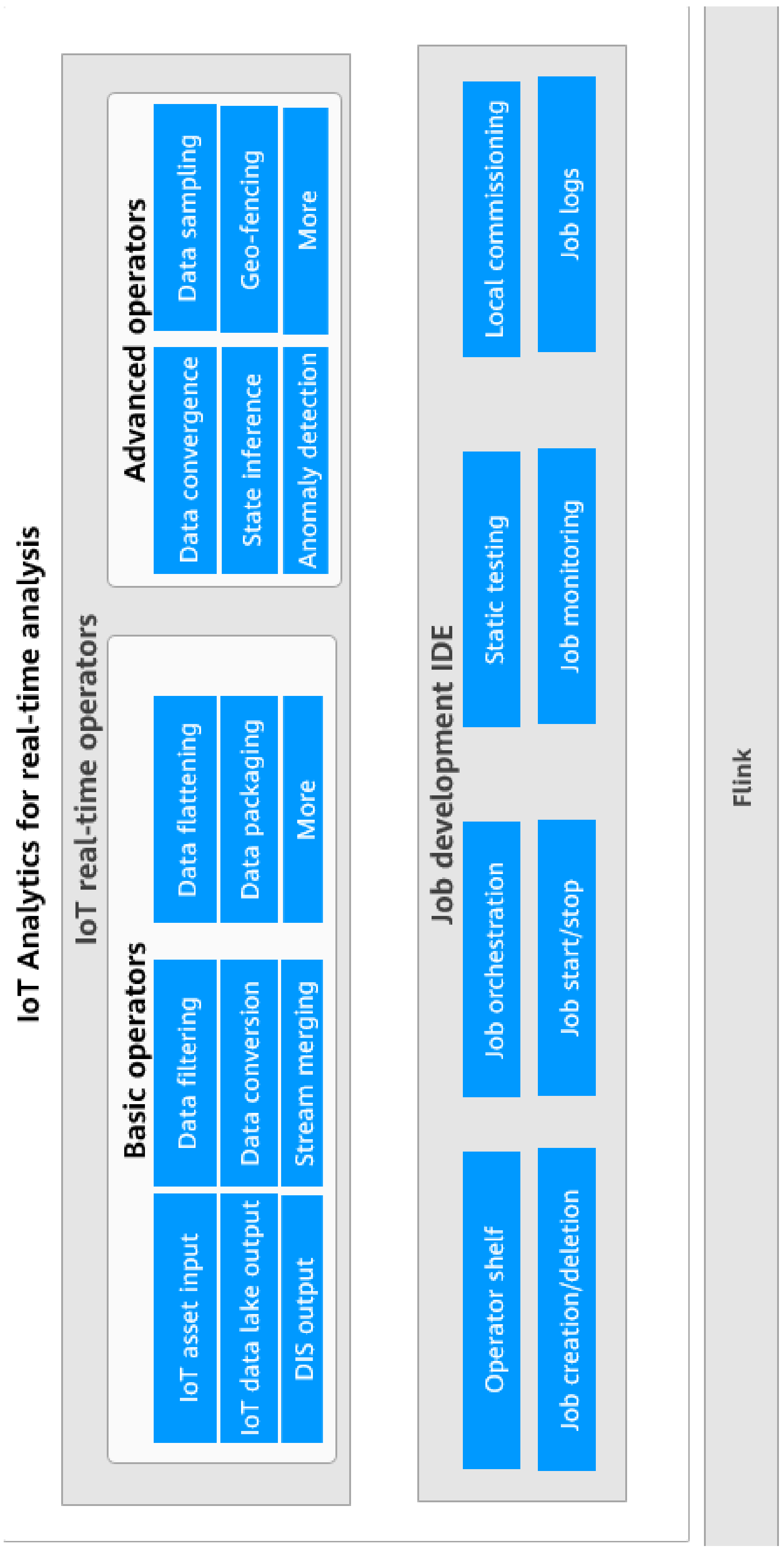


**Figure 16-23 Real-time analysis**

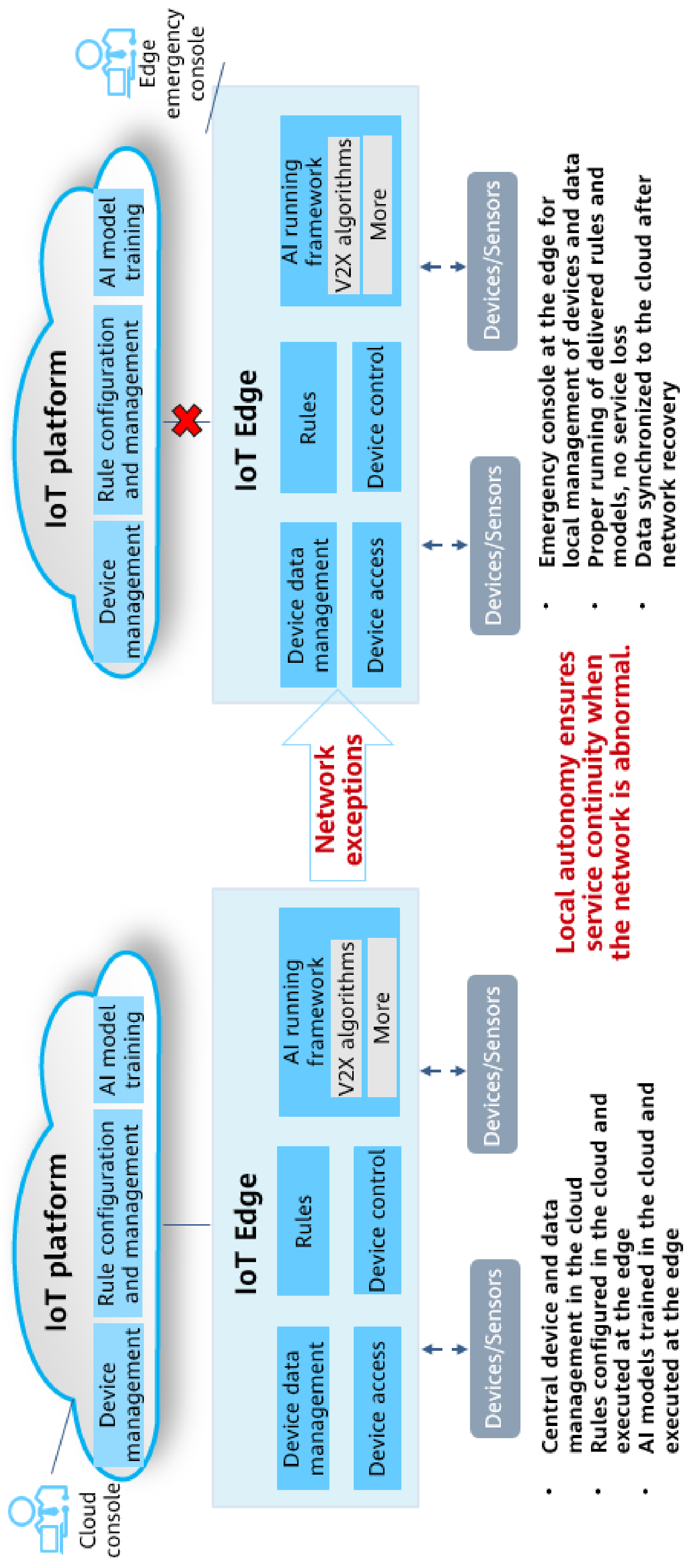

Cloud console
IoT platform
Device management
Rule configuration and management
AI model training
IoT Edge
Device data management
Rules
AI running framework
V2X algorithms
More
Device access
Device control
Devices/Sensors
Devices/Sensors
• Central device and data management in the cloud
• Rules configured in the cloud and executed at the edge
• AI models trained in the cloud and executed at the edge
Network exceptions
Local autonomy ensures service continuity when the network is abnormal.
Edge emergency console
IoT platform
Device management
Rule configuration and management
AI model training
IoT Edge
Device data management
Rules
AI running framework
V2X algorithms
More
Device access
Device control
Devices/Sensors
Devices/Sensors
• Emergency console at the edge for local management of devices and data
• Proper running of delivered rules and models, no service loss
• Data synchronized to the cloud after network recovery

