Efficient Compression Method Based on Object Model for Massive Real-time Data in Power IOT

Jixin Hou\textsuperscript{1,2}, JingHu Zhao\textsuperscript{1,2*}, Zidong Wu\textsuperscript{1,2}, Ying Fan\textsuperscript{1,2}, Xiaochao Wang\textsuperscript{1,2} and Jie Meng\textsuperscript{1,2}

\textsuperscript{1} NARI Group Corporation (State Grid Electric Power Research Institute), Nanjing, 211106, China
\textsuperscript{2} NARI Technology Co., Ltd., Nanjing 211106, China
Email: zhaojinghu@sgepri.sgcc.com.cn

Abstract. With rapid development of the Internet of things (IOT) in power field, equipment of various majors began to access the smart IOT system on a large scale. In this paper, a new data compression algorithm based on object model (CABM) is proposed to solve the problems of large-scale applications of power Internet of Things, such as high-frequency transmission of large quantities, real-time short data can not be effectively compressed, resulting in low efficiency of cloud edge information transmission and serious waste of bandwidth resources. The CABM adds a compression and decompression module between the IOT management platform and the edge agent; the IOT management platform automatically generates a compression dictionary based on the object model file and saves it synchronously in the edge agent. During data transmission, both the sender and receiver complete automatic compression and decompression according to the compression dictionary. The experimental results show that for real-time message data below 1KB, the compression rate using CABM algorithm is 3-5 times higher than that using GZIP compression algorithm; and the compression time is 10-20 times lower than that using GZIP; the efficiency using CABM algorithm is 4.5 times higher than that using GZIP algorithm; that is, CABM can reduce redundant data transmission between cloud edges and improve the efficiency of cloud edge transmission.

Keywords: Equipment modelling; power internet of things; smart IoT system; data compression

1. Introduction
In 2019, State Grid Corporation proposed the goal of building a smart IOT system [1]. Aiming at the lower-level business, the system addresses the IoT and perception needs of various fields, and realizes unified IoT management and standardized terminal access [2-3]. For the upper middle office and business system, the system provides standard data services to maximize information sharing. Eventually, an ecosystem of cross-professional data sharing will be formed, and basic collection support services will be provided for the construction of an observable, measurable, and controllable digital transparent power grid [4-6].

In the smart IOT system architecture, the packets transmitted through the cloud and side have the characteristics of small length and high transmission frequency [7-8]. Therefore, the data transmission mainly adopts JSON format [9]. The JSON format is simple, easy to use, and has strong self-description ability which is convenient for data interaction between heterogeneous systems, but it has the problem of low data transmission efficiency [10-11]. To solve this problem, Cao Zhou proposed a
compression algorithm suitable for large-scale complex nested JSON data, but this algorithm is not suitable for simple and high-frequency JSON objects or arrays [12]. Tang Dali, Xiong Jian et al [13], proposed that the data transmission controller component with compression and decompression function can effectively improve the data transmission efficiency for a large number of WebGIS data transmitted between servers and browsers, but the compression algorithm can not be effectively used in real-time and short JSON scenarios. In the traditional power protocol, on the one hand, there are the problems of single protocol function and difficult interaction, on the other hand, there are the problems of complex protocol configuration and use [14]. For example, IEC101/104 protocol mainly reduces description information transmission and improves transmission efficiency through point number mapping, but point number configuration and maintenance require a lot of manpower and are prone to errors [15-16]. Traditional dictionary compression and statistical compression algorithms are mainly applied to large blocks of data, which is difficult to effectively compress real-time JSON strings in the field of IoT [17-19]. Hossain K and Yang Long et al, proposed corresponding compression algorithms for sequential data storage, but they can not be applied to real-time data transmission [20-21].

This paper proposes a data compression algorithm based on object model (CABM) for JSON format messages transmitted at high frequency in smart IOT system. The CABM algorithm supports the description of business interaction in JSON format. On the premise of ensuring the simplicity and ease of business interaction, it can effectively improve the data transmission efficiency and reduce the production cost through automatic compression and decompression, so as to provide strong support for the large-scale application of smart IOT system in the power field.

2. Background

2.1. Smart IOT system

The smart IOT system includes IOT management platform, edge agents and various terminal devices. The IOT management platform mainly realizes the unified management of edge agents and terminal devices, and realizes the unified aggregation of various collected data, and provides reliable data sources for business systems or enterprise platforms in the form of interfaces and services. The edge agent realizes the access of various terminal devices, shields the differences of communication mode, message format and access mode of terminal devices, and uniformly accesses the IOT management platform. The smart IOT architecture is shown in figure 1.

