Workgroup: Internet Engineering Task Force
**draft-**wang-open-service-access-protocol-00
Published: 18 March 2021
Intended Status: Standards Track
Expires: 19 September 2021
Authors: B. Wang, Ed. S.P. Zhou, Ed. C. Li, Ed.
Hikvision Hikvision Guangzhou University
C.M. Wu, Ed. Z.Z. Wang, Ed.
Zhejiang University Zhejiang University

# Open Service Access Protocol for IoT Smart Devices

## Abstract

With the development of IoT(Internet of Things) technology, everything is interconnected. Mass IoT data, devices, businesses, and services adopt different data descriptions and service access methods, resulting in fragmentation issues, such as data heterogeneous, device heterogeneous, and application heterogeneous, which hinders the development of the industry. In order to solve the problem, this draft proposes the requirements for IoT smart devices to transmit and control, as well as transmission and protocol interfaces. It is for the program design, system testing and acceptance, and related research. Structured, unified, and standardized open service interconnection model reduces business replication cost and removes service barriers to push industrial development.

## Status of This Memo



## Copyright Notice

**Table of Contents**



## 1. Preface

With the development of the IoT technology, everything is widely interconnected, human-machine interact deep, including autonomous vehicles, telemedicine, smart factories, smart cities and other innovative applications. With the development of business, Mass IoT data, devices, businesses, and services adopt different data descriptions and service access methods, resulting in fragmentation issues, such as data heterogeneous, device heterogeneous, and application heterogeneous, which hinders the development of the industry, which mainly refers to:

1. Low value of data: IoT data has the characteristics of multi-source heterogeneity and huge scale, making it difficult for data analysis and sharing. At the same time, the lack of

business relevance between massive amounts of data leads to inefficient use of data.

2. High cost of business replication: different devices use different access standards. The cost of device access is too high and the time is too long. With the growth quantity of applications and devices, new device needs to be customized and developed multiple times for different standards, resulting in increased business replication cost.

3. Difficulty in industrial chain cooperation: There are different access protocols and data models between different manufacturers. The internal industrial chain has its own system, which makes it difficult for industrial chain to collaborate, for devices to be linked, maintained, for service to be compatible, Which seriously affects the user experience.

In order to solve the problem, this draft proposes the requirements for IoT smart devices to transmit and control, as well as transmission and protocol interfaces. It is for the program design, system testing and acceptance, and related research. Structured, unified, and standardized open service interconnection model reduces business replication cost and removes service barriers to push industrial development.

## 2. requirements for Consistency

### 2.1. Terms and Definitions

#### 2.1.1. Area

A set of related functions, which is business independent.

#### 2.1.2. Attribute

Used to describe the sustainable state of the devices during operation, which can be read and set.

#### 2.1.3. Operation

A method that can be called externally by a device or platform. The operation includes "input parameters" and "output parameters". The input parameters are the instruction information that needed to perform the operation, and the output parameters are the feedback information after the instruction is executed.

#### 2.1.4. Event

Information actively reported by the device. This type of information needs to be reported in real time and processed by the platform in time. If the device network is interrupted, it can be cached and reported after recovery.

#### 2.1.5. Resource

An entity that is a relatively independent component of the device and can independently handle user requirements. User applications can independently show or manage the resources of the device. For example, the video channel of NVR device.

#### 2.1.6. IoT Device Management Platform

A system that connects a large number of diverse and heterogeneous sensing devices and can unify access management of devices, collect, process and store data.

#### 2.1.7. Device Access Service

Services for managing, controlling and configing device functions and support attributes, operations, and events.

#### 2.1.8. IoT Smart Devices

Physical entities with video, image, and information perception capabilities, including: video equipment, access control, radar, etc. It can be directly connected to the IoT device management platform, or be a gateway that connects the agent sub-device and the IoT device management platform.