**Figure 16-24 Characteristics of edge computing**

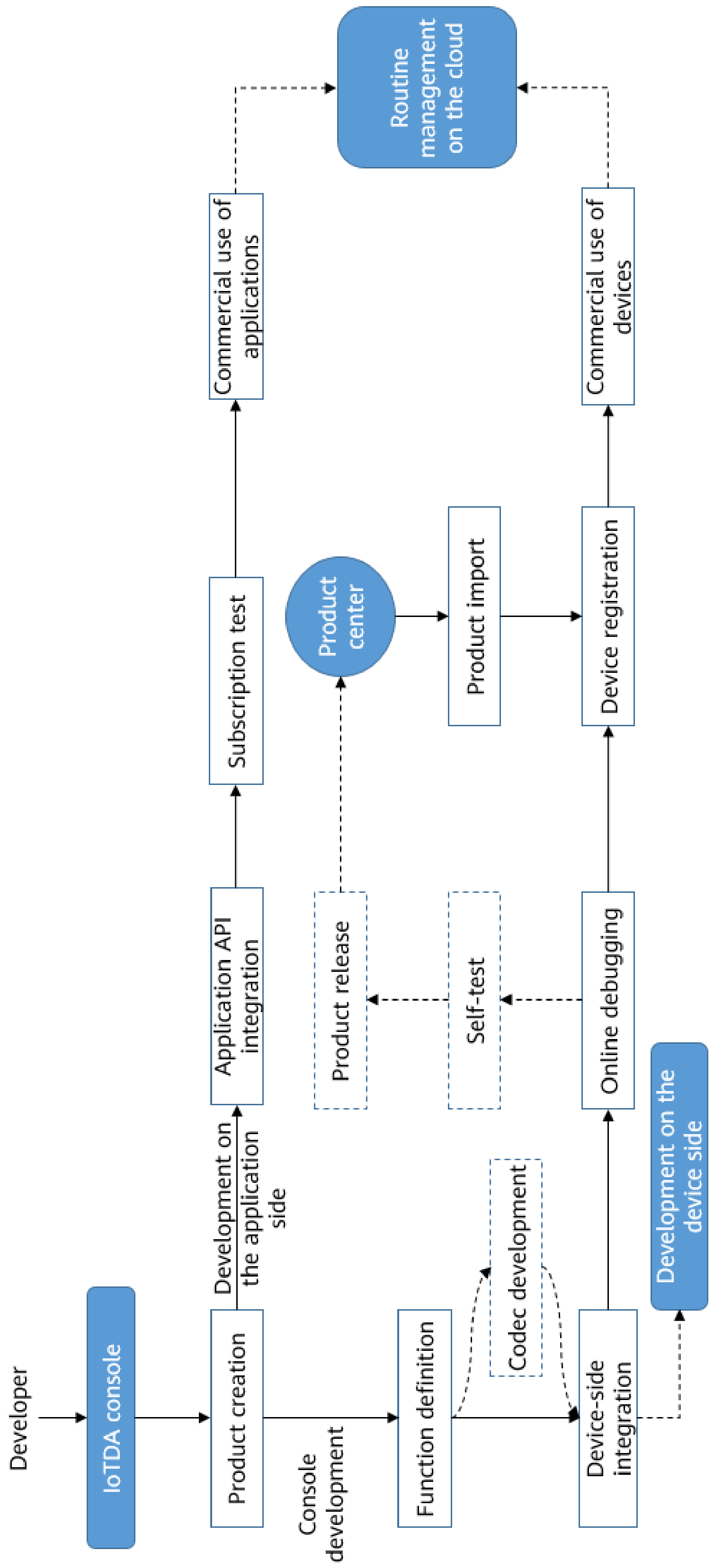


**Figure 16-25 IoT solution development process based on the IoT platform**

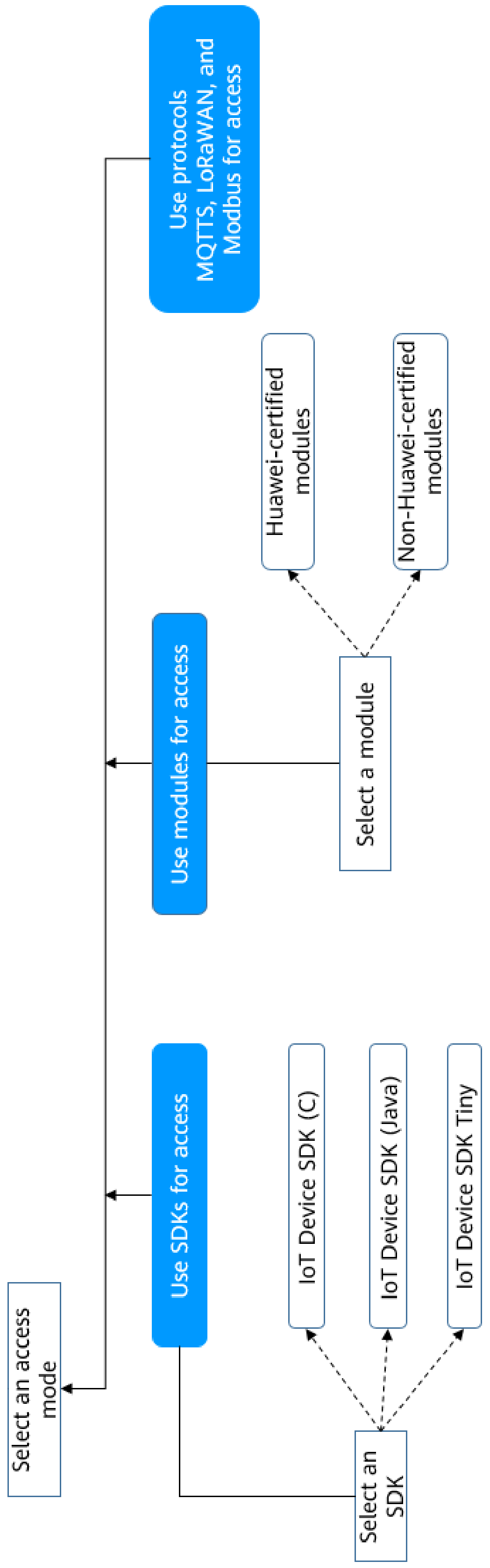


**Figure 16-26 Development on the device side**

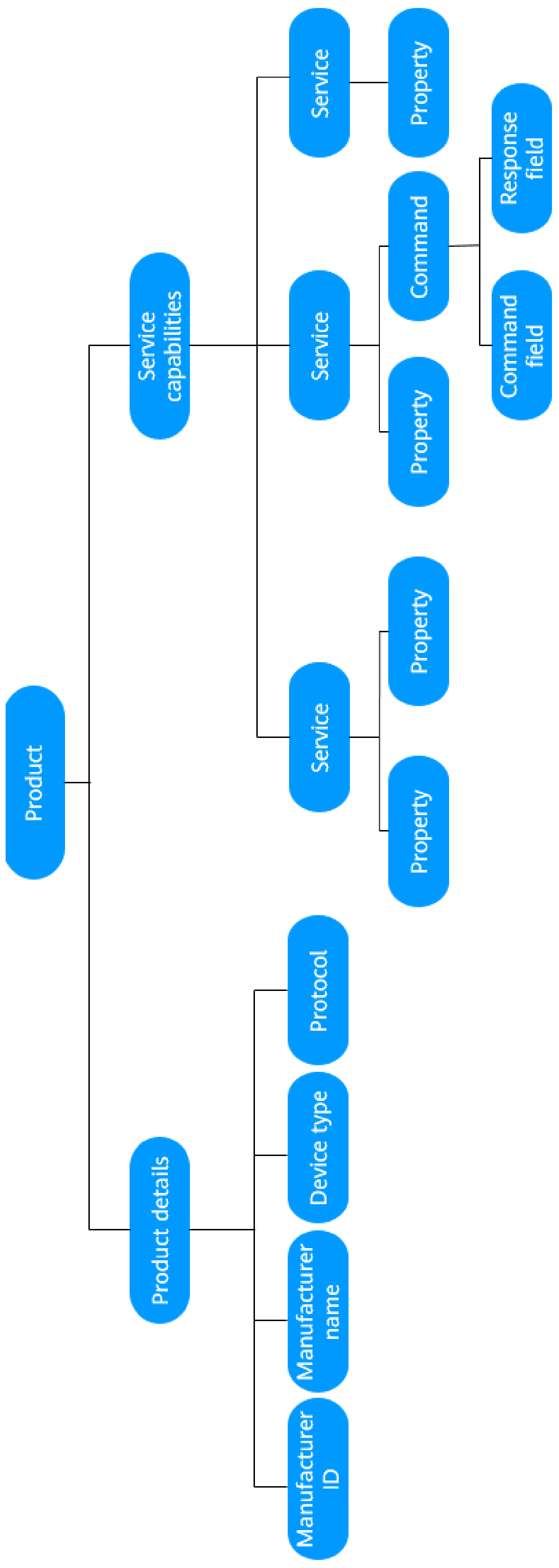
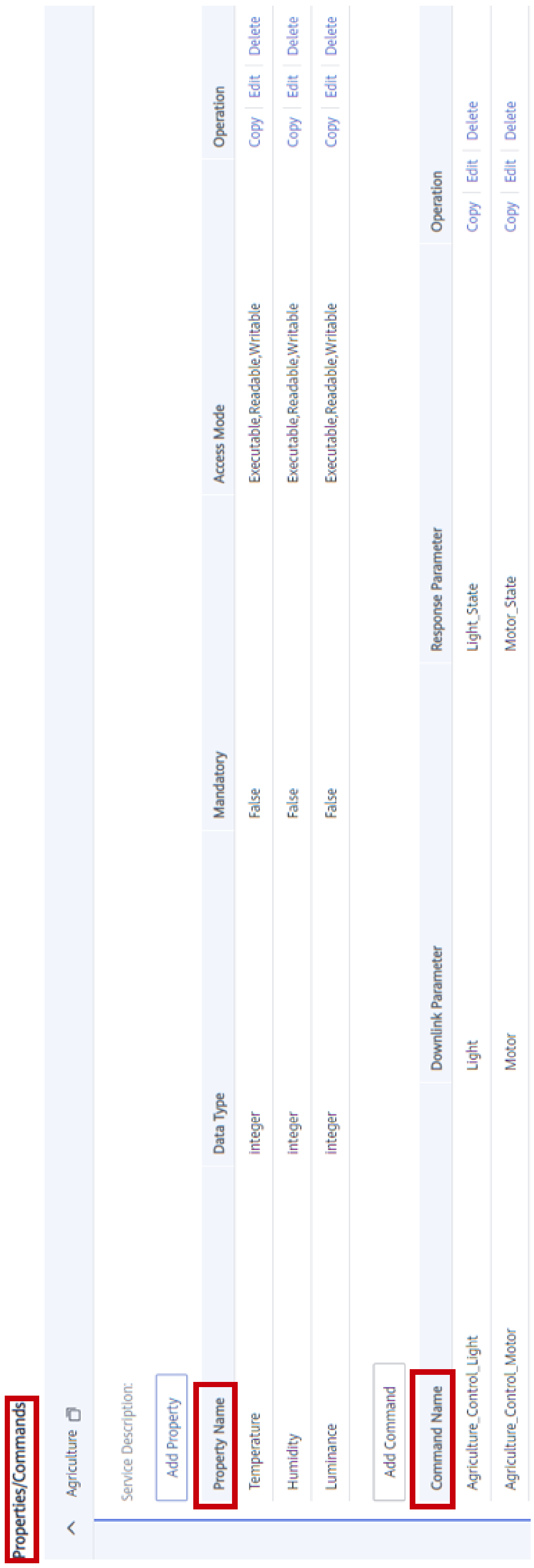


