# **Unified instruction set, multiple communication protocols, optimized for various scenarios**

1. Overview of Metaverse Communication Protocol

The meta-universe communication protocol is a set of simple, easy-to-use, stable and reliable communication protocol. It can obtain information or transmit data through the meta-universe communication protocol. With the support of the meta universe communication protocol, it has more abundant application scenarios.

The latest update:

Meta universe basic API interface (you can obtain other node information through this server to traverse the meta universe):

<http://42.194.159.204:8081/api>

API server open source code:

<https://gitee.com/hayoou/metaverse_api_server_go>

<https://github.com/hayooucom/metaverse_api_server_go>

Special interface, support text and basic json output:

Any interface sends get\_meta\_api\_info:text or {"get\_meta\_api\_info":"json"} or url is added and then the query field &get\_meta\_api\_info=text or json is added to obtain basic API text or json format information. Example 2.5.2 Chapter 0x00 instruction content:

meta\_api\_ver: Supported meta universe protocol version,  
id: meta\_example\_id, custom id or meta universe unified object id  
name: object name,  
meta\_api\_class\_name: meta-universe unified object classification name,  
meta\_api\_class\_id: Meta universe unified object type id,  
info\_url: introduction link\*,  
api\_info: Brief description of API interface (web page, human readable),  
api\_url: API interface\* (URL form, optional),  
...  
The above newline characters are unified as \n, and what json returns is the json format of the above fields. If you need to shield the above-mentioned strings contained in the forwarded data or only respond to recent interactions, it is recommended that machines with time synchronization only respond to the above-mentioned commands with a timestamp (string type) or timestamp self-increment. For example, get\_meta\_api\_info:text,time\_stamp:111111111,CRC\_16:AB12. Similarly, it is recommended that the requester carry the above information.

The communication protocol defaults to binary data stream transmission. If you need JSON format package protocol to send, use the following example: msg\_base64 (the base64 encoding of the protocol packet binary stream) must implement parsing support, DATA\_base64 is the base64 encoding of the data segment, and the sender can choose msg\_base64 Or send in the field below. The lite version is the same as the standard version.

{  
 "meta\_api\_ver": "1.0",  
 "msg\_base64": "",  
  
 "SOF": 59994,  
 "Version": 16, ...  
 "DATA\_base64": "" ...  
  
}

JSON

There are two data packet formats of the Metaverse Communication Protocol, one is a simplified version, with the version number being 0~3, the other is a standard format, and the version number is >=16. You can select the corresponding version according to actual needs. The standard version is suitable for large-scale data transmission in a high-quality channel environment. It is recommended to use a state machine for receiving logic control for protocol processing.

Note: Not specified, the low byte of the data in the protocol comes first

# **2. Description of Metaverse Communication Protocol**

## **2.1 Simplified protocol data format**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| SOF  Start frame header | Ver\_Length  Version/frame length | SEQ  serial number | CmdSet\*  Instruction Set | CmdID\*  Instruction ID | DATA  data | CRC-16/32  16-bit parity |
| 1-Byte | 1~2-Byte | 0~2-Byte | 0~2-Byte | 0~2-Byte | n-Byte | 2~4-Byte |

## **2.2 Field description of the simplified version of the agreement**

|  |  |  |
| --- | --- | --- |
| field | size | describe |
| SOF  Start frame header | 1 | The sender frame header is 0xFA  The response frame is 0xFB |
| Ver\_Length  Version/frame length | 1~2 | Version 0, 1:  [7:6]-Version number  [5:0]-The length of the data frame, up to 64 bytes (including the start frame header)  Version 2:  [15:14]-Version number, default is 0B10  [13]-Encryption flag (default AES256)  [12]-reserved  [11:0]-The length of the data frame, up to 4096 bytes (including the start frame header)  High byte first |
| CmdSet\*  Instruction Set | 0~2 | Command set\*  Version 0: 0 bytes  Version 1: 1 byte  Version 2: 2 bytes |
| CmdID\*  Instruction ID | 0~2 | Command code\*  Version 0: 0 bytes  Version 1: 1 byte  Version 2: 2 bytes |
| SEQ  serial number | 0~2 | Version 2 uses 2 bytes:  [15]-All data packet transmission is completed  [14:0]-Serial number, each time a different command is transmitted or a different command is returned, the serial number is set to 0 |
| DATA | n | The data segment is described below |
| CRC-16 | 2~4 | Frame check (whole frame)  Version 2 is CRC32 (4 bytes) |