### 2.2. Abbreviations and Acronyms

| Abbreviations and Acronyms | Full Name |
|---|---:|
| JSON | Java Script Object Notation |
| MQTT[MQTT2016] | Message Queuing Telemetry Transport |
| TLS[RFC8446] | Transport Layer Security |
| UTF-8 | 8-bit Unicode Transformation Forma |
| URL | Uniform Resource Locator |

Table 1

## 3. Framework of Device Communication Protocol

The framework of the protocol is shown below:

```
Bussiness Protocol-----------------------------------------------+
| +------+ +-----+ +-------+        +-----+     +--------+    |
| |      | |     | |       |        |area|      |Resource|    |
| |store/| |Media| |Upgrate|        +-----+     +--------+    |
| |file  | |     | |       | +---------+ +---------+ +-----+ |
| |      | |     | |       | |attribute| |operation| |event| |
| +------+ +-----+ +-------+ +---------+ +---------+ +-----+ |
+-----------------------------------------------------------+

Fundamental communication protocol---------------------------+
| +--------------------------+                              |
| | +----------------------+ +---------------------------+ |
| | | store | media |update | | attribute|operation| event | |
| | |channel|channel|channel| |  channel | channel |channel| |
| | +----------------------+ +----^-----+----^---------^--+ |
| +--------------------------+    |          |         |    |
|      ^         ^       ^        |          |         |    |
|      |         |   +---+---------+----------+---------+--+ |
|      |         |   |                MQTT                 | |
|      |         |   +----------------^---------------^---+ |
|      |         |                    |               |     |
|      |         |                    |          +----+---+ |
|      |         |                    |          |   TLS  | |
|      |         |                    |          +----^---+ |
|      |         |                    |               |     |
|  +---+---+  +--+--------------------+---------------+---+ |
|  |HTTP(S)|  |                   TCP                     | |
|  +-------+  +-------------------------------------------+ |
+-----------------------------------------------------------+
```

Figure:Framework of Device Communication Protocol

The business is separated from the protocol. In the bottom layer, it adopts MQTT to transmit data. Different transmission channels are used for authentication, media, storage and attributes, operations, and events.

## 4. Interface protocol structure

In this draft, the session channel interface adopts MQTT protocol. Structure of MQTT protocol is divided into three sections: fixed header, variable header and payload. Structure of MQTT protocol is shown below.

| structre | Header | | Payload | | |
|---|---|---|---|---|---|
| | Fixedheader | Variableheader | GeneralPayload | | ApplicationPayload |
| name | Fixedheader | Variableheader | length | content | content |
| symbol | FixedHEADER | VariableHEADER | LEN | Gernal | Func |
| length | 2-5 bytes | variable | 2 bytes | variable | variable |
| des-cription | Depending on the length of the variable header and payload, the length of the fixed header varies between 2 and 5 bytes | Different control message has different variable headers | The length of general payload | See defi-nition for its format | The format depends on specific transaction |

Table: MQTT protocol structure

General protocols and business protocol bodies need AES (128) encryption during transmission, and UTF-8 encoding is used uniformly for character strings.

## 5. Device certification

The overall protocol format of the authentication process is shown as follows:

```
--------+--------------------------+------------------------------------
        |                          |
        |         Header           |            Payload
structre|-----------+--------------+-----------------+------------------
        |           |              |                 |
        |Fixedheader|Variableheader|GeneralPayload   |ApplicationPayload
--------+-----------+--------------+---------+-------+------------------
name    |Fixedheader|Variableheader|version  |con+   |
        |           |              |         |tent   |
--------+-----------+--------------+---------+-------+------------------
symbol  |FixedHEADER|VariableHEADER|PROTOCOL-|       |PROTOCOL-
        |           |              |VERSION  |Func   |VERSION
        |           |              |         |       |
--------+-----------+--------------+---------+-------+------------------
length  |2|5 bytes  |Variable      |3 bytes  |Va-    |3 bytes
        |           |              |         |riable |
        |           |              |         |       |
--------+-----------+--------------+---------+-------+------------------
des-    |Depending  |Different     |The      |See    |The
cription|on the     |control       |version  |tran-  |version
        |length of  |message has   |of       |saction|of
        |the        |different     |protocol |for its|protocol
        |variable   |variable      |         |format |
        |header and |headers       |         |       |
        |payload,   |              |         |       |
        |the        |              |         |       |
        |length of  |              |         |       |
        |the fixed  |              |         |       |
        |header     |              |         |       |
        |varies     |              |         |       |
        |between 2  |              |         |       |
        |and 5 bytes|              |         |       |
--------+-----------+--------------+---------+-------+------------------
```