**Figure 16-27 Product model**



| Property Name | Data Type | Mandatory | Access Mode | Operation |
|---|---|---|---|---|
| Temperature | integer | False | Executable,Readable,Writable | Copy \| Edit \| Delete |
| Humidity | integer | False | Executable,Readable,Writable | Copy \| Edit \| Delete |
| Luminance | integer | False | Executable,Readable,Writable | Copy \| Edit \| Delete |

| Command Name | Downlink Parameter | Response Parameter | Operation |
|---|---|---|---|
| Agriculture_Control_Light | Light | Light_State | Copy \| Edit \| Delete |
| Agriculture_Control_Motor | Motor | Motor_State | Copy \| Edit \| Delete |

**Figure 16-28 Product model example**

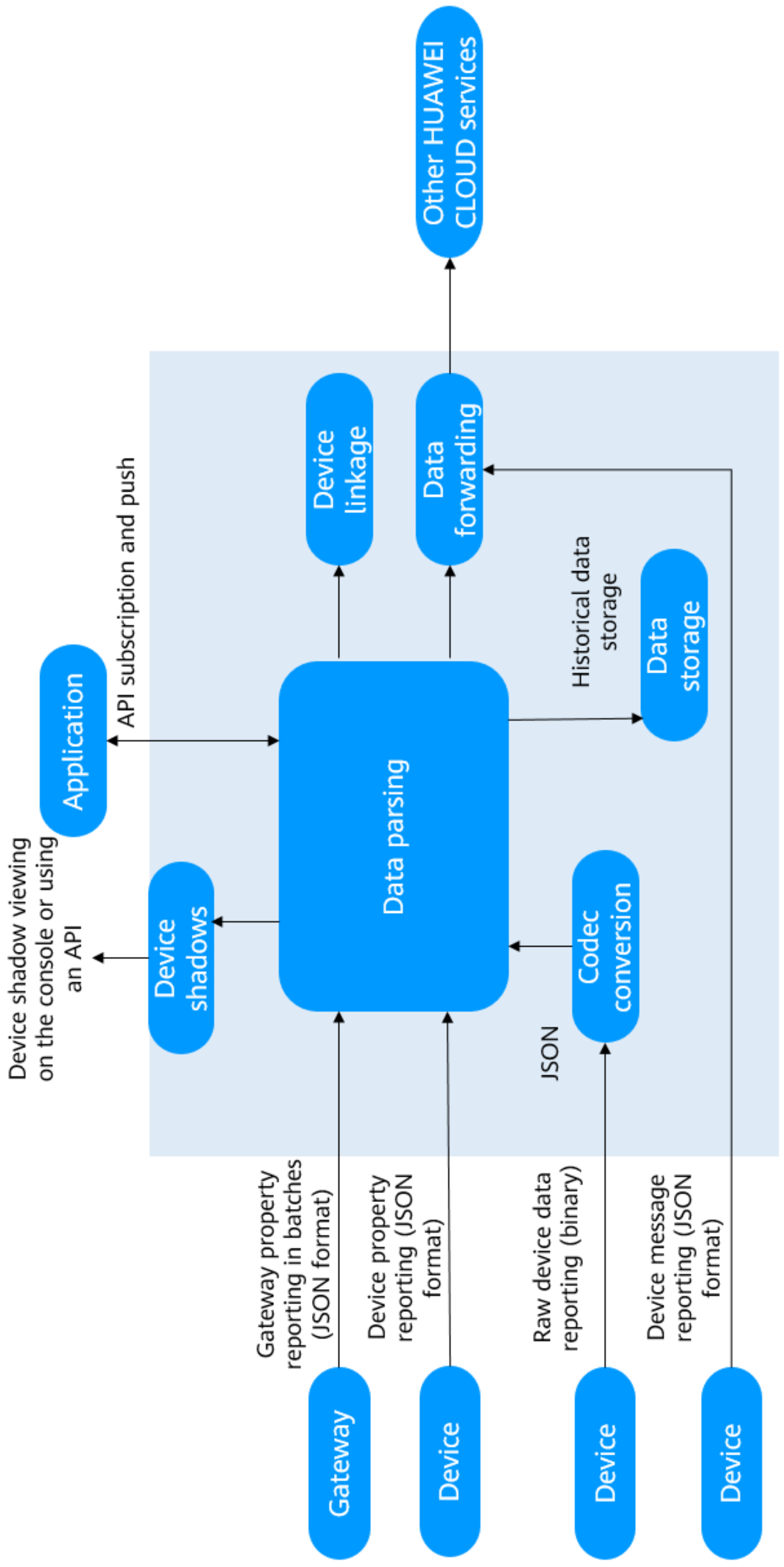

Gateway
Device
Device
Device
Gateway property reporting in batches (JSON format)
Device property reporting (JSON format)
Raw device data reporting (binary)
Device message reporting (JSON format)
JSON
Codec conversion
Data parsing
Device shadows
Device shadow viewing on the console or using an API
Application
API subscription and push
Historical data storage
Data storage
Device linkage
Data forwarding
Other HUAWEI CLOUD services

