# OPC UA Subscriber provider

Version 1.1.0

# User's Guide

December 16, 2024

[Remark]

Some of the images used in this document are taken from Reference 1 and Reference 2.

Appendix A of this document is taken from Reference 1.

Some parts of Appendix B of this document are taken from Reference 2.

## [Revision History]

| Version | Date | Content |
|---|---|---|
| 1.0.0 | 2023-11-13 | First edition. |
| 1.0.0 | 2024-12-16 | Update of internally used library.<br>Added PublishingInterval to AddVariable option with library updates. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Table of Contents

# 1. Introduction

This document is the user's guide for OPC UA Subscriber providers.

OPC UA Subscriber providers can receive OPC UA UDP messages (hereinafter referred to as messages) issued by OPC UA Publisher using the brokerless model.

Chapter2 provides an introduction to OPC UA Subscriber providers and a detailed description of the variables.

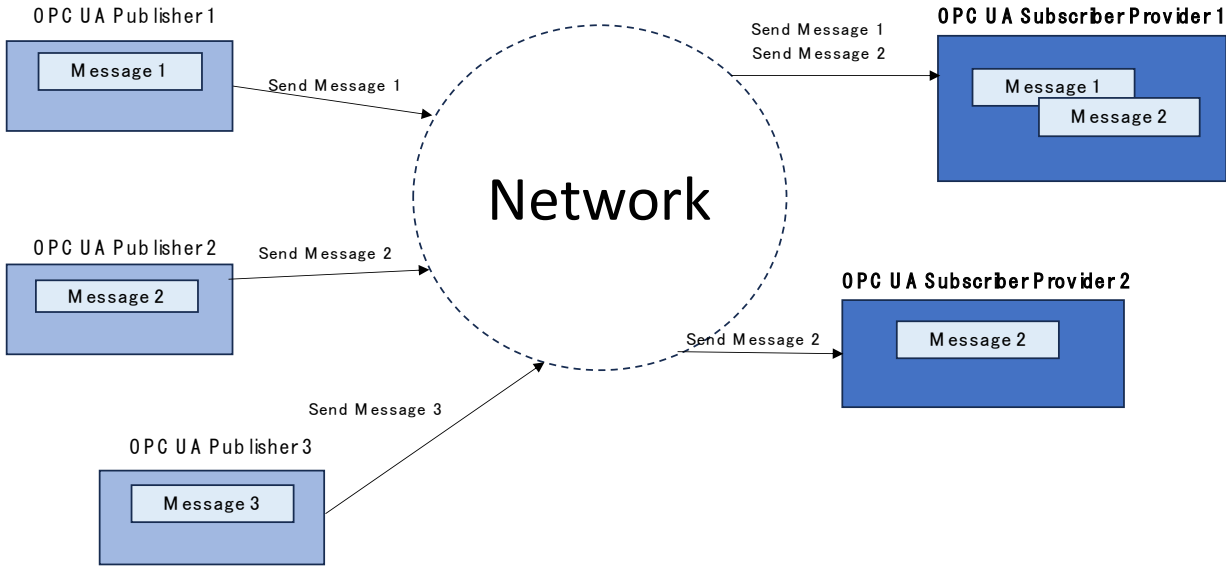


**Figure 1-1 OPC UA Subscriber Provider overview**

# 2. Provider Overview

## 2.1. Overview

OPC UA Subscriber providers are CAO providers that provide the ability to receive specified messages and read received messages. Its filetype is DLL(Dynamic Link Library) and is dynamically loaded at use from CAO.

**Table 2-1 OPC UA Subscriber Provider**

| | |
|---|---|
| File name | CaoProvOPCUASubscriber.dll |
| ProgID | CaoProv.OPCUA.Subscriber |
| Registry registration | Regsvr32 CaoProvOPCUASubscriber.dll |
| Deleting registry entries | Regsvr32 /u CaoProvOPCUASubscriber.dll |

### 2.1.1. OPC UA

OPC Unified Architecture(UA) is a platform-independent, service-oriented architecture that integrates all the functionality of individual OPC Classic specs into an extensible framework.

### 2.1.2. OPC UA UDP

A UDP based protocol used to transport NetworkMessage, one of the Message-Oriented Middleware (Message Oriented Middleware). UDP Transport Protocol URL is written in the following format.

Format: opc.udp://<host>[:<port>]

- If Publisher uses a multicast address in URL <host>, the <host> in URL of Publish provider specifies that multicast address.
- If Publisher uses URL <host> with Subscriber provider's IP address or hostname, then Publisher's IP address or hostname is specified by the <host> in Subscriber provider's URL.
- For OPC UA Subscriber providers, specify the <host> of URL in IPv4 when specifying IP addressing.