![Figure 1. Equipment attribute modeling diagram.](image)

In the smart IOT architecture, each type of device corresponds to an object model file. The IOT management platform and edge agent are responsible for object model file synchronization and data verification. The IOT management platform unifies and standardizes the equipment interaction format through the object model to realize the unified management of massive equipment.
2.2. **Object Model**

Information modeling for primary and secondary power equipment so that the system can interact with equipment is one of the common techniques for information interaction between systems and between systems and equipment in power systems [22-23]. In the field of power IoT, the object model mainly focuses on the information modeling of power secondary equipment. According to the characteristics of secondary equipment, the object model defines the basic equipment information, equipment perception ability and external call method from the three dimensions of attribute, message and service. The attribute includes static attributes and dynamic attributes. Static attributes define basic information such as equipment manufacturer and model. Dynamic attributes refer to business data objects used in equipment interaction, such as voltage, current, stability, etc. The message define events that occur on the device, such as overcurrent events, phase failure events, etc. The service defines the control operations on the equipment, such as time synchronization, call test, etc.

According to the definition of the object model file, the message and service adopt JSON format, including the field name and data type of the data item. For example, one object model file of a temperature sensor (device ID 20210808) defines two data items "temperature" and "collection time", as well as two basic attribute items of device ID and message ID. The field name of device ID is "devId", the data type is string. The field name of message ID is "mId", and the data type is integer. The field name of "temperature" is "temp", and the data type is float. The field name of "collection time" is "time", and the data type is long. The object model is organized in JSON format as follows:

```json
{
    "devId": "20210808",
    "mId": 20008006,
    "temp": 26.7,
    "time": 1617846605
}
```

3. **The CABM Compression Algorithm**

The CABM compression algorithm designed in this paper acts on the "compression / decompression" module deployed in IOT management platform and edge agent to realize automatic data compression and decompression. The business system, edge agent and business collection app in the smart IOT system have no perception of the compression / decompression process. The CABM compression not only ensures the simplicity and flexibility in the process of system interaction, but also improves the data transmission efficiency, as shown in figure 2.

![Equipment attribute modeling diagram](image-url)
The CABM compression algorithm is mainly based on dictionary file for data compression, each field name line is encoded in the dictionary file, and the storage method is defined for each data type. The dictionary file is automatically generated according to the object model file, and the two correspond one by one. IOT management platform and edge agent ensure the consistency of dictionary files with the help of object model file synchronization mechanism.

3.1. Design mentality
The low efficiency of JSON format data transmission is mainly because the data contains a large number of redundant data such as attribute names, labels and symbols [24]. This paper defines the data efficiency, which is used to quantitatively analyze the bandwidth waste of JSON format data in the transmission process, and guide the optimal design of CABM algorithm.

For the data to be transmitted, L is used to represent the total transmission length, \( l \) value or \( l \) is used to represent the effective data length, and \( \eta \) is used to represent the efficiency of data transmission. The calculation formula is:

\[
\eta = \frac{l}{L} \quad (1)
\]

\( \eta \) reflects the proportion of actual useful data in the whole transmission bandwidth. The larger the \( \eta \), the higher the data transmission efficiency and the lower the waste of bandwidth resources. For JSON format strings with \( n \) attributes, the efficiency can be expressed as:

\[
\eta_n = \frac{l_n}{L_n} = \frac{\sum_{i=0}^{n} l_i(value)}{\sum_{i=0}^{n} (l_i(key) + l_i(value) + \delta l_i)} \quad (2)
\]

Where \( n \) represents the number of attributes, \( L_n \) represents the original total length, \( l_n \) represents the total length of valid data, \( l_i(key) \) represents the length of the ith attribute name, \( l_i(value) \) represents the length of the ith attribute value, and \( \delta l_i \) represents the length of JSON key characters required to increase the ith attribute. For example, in the JSON format data of temperature sensor in section 1.2, assuming that the number type occupies 4 bytes, its efficiency is \( \eta = \frac{l_4}{L_4} = \frac{20}{55} = 36.3\% \).