**Figure 16-29 Data reporting process**

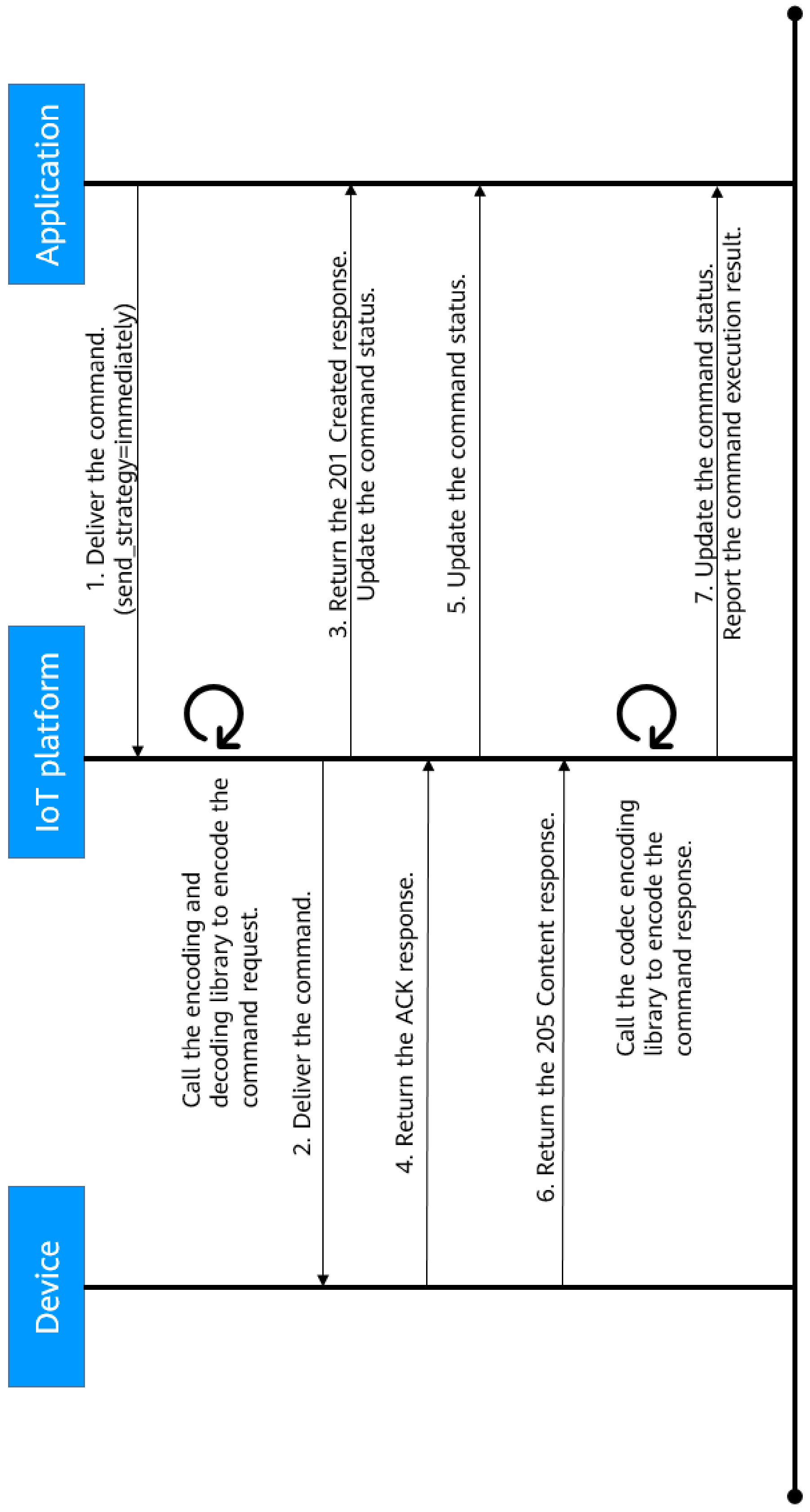

Application
IoT platform
Device
1. Deliver the command.
(send_strategy=immediately)
Call the encoding and decoding library to encode the command request.
2. Deliver the command.
3. Return the 201 Created response.
Update the command status.
4. Return the ACK response.
5. Update the command status.
6. Return the 205 Content response.
Call the codec encoding library to encode the command response.
7. Update the command status.
Report the command execution result.

**Figure 16-30 Immediate delivery of LwM2M/CoAP device commands**

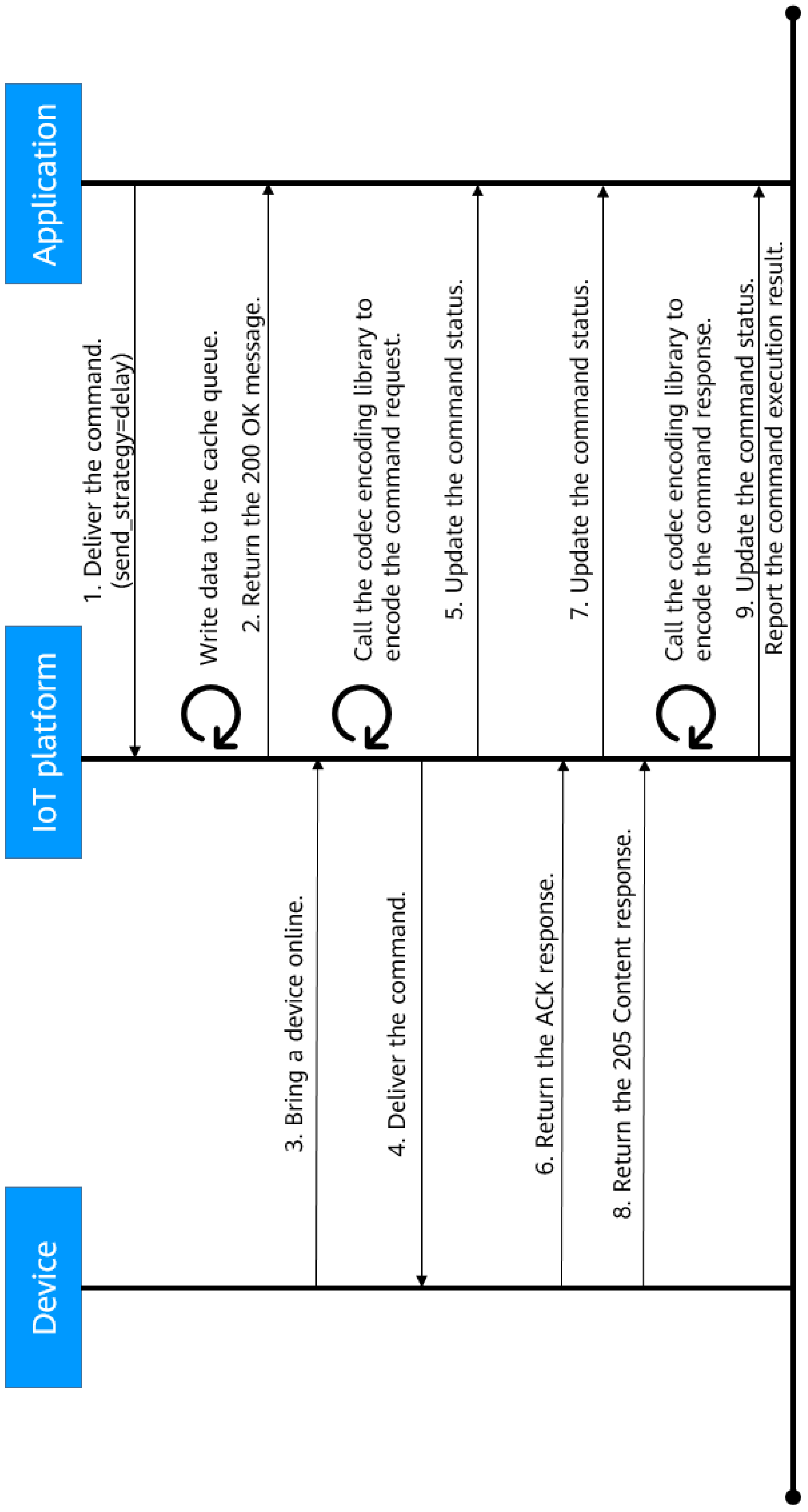

Application
IoT platform
Device
1. Deliver the command. (send_strategy=delay)
Write data to the cache queue.
2. Return the 200 OK message.
3. Bring a device online.
Call the codec encoding library to encode the command request.
4. Deliver the command.
5. Update the command status.
6. Return the ACK response.
7. Update the command status.
8. Return the 205 Content response.
Call the codec encoding library to encode the command response.
9. Update the command status.
Report the command execution result.

