Draft Standard

NeXt group 'NXg SW003'

Version: 1.0.0

Date: 2025-12-31

NXg CANopen-Lift Connector

History

2025-12-31	Publication of draft standard
2025-03-05	Publication of work draft for discussion

Foreword

NeXt group (NXg) e.V. is a nonprofit users' and manufacturers' group. The work of preparing NXg specifications is normally carried out through NXg Special Interest Groups (SIG). Each NXg member, interested in a subject for that such a group has been established, has the right to be represented on that group.

The procedures used to develop this document and those intended for its further maintenance are described in the NXg rules. This document was drafted in accordance with editorial rules similar to those used by ISO and IEC.

Attention is drawn to the possibility that some of the elements of this document are subject of Intellectual Property (IP) rights, especially patent rights. NXg shall not be held responsible for identifying any or all such IP rights. Details of any patent rights identified during the development of this document are in the Introduction.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

There is no warranty for this document, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder and/or other parties provide this document "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the correctness and completeness of the specification is with you. Should this document prove failures, you assume the cost of all necessary servicing, repair or correction.

NXg is committed to use inclusive language in its specifications and technical reports.
The SIG CANopen-Lift Connector has prepared this document.

Contents

1. Scope **4**

1.1. Cyber security 4

2. Normative references **4**

3. Definitions, acronyms and abbreviations **4**

3.1. General 4

3.2. Acronyms 4

3.3. Abbreviations 4

3.4. Terms 4

4. General architecture **5**

5. Connection **5**

5.1. Switching from HTTPS to Websocket 5

5.2. Establish connection on JSON level 5

5.2.1. Sample “CONNECT” request 6

5.2.2. Sample “CONNECT” response 6

5.3. Heartbeat 7

5.3.1. Sample “get_heartbeat” request 7

5.3.2. Sample “get_heartbeat” response 7

5.4. Storing a unique token for enhanced security 7

5.4.1. Sample “update_unique_token” request 7

5.4.2. Sample “update_unique_token” response 8

5.4.3. Sample “update_unique_token” error 8

6. Accessing the CANopen object dictionary **8**

6.1. Get CANopen object from the dictionary 8

6.1.1. Sample “get_canopen_object” request 8

6.1.2. Sample “get_canopen_object” response (stripped) 10

6.1.3. Sample “get_canopen_object” response (naked) 11

6.1.4. Sample “get_canopen_object” response (list) 11

6.2. Get CANopen object from the dictionary (array) 12

6.2.1. Sample “get_canopen_object” request (compact array) 12

6.2.2. Sample “get_canopen_object” response (compact array) 12

6.3. Put an object into the CANopen dictionary 13

6.3.1. Sample “put_canopen_object” request 13

6.3.2. Sample “put_canopen_object” response 14

6.3.3. Sample “put_canopen_object” error 14

6.4. Elevating user permissions 14

6.4.1. Sample “elevate_privilege” request 15

6.4.2. Sample “elevate_privilege” response 15

6.4.3. Sample “elevate_privilege” error 15

6.5. Accessing other nodes in the bus 15

6.5.1. Sample “get_canopen_object” request 15

6.6. Putting values to other nodes in the bus 16

6.6.1. Sample “put_canopen_object” request 16
7. Subscribing to CANopen object changes (push) 17
7.1. Register subscription for one or more CANopen objects 17
7.1.1. Sample “reg_canopen_object” request 17
7.1.2. Sample “reg_canopen_object” response 17
7.1.3. Sample “reg_canopen_object” error 17
7.2. Unregistering from one or more CANopen objects 18
7.2.1. Sample “unreg_canopen_object” request 18
7.2.2. Sample “unreg_canopen_object” result 18
7.3. Receiving the CANopen object changes 18
7.3.1. Sample “canopen_object” notification 18
8. Appendix 19
8.1. Object codes 19
8.2. Physical units 19
8.3. Data type 20
8.4. Privilege type 21
8.5. Access Enumeration 21
8.6. Error codes 21

1. Scope

This document defines a cloud connector protocol for accessing CANopen-Lift objects as specified in CiA-417.

1.1. Cyber security

This document does not guarantee compliance with the **European Cyber Resilience Act (CRA)** or any other regulatory requirements. Users are responsible for assessing their own cybersecurity obligations and ensuring compliance with the CRA and relevant legal frameworks.