Table: MQTT protocol format

The protocol version definition is shown as follows:

| name | type | description |
|---|---|---:|
| FORM_VERSION | char | version number of protocol form |
| HIGH*TYPE*VERSION | char | version number of protocol type(high) |
| LOW*TYPE*VERSION | char | version number of protocol type(low) |

Table 2: Protocol version definition

Device access adopts bidirectional negotiation protocol process. Devices sends the supported type of protocol group to the balance load service, and the server will determine which way to communicate depending on its own situation. After the device being authenticated, it can establish an MQTT connection with the device access service (Das) through the sessionkey to communicate with the bussiness protocol. The specific bidirectional negotiation diagram is as follows:

```
                                  +------+                   +------+
                                  |device|                   |server|
                                  +---+--+                   +------+
                                      |                          |
                                      |                          |
                                      |                          |
                                      |                          |
+-----------------------+----------+  |                          |
| negotiation request  |          |  |                          |
+-----------------------+          |  +--negotiation request---> +--+
|     Control message type:0x1   |  |                          |  |
|                                |  |                          |  |
|     Control flag:0x1           |  |                          |  |
|                                |  |                          |  |
|     protocol type:1            |  |                          |  |
|                                |  |                          |  |
|     protocol group:(1,2,4,8)   |  |                          |  |
|                                |  | <--negotiation response--+  |
|     transaction content:xxxxxx |  |                          |  |
+----------------------------------+  |                          +--+
                                      |                          |
                                      |                          |
+-----------------------+----------+  |                          |
| negotiation response |          |  |                          |
+-----------------------+          |  |                          |
|     Control message type:0x2   |  |                          |
|                                |  |                          |
|     Control flag:0x1           |  |                          |
|                                |  |                          |
|     protocol type:1            |  |                          |
|                                |  |                          |
|     transactionocontent:xxxxxx |  |                          |
+----------------------------------+  |                          |
```

Figure 1: bidirectional negotiation diagram - consistence

```
                    +------+          +------+
                    |device|          |server|
                    +---+--+          +------+
+----------------+-+    |                 |    +----------------+-+
| negotiation    | |    |                 |    | negotiation    | |
|  request2      | |    |                 |    |  response2     | |
+----------------+ |    |  negotiation    |    +----------------+ |
|Control           |    +--  request1 --> |    |Control           |
|message type:0x1  |    |                 |    |message type:0x2  |
|                  |    |                 |    |                  |
|Control flag:0x1  |    |                 |    |Control flag:0x1  |
|                  |    |                 |    |                  |
|protocol type:2   |    |                 |    |protocol type:2   |
|                  |    |                 |    |                  |
|protocol          |    |    negotiation  |    |protocol          |
|group:(1,2,4,8)   |    | <--  response1--+    |group:(2,4,8)     |
|                  |    |                 |    |                  |
|transaction       |    |                 |    |transaction       |
|content:xxxx      |    |                 |    |content:xxxx      |
+------------------+    |                 |    +------------------+
                        |                 |
+-------------------------------------------------------------------+
                        |  negotiation    |
 disconnect             +--  request2 --> |
                        |                 |
+----------------+-+    |                 |    +----------------+-+
| negotiation    | |    |                 |    | negotiation    | |
|  request2      | |    |                 |    |  response2     | |
+----------------+ |    |                 |    +----------------+ |
|Control           |    |                 |    |Control           |
|message type:0x1  |    |    negotiation  |    |message type:0x2  |
|                  |    | <--  response2--+    |                  |
|Control flag:0x1  |    |                 |    |Control flag:0x1  |
|                  |    |                 |    |                  |
|protocol type:2   |    |                 |    |protocol type:2   |
|                  |    |                 |    |                  |
|protocol          |    |                 |    |transaction       |
|group:(1,2,4,8)   |    |                 |    |content:xxxx      |
|                  |    |                 |    |                  |
|transaction       |    |                 |    |                  |
|content:xxxx      |    |                 |    |                  |
+------------------+    |                 |    +------------------+
                        |                 |
                        |                 |
                        +                 +
```

Figure 2: bidirectional negotiation diagram - inconsistence

bidirectional negotiation can be divided into two conditions:

(1) If the service supports this type of protocol, select the most secure protocol in the device's protocol group to complete the negotiation and communicate with the device;

(2) If the service does not support the type of protocol, return the message to the device,which contains the type of protocol and protocol group supported by the service. And then, interupt TCP connection. If the device supports it, use again the type of protocol and protocol group supported by the service to go through the authentication process. Otherwise, the device should give up authentication with the service.