**Figure 16-31 Delayed delivery of LwM2M/CoAP device commands**

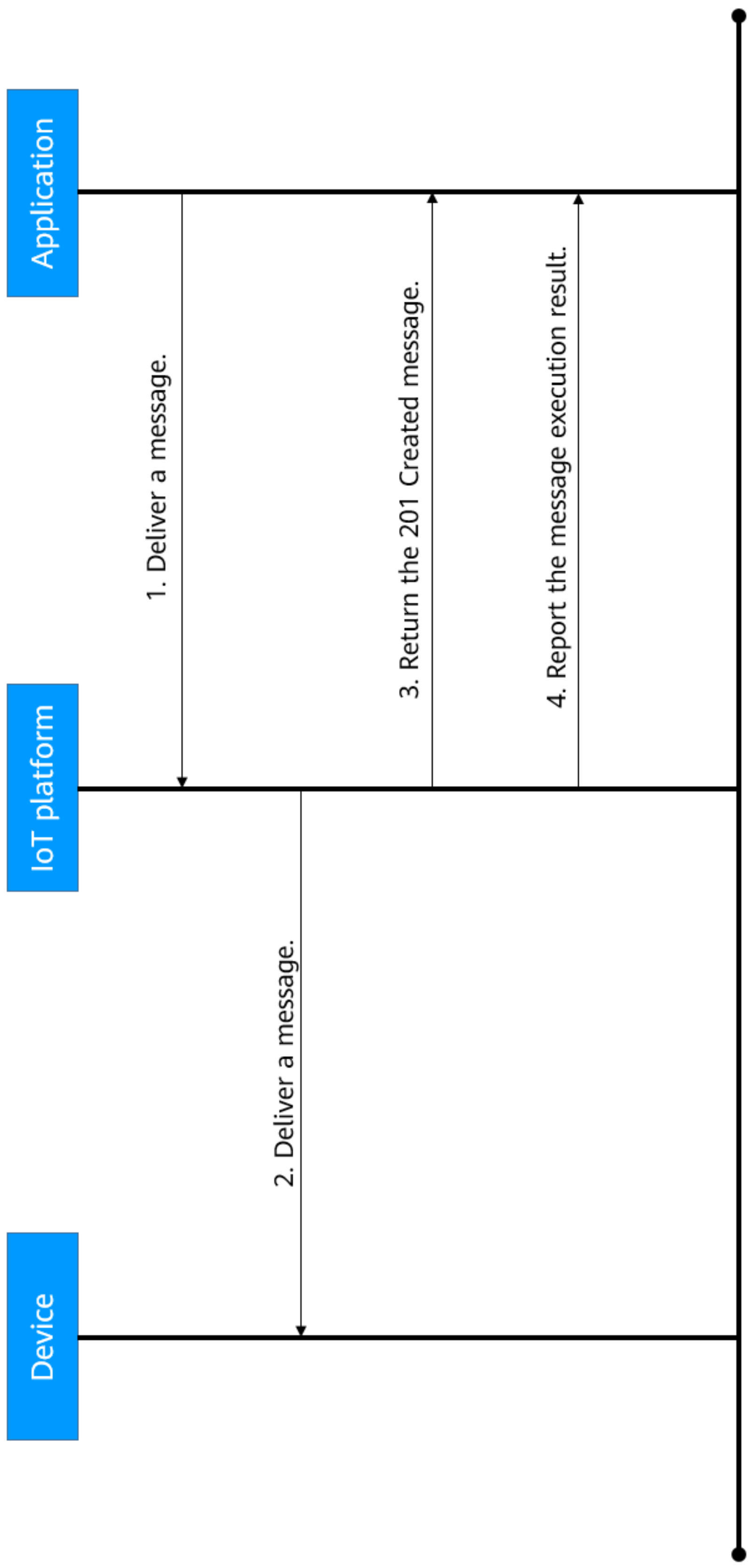

Application
IoT platform
Device
1. Deliver a message.
2. Deliver a message.
3. Return the 201 Created message.
4. Report the message execution result.

**Figure 16-32 Immediate delivery of MQTT device commands**

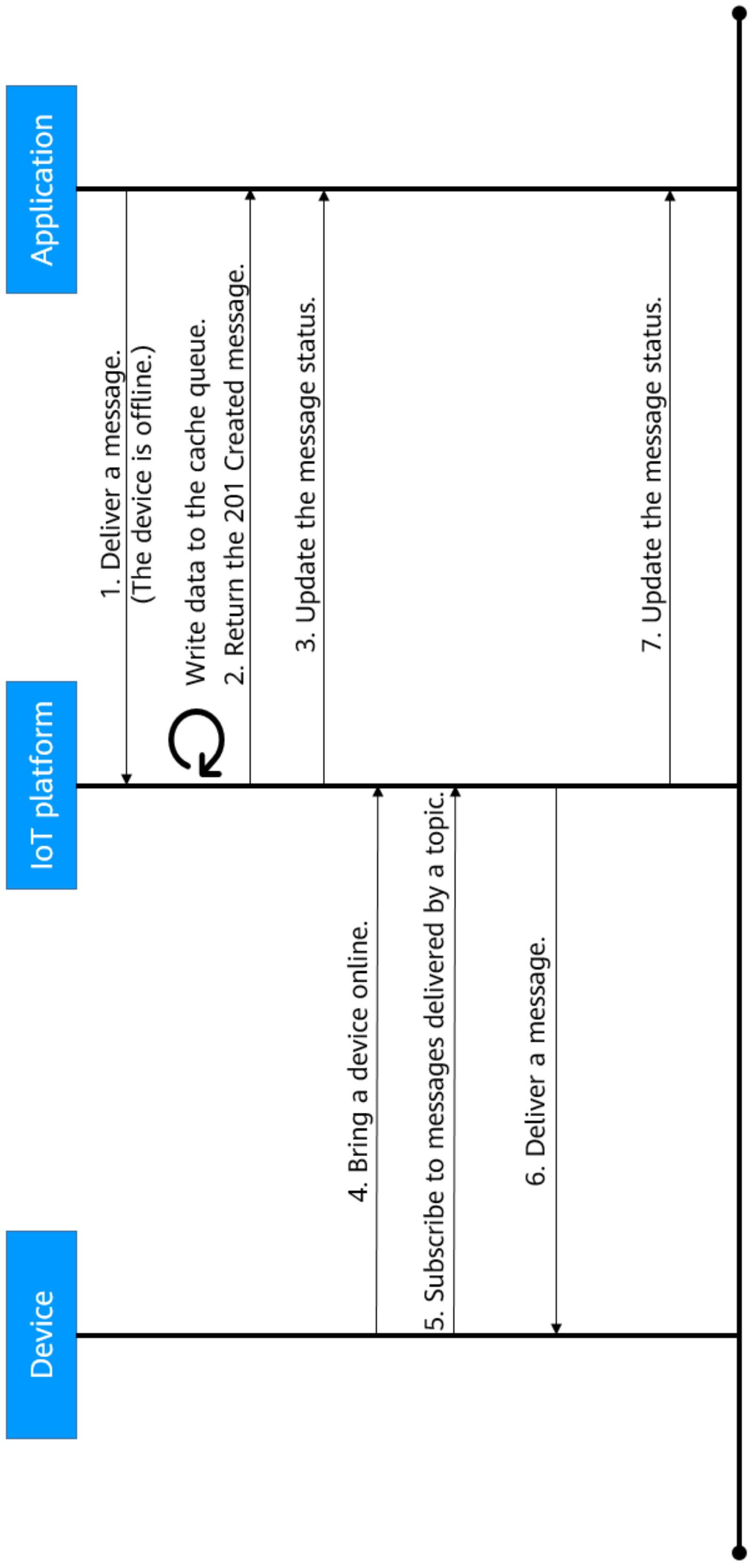
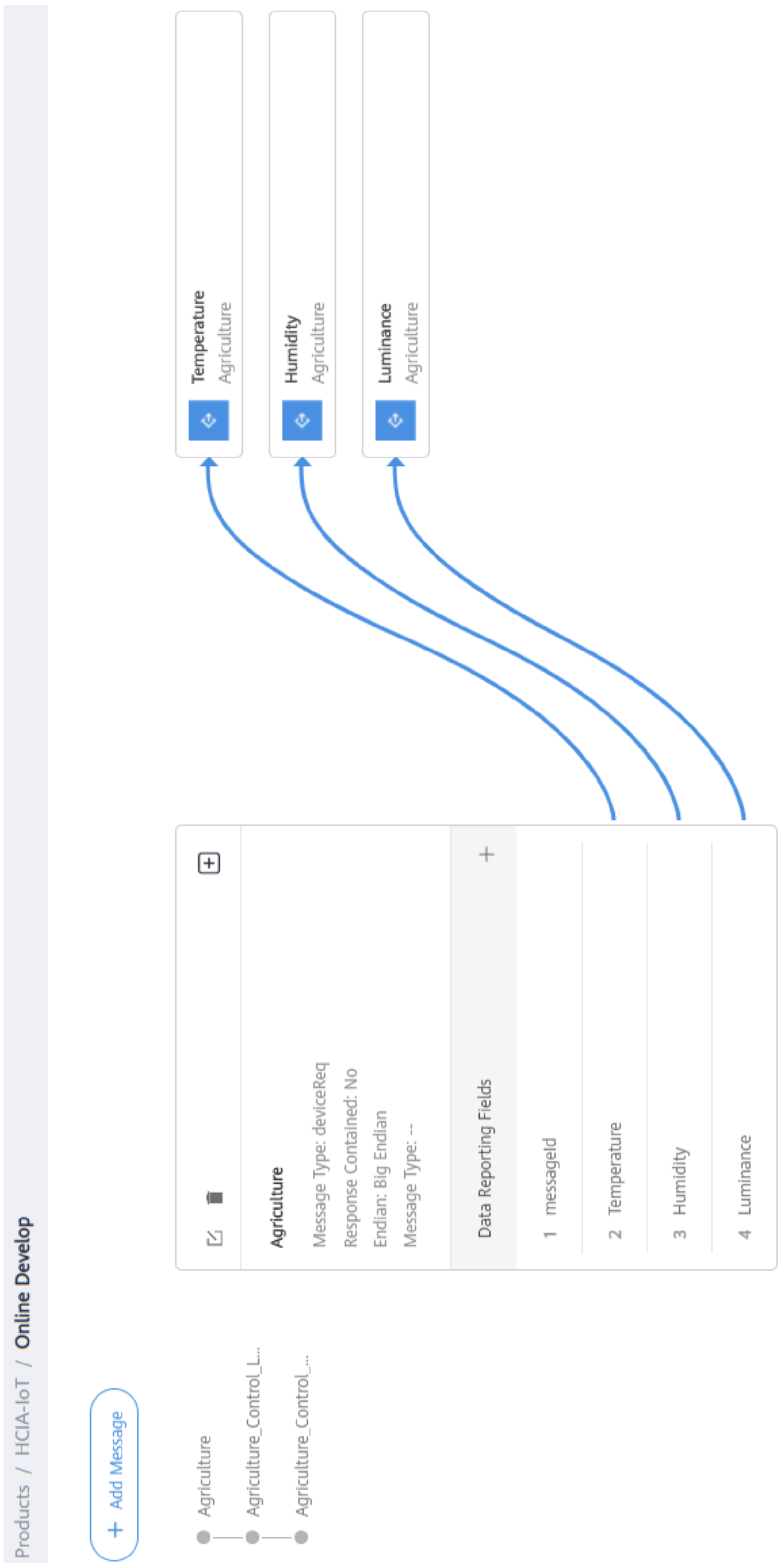

Application
IoT platform
Device
1. Deliver a message.
(The device is offline.)
Write data to the cache queue.
2. Return the 201 Created mesage.
3. Update the message status.
4. Bring a device online.
5. Subscribe to messages delivered by a topic.
6. Deliver a message.
7. Update the message status.