2. Normative references

CiA DS 301: CANopen - Application Layer and Communication Profile, Version 4.0
CiA DS 306: CANopen - CANopen electronic data sheet specification, Version 1.3.7
CiA417-1/ CiA 417-1 version 2, CANopen application profile for lift control systems, Part 4: Detailed application object specification

3. Definitions, acronyms and abbreviations

3.1. General

For the purpose of this document, the following definitions, acronyms and abbreviations and those given in /CiA301/ and /CiA417/apply.

3.2. Acronyms

CiA	CAN in Automation e.V.
EG	Elevator Gateway
HMI	Human-Machine Interface
TLS	Transport Layer Security
JSON	JavaScript Object Notation
REST	Representational State Transfer
CRA	European Cyber Resilience Act

3.3. Abbreviations

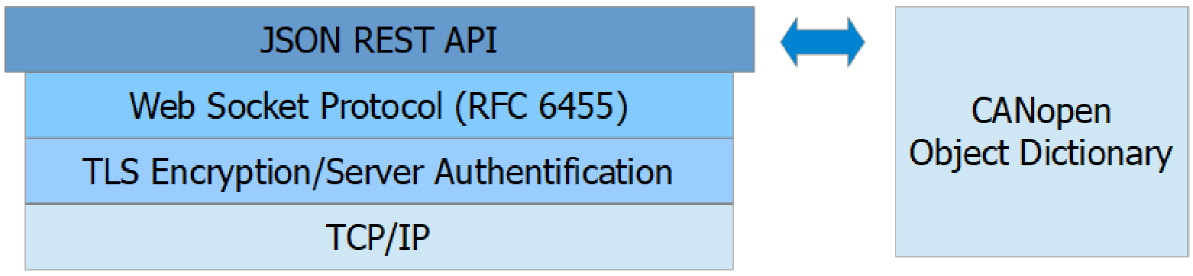
const	constant
ro	read-only
wo	write-only
rw	read and write

3.4. Terms

Elevators Gateway (EG) - Communication end point in the elevator. Provides the elevator data in the defined protocol. Can optionally include a modem, firewall, user management, HMI or be a software module in the lift controller.

4. General architecture

The cloud connector protocol for CANopen CiA-417 elevators is based on the current safety standards and technologies. It basically uses a TLS encrypted communication channel, featuring certificate based server authentication, running web-sockets that transport a JSON REST API based communication layer. The JSON REST API is used to access the CANopen object dictionary as the main information source.


JSON REST API
Web Socket Protocol (RFC 6455)
TLS Encryption/Server Authentification
TCP/IP
CANopen Object Dictionary

5. Connection

This connector *only supports encrypted* (TLS) secured connections. Public and private Certificate Authorities (CA) are supported.

5.1. Switching from HTTPS to Websocket

After the connection has been established on encrypted TLS-socket level and the servers have been authenticated, the elevator gateway will request to switch over from HTTPS to Web-socket based communication using the standardized request for this purpose, as described by the RFC 6455 specification.

```
GET /ws HTTP/1.1
Host: portal.masora.cloud
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: RDJCODhDNDY=
Sec-WebSocket-Protocol: cloud, lift
Sec-WebSocket-Version: 13
Origin: https://192.168.178.101
```

5.2. Establish connection on JSON level

After switching to Web-Socket protocol level, the lift will try to connect to the cloud service. For that purpose, the unit has to identify itself by transmitting the serial number as well as the vendor-id, product code and the domain token.

- The vendor-id is provided by the CiA for every CANopen member.
- The product code is managed by every vendor itself and shall be unique for his product portfolio, that will be connected to the cloud service.

- The serial number is a string that will vary widely across different products. It just has to be unique inside the group of lift controllers from the very same vendor.
- The domain token makes it easier to integrate new units into the cloud by giving the cloud information to which account the unit belongs.