## **2.3 Standard version protocol data format**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SOF  Start frame header | Watch  version number | DataType  type of data | Length  Data length | CmdType  Instruction type | ENC  encryption | CmdSet  Instruction Set | CmdID  instruction | Reseved  Reserve | Extend  Extension byte | SEQ  serial number | CRC-16  check | DATA\* | CRC-32 \*  Full frame check |
| 2-Byte | 1-Byte | 1-Byte | 2-Byte | 1-Byte | 1-Byte | 2-Byte | 2-Byte | 2-Byte | Ex\_len-Byte | 2-Byte | 2-Byte | n-Byte | 4-Byte |

## **2.4 Standard version protocol field description**

|  |  |  |  |
| --- | --- | --- | --- |
| field | Offset | size | describe |
| SOF  Frame header | 0 | 2 | The frame header is fixed to 0xEA5A, according to the data flow sequence is 0x5A, 0xEA |
| Version  version number | 2 | 1 | [7:6]-reserved, default 0  [6:0]-The version number is 16~63, the default is 16 |
| DataType  type of data | 3 | 1 | 0: text  1: Binary (default)  2: Use XML format to deliver  3: Use JSON key-value format to pass  4: Use protobuf for data interaction, schema description  5: Use protobuf for data exchange data segment |
| Length  Data length | 4 | 2 | The length of the data frame n (including the start frame header)  It is not recommended that a data packet exceeds 1024 bytes |
| CmdType  Instruction type | 6 | 1 | [4:0]-Response type  0-No response is required after data is sent  1-A response is required after the data is sent, but it doesn't matter if you don't respond  2-31- You must respond after data is sent  [5]-Frame type  0-command frame  1-Reply frame  [6] -When responding to the frame, the error flag, the data segment is error code and error information, json format, {"error\_code":1,"error\_msg":""}  [7]-Leave the default as 0 |
| ENC  encryption | 7 | 1 | [7:5]-Encryption type  0-no encryption  1-AES256 encryption  [4:0]-Supplementary byte length when encrypting (encryption must be aligned to 16 bytes) |
| CmdSet | 8 | 2 | Command set\* |
| CmdID | 10 | 2 | Command code\* |
| Reseved  Reserved byte | 12 | 2 | [15:4]-reserved, default 0  [3:0]-Extended byte length Ex\_len |
| Extend | 14 | Ex\_len | Extension byte  Version 16 defaults to 0 |
| SEQ | 14 + Ex\_len | 2 | serial number  [15]-All data packet transmission is completed (individual packet, this disposal 1)  [14:0]-Serial number, each time a different command is transmitted or a different command is returned, the serial number is set to 0 |
| CRC-16 | 16 + Ex\_len | 2 | Header check |
| DATA | 18 + Ex\_len | n | The data segment is described below |
| CRC-32 | 18+n+  Ex\_len | 4 | Frame check (whole frame) |

The currently planned use agreement versions are:

|  |  |  |  |
| --- | --- | --- | --- |
| project | Communication Interface | Protocol version implementation | describe |
| Internet of Things | Serial interface, CAN, etc. | 1, 2 (required)  0, 16 (optional) | Convenient and streamlined data transmission  It is recommended to implement the version 16 protocol |
| the Internet | http, socket, etc. | 2,16 (required)  0~1 (optional) | For basic data flow, version 16 protocol is preferred. Large-scale data only uses version 16 protocol. |

## **2.5 Detailed description**

### **2.5.1 Command set and command ID**

For related instruction design, please pay attention to avoid designing the return value of 1 byte, which is easy to be confused with the error code.