**Figure 16-33 Delayed delivery of MQTT device commands**



Agriculture

Message Type: deviceReq

Response Contained: No

Endian: Big Endian

Message Type: --

Data Reporting Fields

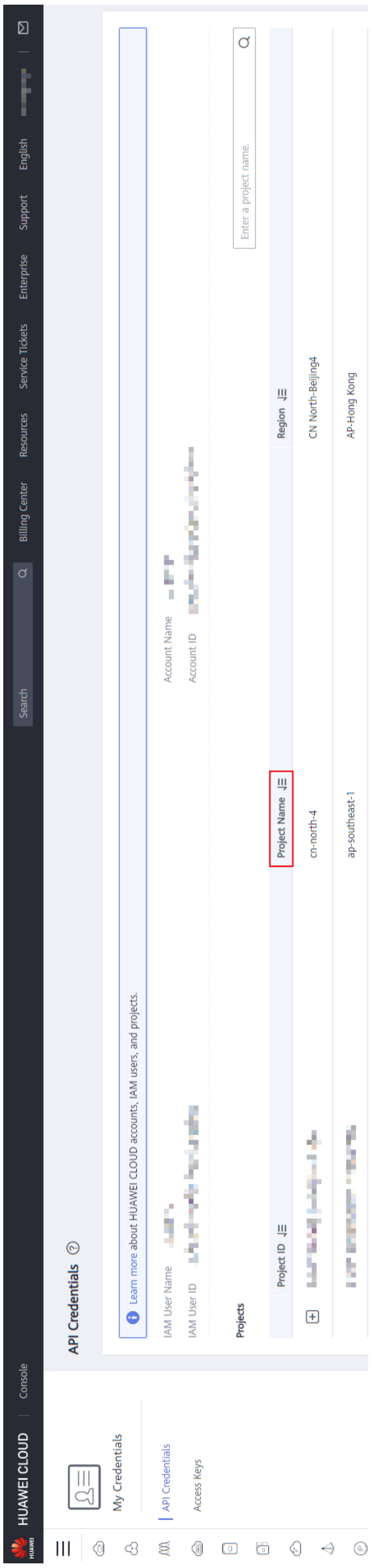
1 messageId

2 Temperature

3 Humidity

4 Luminance

**Figure 16-34 Codec example**



API Credentials

Learn more about HUAWEI CLOUD accounts, IAM users, and projects.

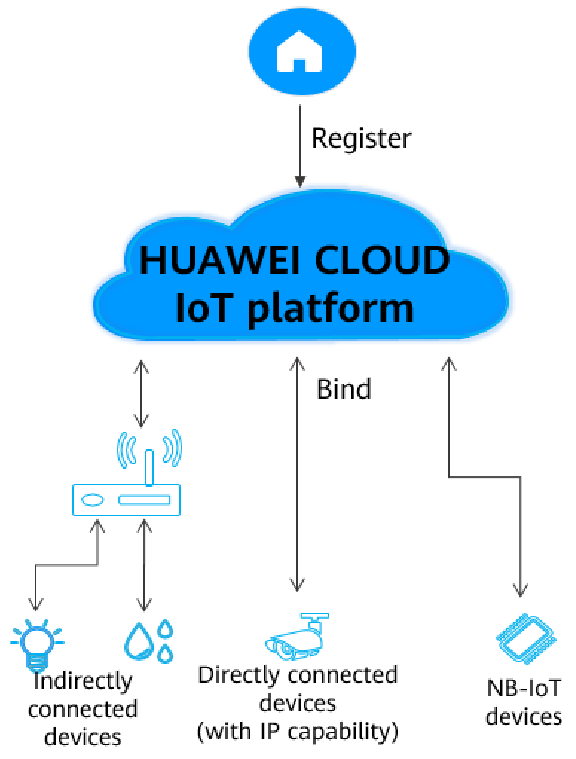
IAM User Name [redacted]
IAM User ID [redacted]
Account Name [redacted]
Account ID [redacted]

Projects

Enter a project name.

| Project ID | Project Name | Region |
|---|---|---|
| [redacted] | cn-north-4 | CN North-Beijing4 |
| [redacted] | ap-southeast-1 | AP-Hong Kong |

**Figure 16-35 API credentials**

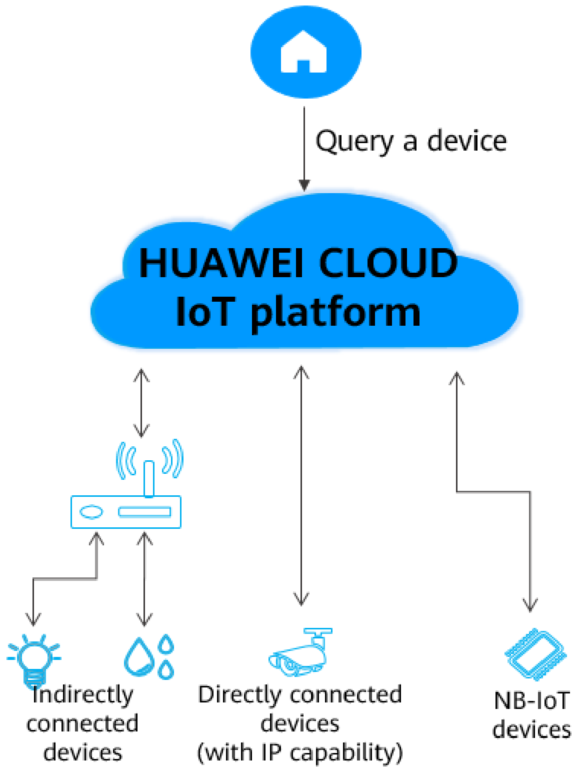

Register
HUAWEI CLOUD IoT platform
Bind
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** POST

```
Request:
https://{Endpoint}/v5/iot/{project_id}/devices
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
{
  "device_id" : "d4922d8a-6c8e-4396-852c-
164aefa6638f",
  "node_id" : "ABC123456789",
  "device_name" : "dianadevice",
  "product_id" :
"b640f4c203b7910fc3cbd446ed437cbd",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secure_access" : true,
    "fingerprint" :
"dc0f1016f495157344ac5f1296335cff725ef22f",
    "secret" :
"3b935a250c50dc2c6d481d048cefdc3c",
    "timeout" : 300
  },
  ...
}
```

```
Response:
// Status Code:
Status Code: 201 Created

Content-Type: application/json
// Body:
{
  "app_id" : "***",
  "app_name" : "***",
  "device_id" : "***",
  "node_id" : "***",
  "gateway_id" : "***",
  "device_name" : "***",
  "node_type" : "***",
  "description" : "***",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "***",
    "fingerprint" : "***",
    "secure_access" : true,
    "timeout" : 300
  },
  ...
  }
}
```

Figure 16-36 API for creating a device

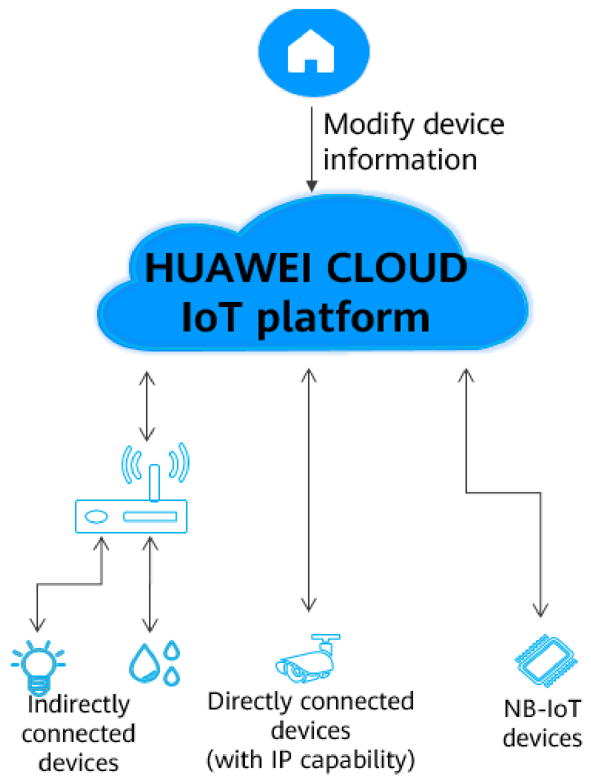

Query a device
HUAWEI CLOUD
IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** GET