5.2.1. Sample "CONNECT" request

```
{
  "jsonrpc": "2.0",
  "method": "CONNECT",
  "id": "<ID>",
  "params": {
    "identity": {
      "sernum": "180917010200000102",
      "vendorid": 1050,
      "productcode": 4096,
      "liftnum": "40/7289/20399-126",
      "hardware": "Thor @ NX-T2",
      "software": "01.23.10",
      "api": 1,
      "boardhash":
"2354A687AF6DC79B203E754E8E426A87FE188A62983B7AC48F9D5E195B728112",
      "cloudtoken": "2A98765DE230176C"
    },
    "domaintoken": "MAS"
  }
}
```

- The property 'cloudtoken' is optional and is only present, if the cloud actually has written a cloud defined token before. If not, the property will not be included in the JSON.
- The elevator gateway indicates via the 'api' field the highest API version supported. The cloud in response will tell the elevator gateway which API version finally is used in the communication with an API level not higher than what the elevator gateway had proposed. So if the elevator gateway would support api3 and the cloud only api2, the lift would indicate the highest possible api=3 and the cloud would decide to go for api=2 as being the highest common api both sides support. If no api version is indicated an api version of 1 will be assumed by both sides.
- The server may refuse the connection by simply closing the socket without notification at any time.

5.2.2. Sample "CONNECT" response

```
{
  "jsonrpc": "2.0",
  "id": "<ID>",
  "result": "CONNACK",
  "api": 1
}
```

5.3. Heartbeat

In order to provide a simple heartbeat on a higher protocol level, the server may use this message as a request. The heartbeat included in the websocket protocol is not enough, as a proxy might not relay the heartbeat or not respect the heartbeat interval at all.

It should be enough to send the heartbeat every five minutes to indicate that the connection is still alive, in order to detect dead sockets or a server breakdown.

5.3.1. Sample "get_heartbeat" request

```
{
  "jsonrpc": "2.0",
  "id": "<ID>",
  "method": "get_heartbeat"
}
```

5.3.2. Sample "get_heartbeat" response

```
{
  "jsonrpc": "2.0",
  "id": "<ID>",
  "result": {
    "done": 0
  }
}
```

5.4. Storing a unique token for enhanced security

On each connection request the elevator gateway submits the token set by the cloud. This ensures that the cloud service can verify the identity of the elevator gateway before granting access, therefore eliminating the risk of identity theft.

The token string can be 128 characters long. To store the string, the cloud has to provide the correct serial number of the unit as well.

5.4.1. Sample "update_unique_token" request

```
{
  "jsonrpc": "2.0",
  "method": "update_unique_token",
  "id": "<ID>",
  "params": {
    "sernum": "1809170102000000102",
    "token": "A token string that is maximum 128 characters long."
  }
}
```

5.4.2. Sample “update_unique_token” response

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "result": {
      "done": 0
   }
}
```

5.4.3. Sample “update_unique_token” error

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "error": {
      "code": <CANOPEN ABORT CODE>,
      "message": "The serial number does not match.",
      "data": "Additional data, may be omitted"
   }
}
```

6. Accessing the CANopen object dictionary

6.1. Get CANopen object from the dictionary

The method is used to fetch an object from the CANopen object dictionary. The caller gives not only the object’s multiplexer but also his preferred and alternative language used for object name and help text, that will be part of the response as extra metadata.

6.1.1. Sample “get_canopen_object” request

This sample is requesting the multiplexer for the “Trip counter” as defined in “CiA 417 part 4”.
Optionally attributes “compact”, “stripped” and “naked” define how much information is returned by the elevator gateway. When not used, those attributes are 'false' by default. See the sample results for detailed examples.

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "method": "get_canopen_object",
   "params": {
      "lang": {
         "preferred_language": "en",
         "alternate_language": "de"
      },
      "object": {
         "mplex": "0x652001",
         "compact": false,
            "stripped": false,
            "naked": false
        }
    }
}