In order to ensure forward compatibility with the ECDH key interaction mode, Bit1 of the control flag bit is enabled. When Bit1 is 0, the control message type remains in the original mode, and when Bit1 is 1, it means that the ECDH key mode is used for interaction. The key algorithm of secret key in the authentication process:

sharekey:pdkdf2*SHA256(md5(md5(MD5(verification code* + *device serial number)+www.88075998.com))) Device masterkey: ecdh*NID*secp384r1 (lbs*publickey, device*privatekey) Server masterkey: ecdh*NID*secp384r1 (device*publickey, lbs_privatekey)

a) First Authentication

When the device requires for working online the first time, useexchange algorithm of ECDH secret key to initialize DEVID and MasterKey. The process is shown as follows:

```
  +------+                                     +-----------+
  |Device|                                     |Lbs service|
  +------+                                     +------+----+
     |                                                |
+-----------------Privatekey exchange------------------------+
     +-----+                                          |
     |     |Generate privatekey and publickey         |
     <-----+                                          |
     +-----+                                          |
     |     |Generate sharekey                         |
     <-----+                                          |
     |-------Request for privatekey interaction------->
     <-------Reponse for privatekey interaction-------+
     +-----+                                          |
     |     |Decrypt lbs_publickey from response       |
     <-----+                                          |
     +-----+                                          |
     |     |Generate materkey                         |
     <-----+                                          |
 +-------------------- apply devid---------------------------+
     |-------| Request for applying devid |----------->
     <---------Reponse for applying devid-------------+
     +-----+                                          |
     |     |Decrypt devid and sessionkey from response|
     <-----+                                          |
     |---|Request for getting Das service address----->
     <----Reponse for getting Das service address-----+
     |                                                |
     +                                                +
```

Figure 3: First Authentication

b) Reauthentication

When the device is disconnected and ask for reauthenticated, it needs to request reauthentication from the platform and update the sessionkey. The specific process is shown as follows:

```
 +------+                                               +-----------+
 |Device|                                               |Lbs Service|
 +------+                                               +-----+-----+
    |                                                         |
+---------------------------------------------------------------+
    +------+ Update request for sessionkey +---------->
    |                                                         |
    <------+ Update respond for sessionkey +----------+
    |                                                         |
    +-----+                                                   |
    |     Decrypt sessionkey                                  |
    |     | from response                                     |
    <-----+                                                   |
+---------------------------------------------------------------+
    |                                                         |
    +---------+ Get service address request +--------->
    |                                                         |
    <---------+ Get service address response +--------+
    |                                                         |
```

Figure 4: Re-authenticate

c) Define the ECDH control message type as follows:

| message direction | control message | name | description |
|---|---|---|---:|
| Dev<--->Lbs | 0x1 | Authentication*ECDH*Req | request for ECDH exchange |
| Dev<--->Lbs | 0x2 | Authentication*ECDH*Rsp | response for ECDH exchange |
| Dev<--->Lbs | 0x3 | Rsrv | reserve |
| Dev<--->Lbs | 0x4 | Refresh*SessionKey*Req | refresh SessionKey request |
| Dev<--->Lbs | 0x5 | Rsrv | reserve |
| Dev<--->Lbs | 0x6 | Refresh*SessionKey*Rsp | refresh SessionKey response |
| Dev<--->Lbs | 0x7 | Authentication*apply*devid_Req | request device ID |
| Dev<--->Lbs | 0x8 | Authentication*apply*devid_Rsq | response device ID |
| Dev<--->Lbs | 0x9 | Authentication*apply*devid_Cfm | confirm device ID |