It can be seen from formula (2) that \( \eta \) which is represents the efficiency of data can be improved by reducing \( l_i(key) + \delta l_i \). The core idea of the CABM algorithm designed in this paper is to reduce the size of the \( l_i(key) + \delta l_i \) by shortening the length of the key and reducing the number of symbols.

3.2. Object Model Attribute Coding
Object model attribute is the basic unit of data to be transmitted. The CABM compression algorithm records the code of each attribute field name and the length of the code (KEY_LEN) in the dictionary file. The KEN_LEN is determined by the number of attributes in the object model file, such as when KEY_LEN is 1, 255 member attributes can be encoded at most. Taking the temperature sensor in section 1.2 as an example, the JSON format of its object model attribute code in the compression dictionary file is defined as:

```json
{
    "KEY_LEN": 1,
    "KEY_CODE": {
        "devId": 0,
        "mId": 1,
        "temp": 3,
        ...
    }
}
```
3.3. Data Type Coding

CABM compression algorithm divides data types such as string, float, int, array, structure and enumeration commonly used in the field of power Internet of things into four categories, as shown in Table 1.

| VALUE_TYPE | DATA TYPES IN THE OBJECT MODEL |
|------------|-------------------------------|
| STRING     | string                        |
| NUMBER     | Int, float, double            |
| ENUM       | Boolean, enum                 |
| COMP       | Struct, array                 |

### 3.3.1. String Type Encoding

The character string in the message can be stored in constant storage or stored directly. During constant storage, the ID of the string constant is directly stored after the key_id. The compression dictionary file stores all string constant codes of the current object model file, and its format is defined as:

```json
{
    "CONST_STR_LEN": 1,
    "CONST_STR_CODE": {
        "EVENT_DATA": 1,
        "U_C_unbalance": 2
    }
}
```

The “CONST_STR_LEN” is used to represent the length required to store constant encoding, in bytes. The “CONST_STR_CODE” records each constant and its corresponding encoding. The direct storage method first stores the field name code, then the string length, and finally the string content, as shown in Figure 3.

![Figure 3. Direct storage schematic diagram for string type.](image)

In the direct storage mode, the string length adopts variable length coding, with a minimum of 1 byte and a maximum of 4 bytes. When the highest bit of each byte is 1, it indicates that the next byte is still stored the string length. The coding range is shown in Table 2.

### Table 2. String length range.

| Byte | Start coding value | Max coding value          |
|------|--------------------|---------------------------|
| 1    | 0(0x00)            | 127(0x7F)                 |
| 2    | 128(0x80,0x01)     | 16383(0xFF,0x7F)          |
| 3    | 16384(0x80,0x80,0x01) | 2097151(0xFF,0xFF,0x7F) |
| 4    | 2097152(0x80,0x80,0x01) | 268435455(0xFF,0xFF,0xFF,0x7F) |
For example, if a data item "devId" with a data type of string is stored directly, the definition of the data item in the dictionary file is:

```json
{  "devId": {   "VALUE_TYPE": "STRING",   "CODE_TYPE": "DIRECT"  }}
```

3.3.2. Numeric type and enumeration type encoding. Numeric type and enumeration type directly store numerical content during compression; As shown in figure 4.

**Figure 4.** Storage schematic diagram for number and enumeration.

In the dictionary file, "VALUE_LEN" represents the number of bytes required to store the data item. For example, the data type of a data item "temp" is numeric, which is defined in the compression dictionary file as:

```json
{  "temp": {   "VALUE_TYPE": "NUMBER",   "VALUE_LEN": 4  }}
```

3.3.3. Composite Type Encoding. Composite type refers to the combined structure of a message composed of one or more sub members. Each sub member can be a string, number, enumeration and other basic types or composite types. When compressing composite type data, first store the field name code; and then store the "number of child members"; the sub member objects are then arranged in order, and the child members are separated by separators, as shown in figure 5.

| KEY_ID | NUMBER OF CHILD MEMBERS | CHILD MEMBER 1 | SEPARATOR | ---- | CHILD MEMBER n | SEPARATOR |
|--------|-------------------------|----------------|-----------|------|----------------|-----------|

**Figure 5.** Storage schematic diagram for Composite type.

The number of sub members adopts variable length coding, with a maximum length of 4 bytes. For example, if the field name of a data item is "devs" and the data type is composite type, the data item is defined in the compression dictionary file as:

```json
{  "devs": {   "VALUE_TYPE": "COMP",   "SEPARATOR": ","  }}
```