{
    "jsonrpc": "2.0",
    "id": "<ID>",
    "result": {
        "lang": {
            "language": "en"
        },
        "object": {
            "mplex": "0x652001",
            "objcode": 8,
            "datatype": 7,
            "formtype": 10,
            "physunit": 26,
            "access": "ro",
            "privilege": 0,
            "name": "Trip counter [trips]",
            "hint": "This object represents the internal trip counter of the lift controller.",
            "value": [
                "2510"
            ],
            "min": 0,
            "max": 4294967295,
            "zero": false,
            "list": []
        },
        "hints": {
            "show_off_at_zero": false,
            "show_unlimited_at_max": false,
            "show_undefined_at_max": false
        }
    }
}
```

- The “objcode” corresponds to the “CiA 301” defined object codes. See the appendix for more details.
- The “datatype” corresponds to the “CiA 301” defined data types. See the appendix for more details.
- The “formtype” describes how the value shall be presented to the user. See the appendix for more details.
- The “physunit” is defined in “CiA 303” but extended by vendor specific values. See the appendix for more details.

- The "access" entry signals if the object is only readable or read- and writable. See the appendix for more details.
- The "privilege" defines the permissions needed for writing to the object. See the appendix for more details.
- The "zero" indicator signals if zero is a valid value, even if the min/max values indicate a different range. This is often used, if a parameter can be set, for example from 10...99 seconds or to zero to indicate 'off'. In that case usually the 'show_off_at_zero' is indicated as well.
- Hint "show_off_at_zero" indicates that a value of zero shall be presented as 'off' to the user. An example would be the lift's 'Load time' parameter or the 'Parking Time' object value.
- Hint "show_unlimited_at_max" indicates that a value matching the given 'max' value shall be presented as 'unlimited' to the user. An example would be the lift's 'Load Time' parameter.
- Hint "show_undefined_at_max" indicates that a value matching the given 'max' value shall be presented as 'undefined' to the user.
- The "min" and "max" indicators limit the range of the object value, if being a numerical value. If being a 'string' value the 'max' indicator defines the maximum count of characters the string object is capable of holding – not including any terminating zero character.

6.1.2. Sample "get_canopen_object" response (stripped)

Taking in account that the 'name' and 'hint' fields are taking up bandwidth on transmission and that the information transmitted will rarely change (only on a software update) the 'stripped' attribute provides an easy to use method to omit the text based fields corresponding with the CANopen object that had been requested.

```
{
  "jsonrpc": "2.0",
  "id": "<ID>",
  "result": {
    "object": {
      "mplex": "0x652001",
      "objcode": 8,
      "datatype": 7,
      "formtype": 10,
      "physunit": 26,
      "access": "ro",
      "privilege": 0,
      "value": [
        "2510"
      ],
      "min": 0,
      "max": 4294967295,
      "zero": false
    }
```

```
   }
}
```

6.1.3. Sample “get_canopen_object” response (naked)

Using the ‘naked’ attribute will return only the multiplexer and the actual value of an object, reducing the required bandwidth to a minimum.

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "result": {
      "object": {
         "mplex": "0x652001",
         "value": [
            "2510"
         ]
      }
   }
}
```

6.1.4. Sample “get_canopen_object” response (list)

If the requested object is a list represented by a numerical value, the ‘list’ property will contain an array with text values that correspond to each allowed numerical value.

```
{
   "jsonrpc": "2.0",
   "id": "1234",
   "result": {
      "lang": {
         "language": "EN"
      },
      "object": {
         "mplex": "0x411200",
         "objcode": 7,
         "datatype": 5,
         "formtype": 15,
         "physunit": 0,
         "access": "rw",
         "privilege": 1,
         "name": "Hall lantern options",
         "hint": "This object defines the moment to turn on the hall lantern..",
         "value": [
            "0"
         ],
         "min": 0,
         "max": 3,
         "zero": false,
         "list": [
            "Turn the hall lantern on, when approaching to the floor",
            "Turn the hall lantern on, after arrival when unlocking the doors.",
            "Turn the hall lantern on, after the car door has been fully opened.",
            "off"
          ]
        },
        "hints": {
          "show_off_at_zero": false,
          "show_unlimited_at_max": false,
          "show_undefined_at_max": false
        }
    }
}
```

6.2. Get CANopen object from the dictionary (array)

Some objects, like the array holding the floor level positions, might be fetched with one single call by using the 'compact' array option. Make sure that you provide a sub-index one in your multiplexer when having set 'compact=true'. In the given example the multiplexer is 0x4010 sub 1 = 0x401001.