Table 3

Table: Protocol version definition

## 6.  Get access service

As the number of device accesses increases, there will be bottlenecks in the performance of single-node accesses, so the platform needs to support the mode of multiple device accesses. To support this mode, the devices are redirected to multiple access services by load balancing server. After the device obtains the sessionKey through two-way authentication, it initiates a request for access service within the same TCP connection, and the message in the request is encrypted with the sessionKey.

```
                        +------------+  +-------------+

+------+ |Balance load| |Device access| |Device| | ser^ice | |
service | +------+ +------+-----+ +------+------+ | | |
Redirect---------------------------------+ | | +--- F1:Request for
getting a -----> | | | | device access service address | | | | | | |
| | <--- F2:Return a device access ----+ | | | | ser^ice information
| | | +--------------------------------------+ | | | |
Bussiness----------------------------------------------------+ | |
| | | | <------- Business message:AES128(message)----------> | | |
AES128 privatekey:sessionkey | | | | | | |
+------------------------------------------------------------+ + +
+
```

Figure: Get access service

# Registration and Deregistration
After the device completes two-way authentication to obtain a specific a

```
+-------------+                                          +------------+
|   Device    |                                          |  Platform  |
|(MQTT Client)|                                          |(MQTT Sever)|
+-----+-------+                                          +------+-----+
      |                                                         |
      +------- F1:Device register and login(MQTT CONNECT) ------->
      |                                                         |
      <-------- F2:Register and respond(MQTT CONNACK) -----------+
      |                                                         |
      <---------------- Business interaction -------------------->
      |                                                         |
      +---- F3:Send request for disconnect(MQTT DISCONNECT) ----->
      |                                                         |
      +                                                         +
```

Figure 5: Registration and Deregistration

a) F1: After the device and platform network connection is established, the device sends a online request to the platform via

MQTTCONNECT, of which payload contains one or more encoded fields, including: unique identifier of the client, Will subject, Will message, username and password.

b) F2: The platform returns the response message to the device via MQTTCONNACK to inform it whether it succeeded or not;

c) F3: Before disconnecting, the device sends a DISSCONNECT message to the platform, indicating that it wants to disconnect normally, and the platform will close the TCP/IP connection after receiving the request.

## 7. Heartbeat

After the device has registered with the platform, it needs to send heartbeat requests periodically according to the heartbeat interval indicated in the registration request. The interval is usually 30s. Used for:

a) Inform the platform that the device is alive when no other control messages are sent from the device to the platform

b) Request the platform to send a response confirming that it is alive.

c) Use the network to confirm that the network connection is not disconnected.

```
+------+                                                    +--------+
|Device|                                                    |Platform|
+------+                                                    +----+---+
   |                                                             |
   +------ F1: Send a heartbeat request(MQTT PINGREQ) ----------->
   |                                                             |
   <--- F2: Respond to the heartbeat request MQTT PINGRESP) -----+
   |                                                             |
```

Figure 6: Heartbeat

## 8. Security Considerations

This entire memo deals with security issues.

## 9. IANA Considerations

This documents has no IANA actions.

## 10. Informative References

**[MQTT2016]**
ISOIEC, "Information technology — Message Queuing Telemetry Transport", <https://www.iso.org/obp/ui/#iso:std:iso-iec:20922:ed-1:v1:en>.

**[RFC8446]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

## Authors' Addresses

Bin Wang (editor)
Hikvision
555 Qianmo Road, Binjiang District
Hangzhou
310051
China

Phone: +86 571 8847 3644
Email: wbin2006@gmail.com

Shaopeng Zhou (editor)
Hikvision
555 Qianmo Road, Binjiang District
Hangzhou
310051
China

Phone: +86 571 8847 3644
Email: zhoushaopeng@hikvision.com

Chao Li (editor)
Guangzhou University
230 Wai Huan Xi Road
Guangzhou
510006
China

Email: lichao@gzhu.edu.cn

Chunming Wu (editor)
Zhejiang University
866 Yuhangtang Rd
Hangzhou
310058
China

Email: wuchunming@zju.edu.cn

Zizhao Wang (editor)

Zhejiang University
866 Yuhangtang Rd
Hangzhou
310058
China

Email: 22021272@zju.edu.cn