The command set and command code list used are shown in the following table:

Default instruction set: All devices implement the interaction of this instruction set.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| project | Instruction set implementation | describe |
| Internet of Things | 0x00  (Achieve basic functions\*) | Unified general instruction set, some functions and attributes that are not available can not be realized\* |
| the Internet | 0x00  (must) | Unified general instruction set |

### **2.5.2 Instruction set 0x00 design content:**

|  |  |
| --- | --- |
| CmdID | describe |
| 0x00 | Get object information\*  send:  [0] byte, binary:  0x00-request human-readable data (must be implemented, UTF-8 encoding)  0x01-Request JSON structured data (must be implemented, UTF-8 encoding)  0x02-Request XML structure data  0x03-Submit a request in JSON format, the subsequent data is a json string, and the request result is JSON data  0x04-Submit its own information in JSON format, and the subsequent data is a json string  [1-n] byte field\_name (optional): specially selected field data, string type, padded with 0x00 at the end  [1-n] Byte json string (optional): query request in json format (extended)  return:  Provide at least the following field information:  meta\_api\_ver: Supported meta universe protocol version, default 1.0  id: object ID, up to 512 bytes  name: object name  meta\_api\_class\_name: Meta universe unified object classification name (up to 512 bytes, registered with group collaboration documents )  meta\_api\_class\_id: Meta universe unified object type id (maximum 512 bytes, same as above, register through collaboration documents, apply for unified object type id identifier ), same type id, interface and interaction method are consistent  info\_url: introduction link\*  api\_info: Brief description of API interface (web page, human readable, required)  api\_url: API interface\* (URL form, optional)  get\_meta\_api\_info:json If the requester receives this field, it needs to submit its own information through the 0x04 instruction  More custom information can be added |
| 0x01 | Request to retransmit the data frame with the specified sequence number  According to the carried data, point out the corresponding sequence number and request the sender to retransmit  send:  Command set: same as receiving frame  Command ID: the same as the received frame  Sequence frame: 0  Data segment: [0-1] byte: serial number |
| 0x02 | Request all data retransmission\*  According to the carried data, request the sender to retransmit all the instruction data.  send:  Command set: same as receiving frame  Command ID: the same as the received frame  Sequence frame: 0 |
| 0x03 | Data reception/execution error, used for the simplified version of the protocol\*  Send from the device:  [0] Byte: Error code  [1] Byte: Wrong instruction set executed  Byte: the instruction code of the execution error |
| 0x04 | Request access\*  0x00: request access  return:  0x01: Accept the request  0x02: Reject the request |
| 0x05 | Encryption request  [0] Byte:  0x00: request encrypted data  [1~9] Byte: Encryption algorithm identification, "AES256", etc., padded with 0 at the end  [10~42] Byte: The key can be sent in the first handshake (not absolutely secure)  return:  0x01: Accept the request (update the key key)  0x02: Reject the request |
| 0x06 | Connection topology traversal  [0] Bytes: Number of queries  [1-4] Byte: Offset  [5]Byte: Query method:  0x00: connection creation time  0x01: id sort  [6] Byte: Whether to select only internal objects (objects generated within the object)  0x00: No, view all connected objects  0x01: Yes, only filter internal objects  Return, json data structure:  total\_connector: the total number of object connections  List, output the following information for each object:  id: connected object id  meta\_api\_class\_id: unified meta universe type id of the connected object  api\_url: API interface  meta\_api\_ver: Supported meta universe protocol version  route\_counter: routing times, relay jump times  Test: http://42.194.159.204:8081/api?do=get\_nodes&limit=10&offset=0 |