**Request:**
```
https://{Endpoint}/v5/iot/{project_id}/devices/{d
evice_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

**Response:**
```
// Status Code:
Status Code: 200 OK
Content-Type: application/json
//Body:
{
 "app_id" : "***",
  "app_name" : "***",
  "device_id" : "***",
  "node_id" : "***",
  "gateway_id" : "***",
  "device_name" : "***",
  "node_type" : "***",
  "description" : "***",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "***",
    "secret" : "***",
    "fingerprint" : "***",
    "secure_access" : true,
    "timeout" : 300
  },
...}
```

**Figure 16-37 API for querying a device**

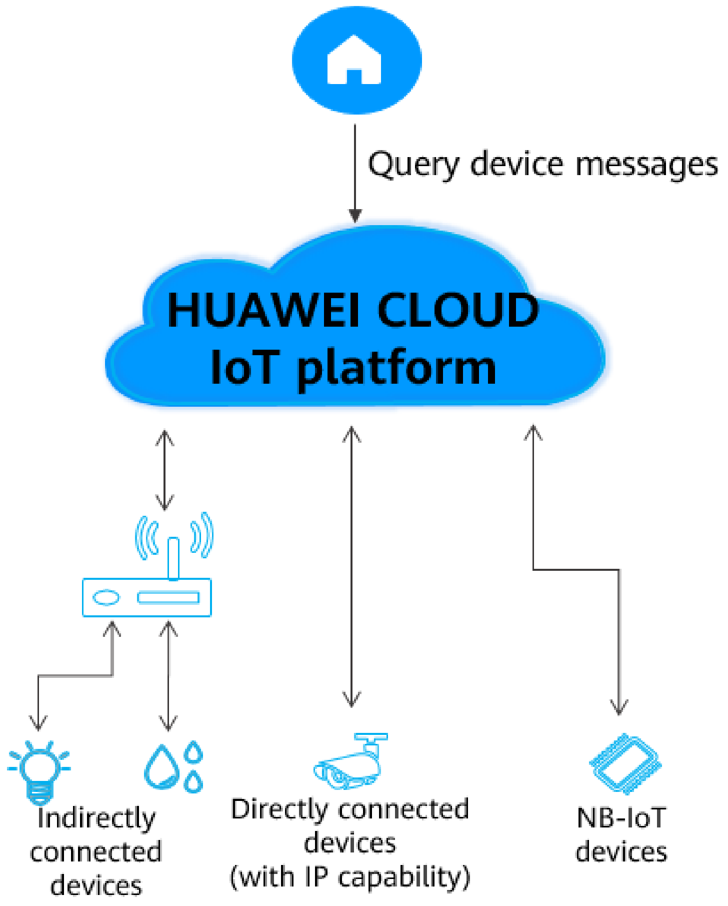

Modify device information
HUAWEI CLOUD IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** PUT

```
Request:
https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********

{
  "device_name" : "dianadevice",
  "description" : "watermeter device",
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  },
  "auth_info" : {
    "secure_access" : true,
    "timeout" : 300
  }
}
```

```
Response:
// Status Code:
Status Code: 200 OK
```

**Figure 16-38 API for modifying device information**


Query device messages
HUAWEI CLOUD IoT platform
Indirectly connected devices
Directly connected devices (with IP capability)
NB-IoT devices

**Method:** GET

**Request:**
```
https://{Endpoint}/v5/iot/{project_id}/devices/{device_
id}/messages
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

**Response:**
```
// Status Code:
Status Code: 200 OK
Content-Type: application/json

{
  "device_id" : "d4922d8a-6c8e-4396-852c-
164aefa6638f",
  "messages" : [ {
    "message_id" : "b1224afb-e9f0-4916-8220-
b6bab568e888",
    "name" : "message_name",
    "message" : "string",
    "topic" : "string",
    "status" : "PENDING",
    "created_time" : "20151212T121212Z",
    "finished_time" : "20151212T121212Z"
  } ]
}
```

Figure 16-39 API for querying device messages

**Method:** DELETE

```
Request:
https://{Endpoint}/v5/iot/{project_id}/devices/
{device_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