6.2.1. Sample "get_canopen_object" request (compact array)

```
{
  "jsonrpc": "2.0",
  "id": "<ID>",
  "method": "get_canopen_object",
  "params": {
    "lang": {
      "preferred_language": "en",
      "alternate_language": "de"
    },
    "object": {
      "mplex": "0x401001",
      "compact": true
    }
  }
}
```

6.2.2. Sample "get_canopen_object" response (compact array)

```
{
  "jsonrpc": "2.0",
  "id": "<ID>",
  "result": {
    "lang": {
      "language": "en"
    },
    "object": {
      "mplex": "0x401001",
      "objcode": 8,
      "datatype": 4,
      "formtype": 26,
      "physunit": 15,
      "access": "ro",
      "privilege": 1,
      "name": "Floor level positions",
      "hint": "This object holds the floor levels in millimeter.",
      "value": [
        "1000",
        "4500",
        "8000",
        "11500",
        "15000",
        "18500",
        "22000",
        "25500"
      ],
      "min": 0,
      "max": 2147483647,
      "zero": false,
      "list": []
    },
    "hints": {
      "show_off_at_zero": false,
      "show_unlimited_at_max": false,
      "show_undefined_at_max": false
    }
  }
}
```

6.3. Put an object into the CANopen dictionary

The method 'put_canopen_object' is used to update an object value in the object dictionary.

The elevator gateway might heavily limit write access to objects within the object dictionary for Cyber-Security reasons.

The caller provides the new value as well as the physical unit and the data type. This extra meta information is used by the lift to double check that both parties (cloud and lift) are using the very 'same model' of the object in question.

Some of the objects require elevated user rights (privileges) to actually complete a write operation. Other objects can only be written locally by an engineer on site.

6.3.1. Sample "put_canopen_object" request

The given example writes the value '4' to the manufacturer specific object 0x5022 sub 0, which is used to trigger a passenger call at floor 4.

```
{
  "jsonrpc": "2.0",
  "method": "put_canopen_object",
    "id": "<ID>",
    "params": {
        "sernum": "180917010200000102",
        "object": {
            "mplex": "0x502200",
            "datatype": 7,
            "physunit": 0,
            "value": "4"
        }
    }
}
```

6.3.2. Sample "put_canopen_object" response

```
{
    "jsonrpc": "2.0",
    "id": "<ID>",
    "result": {
        "object": {
            "mplex": "0x502200"
        }
    }
}
```

6.3.3. Sample "put_canopen_object" error

An error might occur, if the object is not writable at all or requires elevated user rights to be written.
The CANopen abort codes can be found in the "CiA-301" paper. The most common ones can be found in this document's appendix.

```
{
    "jsonrpc": "2.0",
    "id": "<ID>",
    "error": {
        "code": "<CANOPEN ABORT CODE>",
        "message": "<Error text in the last requested language.>",
        "data": "0x502200"
    }
}
```

6.4. Elevating user permissions

In order to use the method "put_canopen_object" on objects that require a higher user-right (privilege) level, the cloud may elevate its rights temporarily by using the correct SHA-1 hash corresponding to one of passwords matching the elevator gateway security settings. There are some objects that require an 'Engineer on Site' privilege, which can not be granted remotely.

6.4.1. Sample “elevate_privilege” request

```
{
   "jsonrpc": "2.0",
   "method": "elevate_privilege",
   "id": "<ID>",
   "params": {
      "sernum": "180917010200000102",
      "privilege": 1,
      "passhash": "<The unsalted SHA1 password hash for the desired priv.>"
   }
}
```

6.4.2. Sample “elevate_privilege” response

```
{
         "jsonrpc": "2.0",
         "id": "<ID>",

         "result": {
         "duration": 300
         }
}
```

- “duration” signals how long the permissions will be granted by the elevator gateway in seconds.

6.4.3. Sample “elevate_privilege” error

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "error": {
      "code": "<CANOPEN ABORT CODE>",
      "message": "The password hash does not match.",
      "data": "1"
   }
}
```

6.5. Accessing other nodes in the bus

In order to access the object dictionary from other nodes than the lift controller, the JSON for the “get_canopen_object” method shall be extended by the ‘node’ and the ‘net’ properties. Furthermore a ‘datatype’ property shall be given to allow the read data to be interpreted in the right way.

6.5.1. Sample “get_canopen_object” request

In this given example we fetch the parameter of the first virtual input of a node 12 (LXC I/O card).