The compression algorithm of composite type message is shown in table 3.

```
setCompData(key, valObject, buf){
    keyCode = getKeyCode(key, dic);
}
```

**Table 3.** Compression algorithm for composite type
separator = getSeparator(key, dic);
sub_num = getSubNum(valObject);
setBuffer(keyCode, buf);
setBuffer(sub_num, buf);
foreach(elem: valObject){
    if (key is ARRAY) {
        setBuffer(elem, buf);
        setBuffer(separator, buf);
        continue;
    }else if (elem.key is STRING)
        setStringData(elem.key, elem.value, buf);
    else if (elem.key is NUMBER)
        setNumberData(elem.key, elem.value, buf);
    else if (elem.key is ENUM)
        setEnumData(elem.key, elem.value, buf);
    else if (elem.key is COMP)
        setCompData(elem.key, elem.value, buf);
    else
        goto error;
    setBuffer(separator, buf);
}error:
return;
}

4. Experiment Validation
In order to verify the effectiveness of the CABM algorithm, this paper compares it with GZIP algorithm in three indexes: effective rate, compression rate and compression time. When verifying the effective rate and compression rate, 10 groups of JSON objects with different lengths are selected and compressed by CABM and gzip algorithms respectively. The results are shown in table 4.

| Original length / byte | Effective length / byte | CABM compressed / byte | GZIP compressed / byte | CABM compression rate | GZIP compression rate | CABM effective rate | GZIP effective rate |
|------------------------|-------------------------|------------------------|------------------------|-----------------------|-----------------------|---------------------|---------------------|
| 242                    | 78                      | 61                     | 270                    | 0.252                 | 1.116                 | 1.279               | 0.289               |
| 319                    | 98                      | 76                     | 320                    | 0.238                 | 1.003                 | 1.289               | 0.306               |
| 407                    | 118                     | 91                     | 360                    | 0.224                 | 0.885                 | 1.297               | 0.328               |
| 489                    | 138                     | 106                    | 398                    | 0.217                 | 0.814                 | 1.302               | 0.347               |
| 550                    | 158                     | 121                    | 426                    | 0.22                  | 0.775                 | 1.306               | 0.371               |
| 620                    | 178                     | 136                    | 472                    | 0.219                 | 0.761                 | 1.309               | 0.377               |
| 708                    | 198                     | 151                    | 504                    | 0.213                 | 0.712                 | 1.311               | 0.393               |
The effective rates of GZIP and CABM when the JSON data length gradually increases is shown in figure 6.

It can be seen from the figure that the effective rate of CABM algorithm is much greater than that of gzip, and the effective rate of CABM algorithm is always greater than 1, that is, transmitting CABM compressed data occupies less bandwidth than transmitting original effective data.

The compression rate of GZIP and CABM when the JSON data length gradually increases is shown in figure 7. According to the graph, the compression rate of CABM algorithm is much lower than that of gzip compression algorithm, and the compression rate of CABM algorithm is very stable and less affected by the change of JSON data length. GZIP compression algorithm is greatly affected by JSON data length, When the length of JSON data is small (less than 319 bytes), the compression rate of GZIP algorithm is greater than 1, which means the compressed data is longer than the original data.

---

**Figure 6.** Comparison for effective rate.

**Figure 7.** Comparison for compression rate.
When comparing the compression time of the two algorithms, JSON with a length of 550 bytes is selected as the object for cyclic compression. The number of cycles ranges from 5000 to 100000, with an increase of 5000 each time, a total of 20 groups. The test situation is shown in figure 8. It can be seen from the figure that with the increase of compression times, the increasing trend of compression time of CABM is much lower than that of GZIP, indicating that this algorithm has higher performance in massive real-time data transmission.

![Figure 8. Comparison for compression time.](image)

Through the analysis of the above test results, it can be seen that CABM is much better than GZIP algorithm in data effective rate, compression rate and compression time, and the test results of the three indicators are very stable. In conclusion, CABM is more suitable for real-time, high-frequency and short message data transmission scenarios in the field of power IoT than GZIP.