| 0x07 | Query the id information of the specified object  Request data in json format (all are of string type):  data\_type:  0x00-request human readable data  0x01-Request JSON structure data  0x02-Request XML structure data  field\_name: specially selected field data, string type, filled with 0x00 at the end  object\_id: Query the specified object id, a string, and fill with 0x00 at the end  meta\_api\_class\_id: unique category id of the meta universe  limit: query quantity string  offset: offset string  return:  Query the information of the object and forward the execution result of the query instruction  Test: http://42.194.159.204:8081/api?do=search\_nodes&object\_id=meta-api-server-id-42.194.159.204&field\_name=&meta\_api\_class\_id=&limit=10&offset=0 |
| 0x08 | Disconnect\*  0x00: Disconnect directly  0x01: delete the connection information and then disconnect |
| 0x09 | Request agent to transfer data  Send json format data packet: including url, timeout, post\_data, id (target id), data\_packet\_base64 (all contents of the data frame, base64 encoding, including cmd\_set, cmd\_id, data and other information).  return:  Data results requested by the agent |
| 0x10 | Host mode, do not disturb\*  The requesting object does not send out any information unless requested by itself  Send: 0x00: do not disturb, 0x01: return to normal mode  Response: 0x00: success, 0x01: failure |
| 0x11 | Get the prototype of the supported Metaverse API standard definition , the default json format  Sending: 0x00: json format, 0x01: xml format |
| 0x20 | Get the properties of the interactive device (without this property, there is no need to implement related interfaces):  Send: [0] Byte: 0x00 to get interactive attributes, 0x01 is json format  [1-2] Byte: device id, 0x00 or leave blank to query any device  answer:  Device side sending (can send actively): refer to General\_input\_device\_interface\_data\_define\_ general input device interaction interface definition  [0-1] Byte: device id  [1-3] Bytes: data attributes  1-keyboard 10 bytes  2-Mouse 10 bytes  3-Acceleration sensor 12 bytes (X (2 bytes), Y, Z, FREQ, SEQ, TIME (time stamp after device startup, milliseconds 4 bytes))  4-Gyro sensor (reference acceleration sensor)  5-Gamepad (refer to the definition of USB gamepad)  6-steering wheel  7-Shooting game toy gun simulation equipment  [4-14] Byte: interface type, USB, socket, etc.  [15-55] Byte: Interface parameters (USB\_ID, socket ip:port, etc.) |
| 0x21 | Interactive device data (sent by the device side)  [0-1] Byte: device id  [1-n] Byte: interactive device data (default is the data format defined by USB)  [n+1-n+5] bytes: timestamp after device startup, 4 bytes in milliseconds |
| 0x22 | Get a list of object ids that can be interacted with, in JSON format  List, bind the following properties in a group:  target\_id: the id bound inside the current object, 4 bytes, there will be changes as the digital environment (scene id) changes  id: the id of the current object (digital environment suppliers and creators should try to avoid duplication of this id) |
| 0x23 | Interactive/interactive interface, binary mode:  Interactive/interactive interface, binary mode:  [0-3] Byte: own target\_id  [4-7] Byte: target\_id of the interactive target  [8-9] Byte: interact\_type interaction type  1-Perform interaction  2-Try to interact, not cause major damage or remove data, etc.  3-Cancel interaction  4-Forced interaction  [10-11] Byte: interact\_function interactive function  0~1000: Default function, see: [Interaction\_function\_list](https://thoughts.aliyun.com/share/6195ddf08fd3d9001a4f67b5#title=Interaction_function_list)  1000~2000: Custom function  [12-n] Byte: interact\_data interactive data  json format:  Carry meta\_id and the above data |
| 0x24 | Scene update notification  Command: The target\_id of 0x22 is invalid and needs to be re-acquired  send:  [0-7] Byte: scene id (required)  [8] Byte: Number of invalidations, 0-all invalids, 1-n invalids, no more than 250, 255-invalid id list in JSON format  [9-4\*n+9] bytes: n invalid target\_id, interactive target id  Or [9-n+9] bytes: invalid id list JSON format |