```
{
   "jsonrpc": "2.0",
    "id": "<ID>",
    "method": "get_canopen_object",
    "params": {
        "lang": {
            "preferred_language": "en",
            "alternate_language": "de"
        },
        "node": {
            "id": 12,
            "net": 1,
            "datatype": 25
        },
        "object": {
            "mplex": "0x610001",
            "compact": false
        }
    }
}
```

6.6. Putting values to other nodes in the bus

In order to access the object dictionary from other nodes than the elevator gateway, the JSON for the "put_canopen_object" method shall be extended by the 'node' and the 'net' properties.

- The serial number is still the serial number of the elevator gateway to ensure that you are sending this request to the correct site.
- Accessing other nodes than the elevator gateway, via the underlying field bus system, always requires 'Setup Privileges' for a 'put' operation.

6.6.1. Sample "put_canopen_object" request

```
{
    "jsonrpc": "2.0",
    "method": "put_canopen_object",
    "id": "<ID>",
    "params": {
        "sernum": "180917010200000102",
        "node": {
            "id": 12,
            "net": 1
        },
        "object": {
            "mplex": "0x502200",
            "datatype": 7,
            "physunit": 0,
            "value": "4"
        }
    }
}
```

7. Subscribing to CANopen object changes (push)

7.1. Register subscription for one or more CANopen objects

In order to be informed, when an object value has been changed, the cloud server can subscribe to one or more objects, using the multiplexer values, that have been standardized in the lift industry. Those objects will be automatically pushed on demand to the cloud then.

- In the case of a disconnect, all subscriptions will be automatically canceled.

7.1.1. Sample “reg_canopen_object” request

In this example the cloud registers for the CANopen multiplexer 0x6520 sub 0x01, which is regarding to the CANopen 417 lift industry standard the ‘Trip counter’ and the multiplexer 0x6520 sub 0x02 which is the ‘Operating hours’. The inhibit time, given in [ms] indicates how quick the very next update shall be pushed, if the value has changed again, in order to prevent ‘spamming’ the cloud. The threshold value indicates how far the current value shall be different in absolute terms from the last value transmitted, before the unit would push an update of the value again.

```
{
   "jsonrpc": "2.0",
   "method": "reg_canopen_object",
   "id": "<ID>",
   "params": {
      "mplex": [
         "0x652001",
         "0x652002"
      ],
      "inhibit": 1000,
      "threshold": 1
   }
}
```

7.1.2. Sample “reg_canopen_object” response

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "result": {
      "done": 0
   }
}
```

7.1.3. Sample “reg_canopen_object” error

```
{
   "jsonrpc": "2.0",
   "id": "<ID>",
   "error": {
        "code": 100794368,
        "message": "Object does not exists.",
        "data": "0x652099"
    }
}
```

7.2. Unregistering from one or more CANopen objects

7.2.1. Sample “unreg_canopen_object” request

```
{
    "jsonrpc": "2.0",
    "method": "unreg_canopen_object",
    "id": "12345",
    "params": {
        "mplex": [
            "0x652001",
            "0x652002"
        ]
    }
}
```

7.2.2. Sample “unreg_canopen_object” result

```
{
    "jsonrpc": "2.0",
    "id": "<ID>",
    "result": {
        "done": 0
    }
}
```

7.3. Receiving the CANopen object changes

When the value of one of the objects changes, the unit shall send the designated value (push) automatically, taking the given inhibit time and threshold into account. The transmitted JSON from the unit would be a ‘stripped down’ JSON just containing the value. This JSON does not require a response from the cloud server as it is a notification, a request object without an “id” member according to “JSON-RPC 2.0 4.1 Notification”.

7.3.1. Sample “canopen_object” notification

```
{
    "jsonrpc": "2.0",
    "method": "canopen_object",
    "result": {
        "mplex": "0x652001",
        "value": "2510"
    }
}
```

8. Appendix

8.1. Object codes

Dictionary structure and entries as defined in "CiA 301, chapter 10.2 table 'Object Dictionary Object Definitions'.

- 0 - NULL
- 2 - DOMAIN
- 5 - DEFTYPE
- 6 - DEFSTRUCT
- 7 - VAR
- 8 - ARRAY
- 9 - RECORD

8.2. Physical units

// CiA-303 base SI units
- 0 // none
- 1 // [m]
- 2 // [kg]
- 3 // [s]
- 4 // [A]
- 5 // [K]
- 6 // [mol]
- 7 // [cd]

// Custom units
- 8 // [%]
- 9 // [‰]
- 10 // [V]
- 11 // [ms]
- 12 // [min]
- 13 // [h]
- 14 // [d]
- 15 // [mm]