5. Conclusion
In this paper, a CABM compression algorithm is proposed for the message transmission in the power IoT scenario. The CABM algorithm can not only meet the requirements of business system to transmit data in JSON format, but also improve the efficiency of data transmission and reduce production costs. The experimental results show that compared with GZIP algorithm, the CABM has obvious advantages in effective rate, compression rate and compression time for high-frequency, short and real-time JSON format message data. The CABM algorithm is not only applicable to the scenario of high-frequency transmission of short messages in the field of power Internet of things, but also can be applied to the scenarios sensitive to bandwidth and traffic in the fields of industrial Internet of things, consumer Internet of things and so on. However, it should be noted that CABM is not a general data compression algorithm; it depends on the object model file and its synchronization mechanism. In the system without object model, we can also refer to the idea of this scheme to realize similar compression and decompression mechanism.

References
[1] Li Q H, Zhang Y J, Chen J Q, Yi Y Q, et al. 2020 Development patterns and challenges of ubiquitous power internet of things Automation of Electric Power System 44(001): 13-22.
[2] Zeng M, Wang Y Q, Li M Z, et al. 2019 Preliminary study on the architecture and implementation plan of widespread power internet of things Smart Power 047(004): P.1-7.
[3] Lv J, Luan W P, Liu R L, et al. 2018 Architecture of distribution internet of things based on widespread sensing & software defined technology Power System Technology 042(010): 3108-3115.
[4] Sun H B, Guo Q L, Pan Z G. 2015 Energy internet: concept, architecture and frontier outlook Automation of Electric Power Systems 000(019):1-8.
[5] Wang Y, Su B, Zhao H B 2010 The concept and development trend of power Internet of Things Telecommunications Science S3: 9-14.
[6] Li X, Gong Q W, Qiao H. 2010 The application of IOT in power systems Power System Protection and Control 38(22): 232-236.
[7] Hou J, Zhao J, Lv F, et al. 2020 Design scheme for data transmission component of electric internet of things management platform 2020 IEEE/IAS Industrial and Commercial Power System Asia (I&CPS Asia).
[8] Chen Y B, Liu J Y, Chen J J 2017 Applications of intelligent energy internet of things ZTE Technology Journal 23: 37-42.
[9] Org J 2006 The application/json media type for javascript object notation (JSON) RFC 4627.
[10] Zhu F 2016 Application Research of Internet heterogeneous data integration based on JSON Nanjing University of Posts and Telecommunications.
[11] Liang Z H, Lu G H, Huang Y X, et al. 2018 Research on heterogeneous data integration based on JSON Computer Science and Application 008(003): 314-322.
[12] Cao Z 2015 Research and Implementation of Mobile Data Transmission Optimization Scheme Based on Data Serialization and String Compression Ocean University of China.
[13] Tang D L, Xiong J 2006 Scheme of webGIS large quantity of data transformation based on compressive transformation component Computer Engineering and Design 027(006): 1048-1051.
[14] Wu Z X, Alioasha Y, Zheng A, et al. 2016 Communication protocol design of power user electric energy data acquisition system based on object-oriented technology Electrical Measurement & Instrumentation 24: 69-74.
[15] Lei S Y, Liang Z F, Zhu Q C 2006 Application of IEC101/104 protocol in ecological small hydropower integrated LCU Microcomputer Information 04: 226-228.
[16] Zhang J S, Ma W Q, Guo J Y 2003 Application of IEC 60870-5-104 protocol in telecontrol communication Automation of Electric Power Systems 27(11): 91-93.
[17] Zheng C F 2011 Research of several common lossless data compression algorithms Computer Technology and Development 21(009): 73-76.
[18] Zhang W M 2002 Information System Modelling Electronic Industry Press.
[19] Ziv J, Lempel A 1997 A universal algorithm for data compression IEEE Transactions on Information Theory 23(3): 337-343.
[20] Hossain K, Rahman M, Roy S 2019 IoT data compression and optimization techniques in cloud storage: Current prospects and future directions International Journal of Cloud Applications and Computing 9(2): 43-59.
[21] Yang L 2013 Research on Real-time Data Compression Algorithm based on RTDB Nanjing University of Posts and Telecommunications.
[22] Wang Q, Deng L J, Cheng X X, et al. 2016 Data exchange model between the system of internet of things and distribution automation system based on IEC61968 and SG-CIM. Electric Power Information and Communication Technology 014(010): 28-33.
[23] Wang D W, Di J, Zhang C M 2012 Information modeling and implementation for status monitoring ied in substation based on IEC 61850 Automation of Electric Power Systems 36(003): 81-86.
[24] Gao J, Duan H C 2011 Research on data transmission efficiency of JSON Computer Engineering and Design.