| 0x25 | Scene switching, both parties can send scene id for switching interaction  send:  [0-7] Byte: scene id (required)  [8] Byte: scene type  0-Temporary scene  1-For subsequent available scenes, it is recommended to cache the object data  2-Normal scene, it is recommended to save to disk  3-Permanent scene, it is recommended to store to disk  [9] Bytes: the amount of scene change, processing demand  0-almost no change (smaller load)  1-Processing supported by mobile phone  2-It is recommended to use a computer for processing  3-High-performance computer, scene with display acceleration  4-Special high-performance computer such as RTX2080 graphics card  5-High-performance service cluster  [10-13] Byte: Bandwidth requirement: bps  [14-21] Bytes: Storage requirements: Bytes  [22-29] Bytes: Memory requirements: Bytes  [30-33] Bytes: Estimated number of interactive objects  [34-37] Bytes: Estimated number of objects  [38-45] Byte: unix timestamp ms |
| 0x26 | Scene information acquisition:  send:  [0] Byte: Function 0-traverse scene id, 1-get specified scene id information, 2-get scene information in JSON format, specified query needs to be accompanied by scene id  [1-8] Byte: scene id (optional)  return:  Function 0 traverse the scene id list, 8 bytes/a  Function 1 Data corresponding to command 0x25  Function 2 JSON format, information set list, data corresponding to instruction 0x25 (data\_base64 field) + additional information such as title, meta\_scenes\_id meta\_scenes\_id meta universe unified scene id, etc. |
| 0x27 | Debug mode  [0-1] Byte: debug function, 0-exit debug mode, 1-debug mode, 2-debug mode command line  [2-n] Byte: Debugging data (customized by the developer)  Return: Debug data |
| 0x28 | Interactive data range calibration. Ensure that the data range of the current interactive device/tool ​​is consistent. After the calibration starts, the user operates the input device from the smallest to the largest range. The calibration time is recommended not to exceed 20 seconds  Send: [0] Byte: 0-Exit calibration mode 1-Start calibration  Response: [0] Byte: 0-Calibration failed 1-Calibration completed 3-Calibration progress  [1]Byte: Calibration progress% |
| 0x29 | Suitable for large-scale batch data packet transmission of streamlined protocol version 1, 2, supplementary part, optional implementation  Note: Regular use of version 1 for small-scale data transmission, and version 2 for slightly more data transmission  Notification to turn on bulk transmission:  [0] Byte: batch transmission control 0-start batch transmission, 1-end batch transmission, 2-query receiving status (return the following data, the offset is the same), 3-synchronization sequence number (the receiver checks whether it has received this The serial number, if it is missing in the previous period, use the 0x01 command to request retransmission of the data frame with the specified serial number)  [1-2] Byte CmdSet,  [3-4] Byte CmdID,  [5-8] The number of byte data packets (can be set to 0 when opening, set when ending),  [9-12] Byte CRC-32 of all data packets in this batch,  [13-16] Byte transmission batch id  [17-20] Byte sequence number |
| 0x30 | Authentication request, hoping to verify whether the object is trustworthy and whether it matches the public key on the network  Send: [0] Bytes: 0-close/cancel authentication, 1-start authentication, 2-get mata\_api\_public\_key  [1-100]Byte: String to be authenticated  Reply: [0] byte response code  [1-n] bytes: encrypted string |
| 0xF0 | Switch machine (equipment/object may not be executed)\*  0x00: request to close  0x01: Closing is complete (reply)  0x02: request to start  0x03: Startup completed (reply)  0x04: request to enter standby mode  0x05: has entered standby mode (reply) |
| 0xF1 | Restart (device/object may not be executed)  1: Restart  2: Restart complete (reply) |
| 0xF3 | Prompt information\*  The connected party sends related prompt information, such as alarm, shutdown, ready to disconnect, etc.  [0] Byte: Importance 0-not important, 1-corner prompt, 2-prompt, 3- more important, centered prompt, 4-need to confirm, 5-need to read carefully  [1-n] bytes: prompt information |