// CiA-303 supplementary SI units
- 16 // [rad]
- 17 // [sr]

// Custom units
- 18 // [mm/s]
- 19 // [mm/s2]
- 20 // [mm/s3]
- 21 // [Bit/s]
- 22 // [kBit/s]
- 23 // [°C]
- 24 // [increments]
- 25 // [kg] or [%], depending on 0x6480:2 "Load value – SI Unit", see CiA 417
- 26 // trips/travels
- 27 // gravity
- 28 // mm/inc

- 29 // W (Watts) Power
- 30 // Wh (Watts hours) Energy
- 31 // Node-Id
- 32 // Rounds per minute (RPM)
- 33 // Hertz

// Imperial Units

- 34 // [in] Inch
- 35 // [fpm] feet per minute
- 36 // [lb] pound
- 37 // [oF] Fahrenheit
- 38 // [ft/s2]
- 39 // [ft/s3]
- 40 // [ft]

// Special

- 41 // [delta oC]
- 42 // [delta oF]
- 43 // [switching cycles]

8.3. Data type

The data type enumeration is based on the CiA 301, chapter 7.4.7, table 'Object dictionary data types'. These are the typical CANopen data types as used in the object dictionary.

- 0x01// CO_BOOLEAN
- 0x02// CO_INTEGER8
- 0x03// CO_INTEGER16
- 0x04// CO_INTEGER32
- 0x05// CO_UNSIGNED8
- 0x06// CO_UNSIGNED16
- 0x07// CO_UNSIGNED32
- 0x08// CO_REAL32
- 0x09// CO_VISIBLE_STRING
- 0x0A// CO_OCTED_STRING
- 0x0B// CO_UNICODE_STRING
- 0x0C// CO_TIME_OF_DAY
- 0x0D// CO_TIME_DIFFERENCE
- 0x0F// CO_DOMAINTYPE
- 0x10// CO_INTEGER24
- 0x11// CO_REAL64
- 0x12// CO_INTEGER40
- 0x13// CO_INTEGER48
- 0x14// CO_INTEGER56
- 0x15// CO_INTEGER64
- 0x16// CO_UNSIGNED24
- 0x18// CO_UNSIGNED40
- 0x19// CO_UNSIGNED48
- 0x1A// CO_UNSIGNED56
- 0x1B// CO_UNSIGNED64

- 0x20// CO_PDO_COMMUNICATION_PARAM [UNSIGNED32]
- 0x21// CO_PDO_MAPPING [UNSIGNED32]
- 0x22// CO_SDO_PARAM [UNSIGNED32]
- 0x23// CO_IDENTITY [UNSIGNED32]

8.4. Privilege type

- 0 – User Right Level
- 1 – Service Right Level
- 2 – Setup Right Level

8.5. Access Enumeration

- “rw”: Read and write access
- “wo”: Write only access
- “ro”: Read only access
- “const”: Read only access but value is constant

8.6. Error codes

The error codes are taken from “CANopen CiA-301” to simplify including this cloud solution into existing CANopen applications.

- 0x00000000 // “No error, everything is fine.”
- 0x05030000 // “Toggle bit not alternated.”
- 0x05040001 // “Client/server command specifier not valid or unknown.”
- 0x05040000 // “SDO protocol timed out.”
- 0x05040005 // “Out of memory.”
- 0x06010001 // “Attempt to read a write only object.”
- 0x06010002 // “Attempt to write a read only object.”
- 0x06020000 // “Object does not exist in the object dictionary.”
- 0x06010000 // “Unsupported access to an object.”
- 0x06040043 // “General parameter incompatibility reason.”
- 0x06070010 // “Data type does not match, length of param. does not match.”
- 0x06070013 // “Data type does not match, length of service parameter too low.”
- 0x06070012 // “Data type does not match, length of service param. too high.”
- 0x06090011 // “Sub-index does not exist.”
- 0x06090036 // “Maximum value is less than minimum value.”
- 0x06090030 // “Invalid value for parameter.”
- 0x06090032 // “Value of parameter written too low (download only).”
- 0x06090031 // “Value of parameter written too high (download only).”
- 0x08000020 // “Data cannot be transferred or stored to the application.”
- 0x08000021 // “Data cannot be transferred or stored because of locacontrol.”
- 0x08000024 // “No data available”
- 0x08000000 // “General error”