```
Response:
Status Code: 204 No Content
```

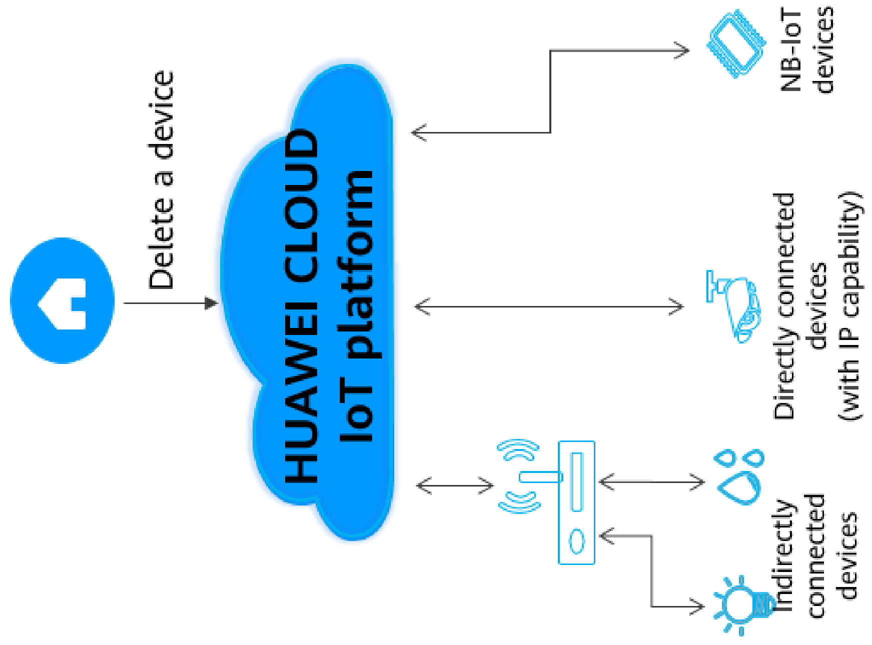

Delete a device
HUAWEI CLOUD IoT platform
NB-IoT devices
Directly connected devices (with IP capability)
Indirectly connected devices

Figure 16-40 API for deleting a device

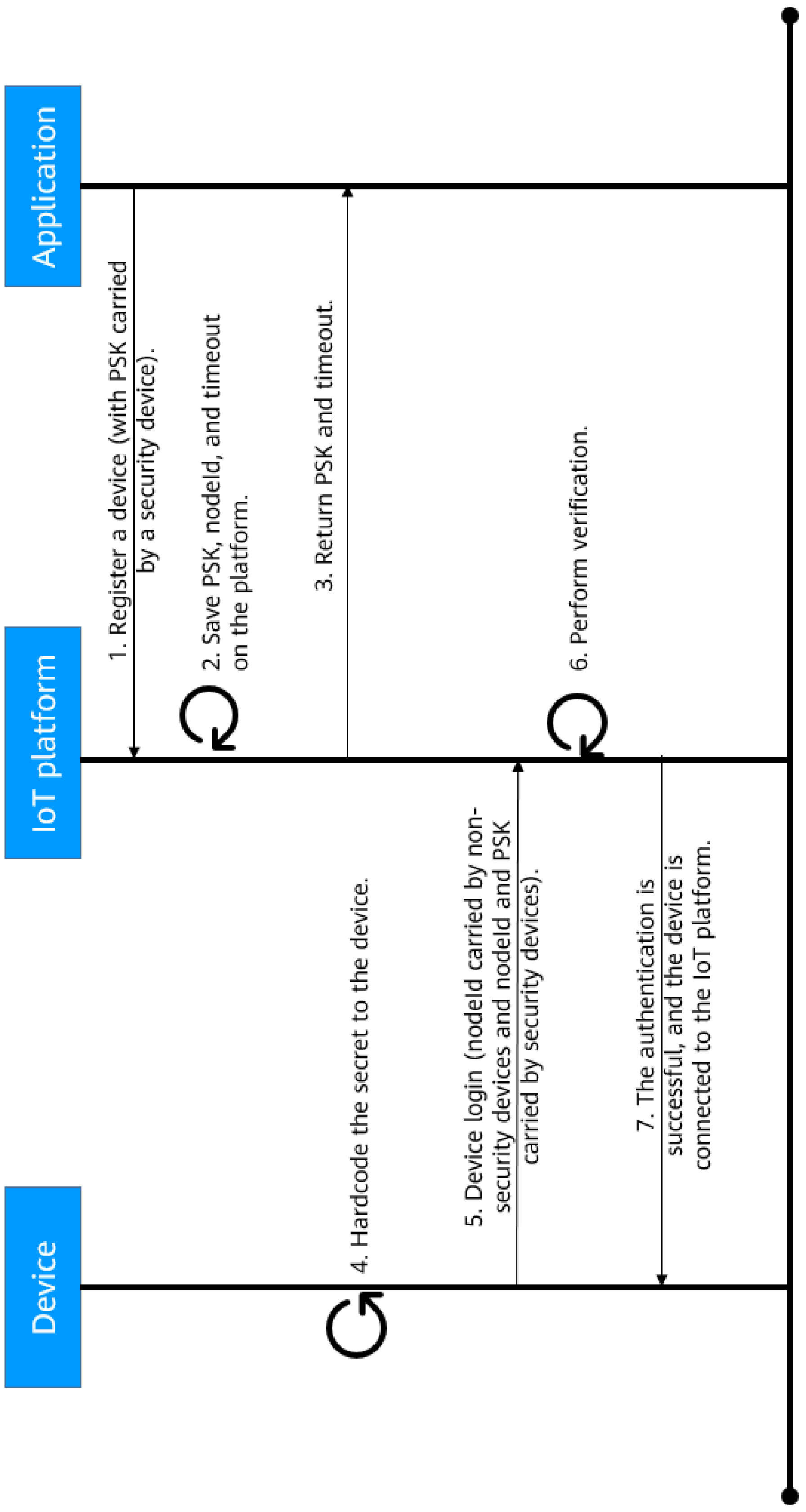

Application
IoT platform
Device
1. Register a device (with PSK carried by a security device).
2. Save PSK, nodeId, and timeout on the platform.
3. Return PSK and timeout.
4. Hardcode the secret to the device.
5. Device login (nodeId carried by non-security devices and nodeId and PSK carried by security devices).
6. Perform verification.
7. The authentication is successful, and the device is connected to the IoT platform.

**Figure 16-41 Authentication for devices using LwM2M over CoAP**

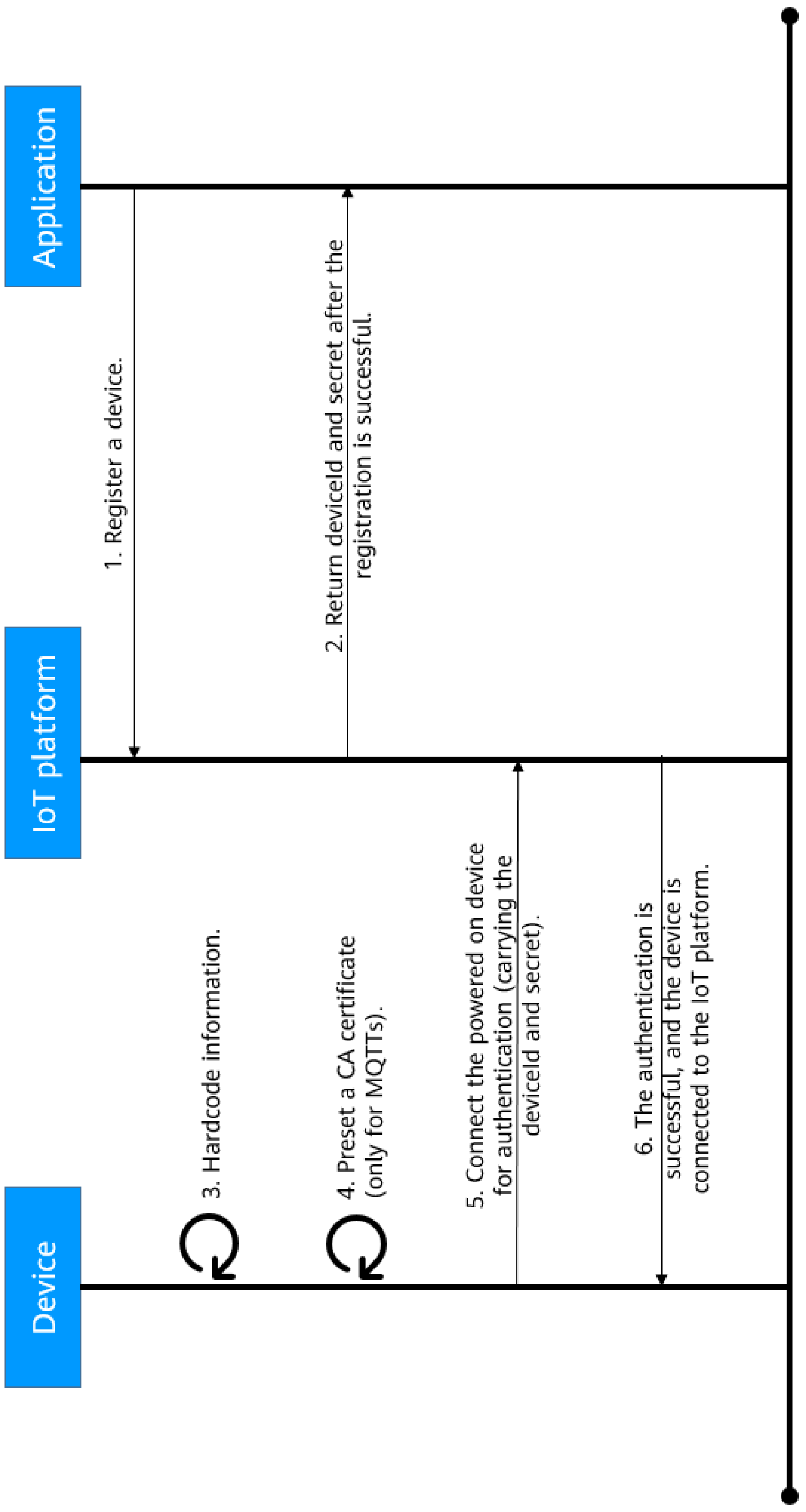

Application
IoT platform
Device
1. Register a device.
2. Return deviceId and secret after the registration is successful.
3. Hardcode information.
4. Preset a CA certificate (only for MQTTs).
5. Connect the powered on device for authentication (carrying the deviceId and secret).
6. The authentication is successful, and the device is connected to the IoT platform.

**Figure 16-42 Authentication for devices using Native MQTT or MQTTS**

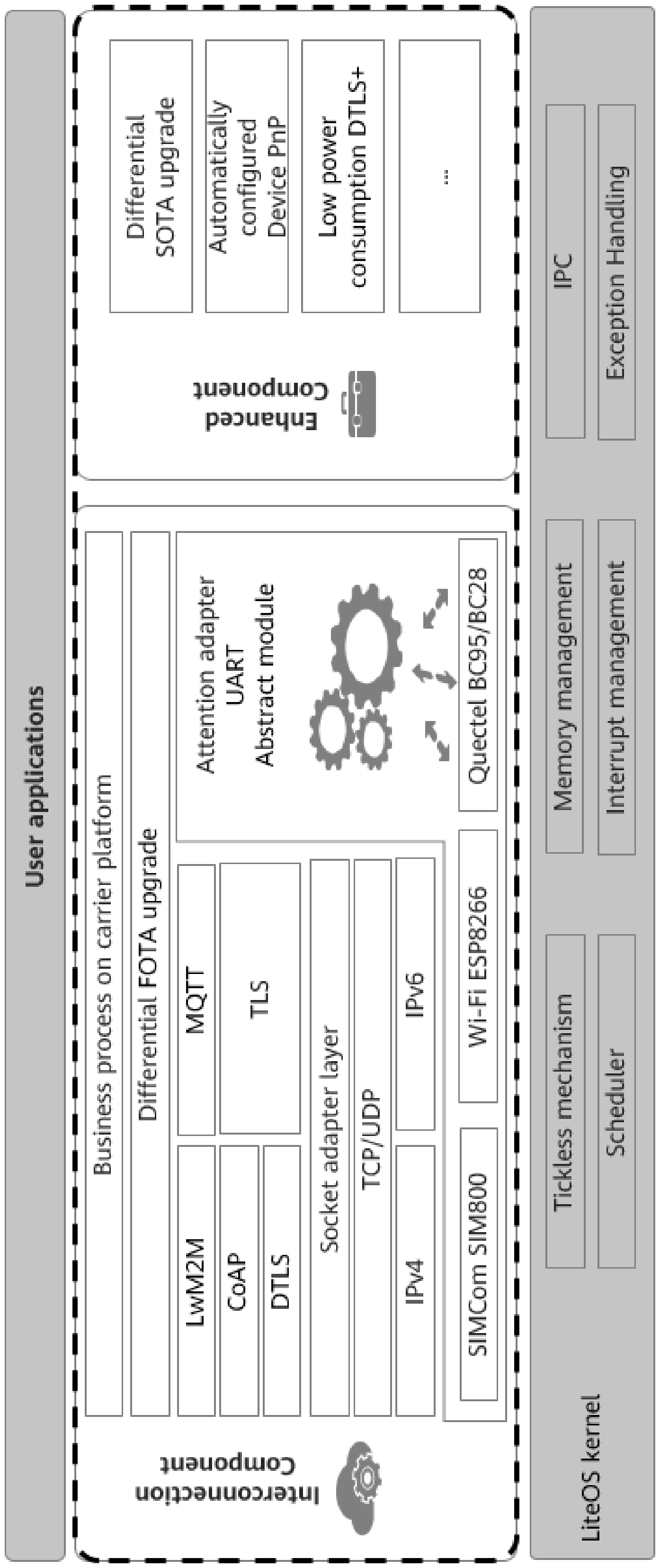


**Figure 16-43 Huawei LiteOS framework**