The above response/reply data content needs to be packaged through the communication protocol.

Instruction set application:

Reserved instruction set: 0x00-0x6F, 0x80~0xFF, 0x8000~0xFFFF

To add a new instruction set, please register through the collaboration document .

The manufacturer/creator can customize the 0x70-0x7F instruction set without submitting an application. It is recommended to implement an open support for a broad instruction set.

### **2.5.3 Default response return code**

Currently, the supported return codes are shown in the following table:

For data packets containing a sequence number, the sequence number needs to be set in the frame header when responding

|  |  |
| --- | --- |
|  |  |
| Error code value | meaning |
| 0x00 | The instruction is executed successfully |
| 0x01 | Instruction parsing error |
| 0x02 | Command execution failed |
| 0xFF | Undefined error |

# **3.CRC check**

The CRC16 and CRC32 parameters used in the data packet are as follows

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Name | Width | Poly | Init | RefIn | RefOut | XorOut |
| CRC16 | 16 | 0x8005 | 0xc55c | True | True | 0x0000 |
| CRC32 | 32 | 0x04c11db7 | 0xc55c0000 | True | True | 0x00000000 |

## **3.1 CRC code example**

CRC16 used in this agreement can refer to: custom\_crc16.c, custom\_crc16.h

CRC32 used in this agreement can refer to: custom\_crc32.c, custom\_crc32.h

Link: https://pan.baidu.com/s/15PgpC4OkKF3XO\_Iv4XAKjA

Extraction code: 5566

# **4. Open source code**

## **4.1 Communication protocol basic interface and code example**

Meta universe basic API interface (you can obtain other node information through this server to traverse the meta universe):

<http://42.194.159.204:8081/api>

Example:

Get basic information about API:

<http://42.194.159.204:8081/api?get_meta_api_info=text>

Get connected node information:

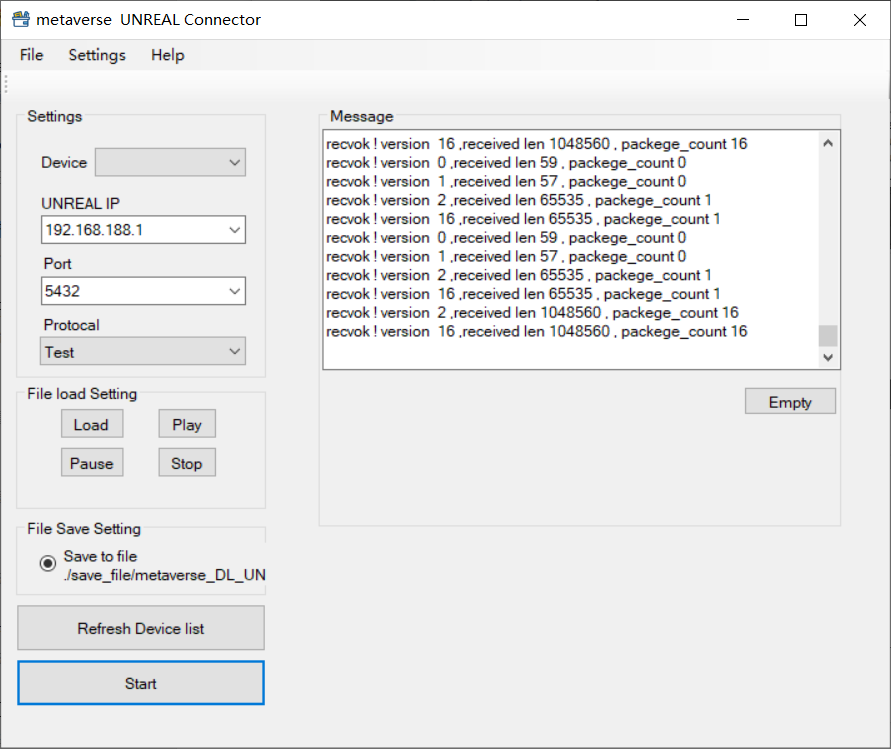
<http://42.194.159.204:8081/api?do=get_nodes&limit=10&offset=0>

Search node information: test

API server open source code:

<https://gitee.com/hayoou/metaverse_api_server_go>

## **4.2 Communication protocol code example**



[C# implementation of meta-universe communication protocol](https://gitee.com/hayoou/meta_api_protocal_c_sharp)

﻿[Metaverse API basic information primitive description(schema) V1.0](https://thoughts.aliyun.com/share/61988f274cbd0a001a03bc68#title=Metaverse_API_basic_information_primitive_description(schema)_V1.0)

Homepage:

[Metaverse standard API Metaverse standard API](https://thoughts.aliyun.com/share/61988d0bbdc2c4001aea00b9#title=Metaverse_interface_standard)