For more information, see Reference 1. For a summary of.OPC UA PubSub communication, see Appendix B.

### 2.1.3. Capabilities of OPC UA Subscriber provider

OPC UA Subscriber providers provide the following features:

- Receiving messages
  - When a specified message is received, an event notification (CaoController::OnMessage) is sent.
    - How to specify the reception target
      - ✧ It can be specified as a PublisherId, DataSetWriterId pair or as a PublisherId, WriterGroupId, DataSetWriterId pair.
      - ✧ One or more sets of receive targets can be specified for a controller object.
  - A message buffer is generated and stored for each receive target of the received message.
  - An unsupported event is notified (CaoController::OnMessage) when a message using an unsupported type is received.

- Read received message

When the message is read (CaoVariable::get_Value), the data is read from the corresponding message buffer.

- The read data does not include IP headers, UDP headers, various headers of UADP messages, options, etc.
- If there are no received messages, VT_EMPTY is returned.
- The maximum message length is 65535 bytes.
- The message length includes the lengths of IP headers, UDP headers, various headers of UADP messages, and options.

- Notes

OPC UA Subscriber providers need meta-information (DataSetMetaData) with type information to correctly retrieve data from received messages.

- If DataSetMetaData used by OPC UA Publisher does not contain version information, Subscriber providers may not be able to retrieve the data correctly from the message.
- Subscriber providers retrieve DataSetMetaData from OPC UA Publisher each time they are designated for receipt.
- When Subscriber provider detects a DataSetMetaData refresh when it receives a message, it reacquires DataSetMetaData from OPC UA Publisher.

The OPC UA Subscriber provider can read up to 10 NetworkMessages per second from a single UADPUrl. If you want to read more NetworkMessages, change the UADPUrl to distribute them, or modify the OPC UA Publisher to send multiple DataSets together in a single NetworkMessage.

## 2.2. Method Properties

### 2.2.1. CaoWorkspace::AddController method

Subscriber providers refer to the connectivity parameters for communication during AddController and create a PubSubConnection.

Format AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

BstrCtrlName : [in] Controller name

BstrProvName : [in] Provider name. Fixed value="CaoProv.OPCUA.Subscriber"

BstrPcName : [in] Provider's running machine name

BstrOption : [in] Option string

The following lists the option strings.

**Table 2-2 Optional String for CaoWorkspace::AddController**

| Option name[1] | Meaning[2] |
|---|---|
| UADPUrl=<Publisher's IP address> | Required specification item.<br>Specify the connection parameters.<br>(Ref: 2.2.1.1) |
| Request destination of DiscoveryUrl<br>[=<DataSetMetaData >] | Optional field.<br>Specifies Request destination for DataSetMetaData.<br>(Default: opc.udp://224.0.2.14:4840)[3]<br>(Ref: 2.2.1.2) |
| NetworkInterface<br>[=<IP addressof NIC to be received>] | Optional field.<br>Specifies the network interface to be received.<br>(Default: 0.0.0.0)<br>(Ref: 2.2.1.3) |

#### 2.2.1.1. UADPUrl Optional

UADPUrl optional parameter string is shown below.

"UADPUrl=opc.udp://<host>[:<port>]"[4]

[1] Square brackets ("[]") indicate optional items.
[2] Optional items can omit the option name itself.
[3] 224.0.2.14 is IPv4 multicast address for discovering OPC UA registered in IANA.
[4] The "opc.udp" part connects "opc" and "udp" with a period.

Example :"UADPUrl=opc.udp://239.0.0.1:4840"

Example :"UADPUrl=opc.udp://239.0.0.1"

| | | |
|---|---|---|
| <host> | : | Host name or IP address to connect to.<br>If URL of Publisher specifies a multicast address, specify the multicast address in <host>.[5]<br>Example :"239.0.0.1"<br>If URL of Publisher specifies a unicast address, specify IP address or hostname in <host>.[6] |
| <port> | : | Specify the same connection-port number as Publisher. (default: 4840)<br>Example :"4840" |

#### 2.2.1.2. DiscoveryUrl Optional

DiscoveryUrl optional parameter string is shown below.

"DiscoveryUrl[=<opc.udp://<host>[:<port>]]"

| | | |
|---|---|---|
| <host> | : | IP addressing for sending and receiving Request of DataSetMetaData.<br>Specify the same multicast address as Publisher for <host>.[7]<br>(default: 224.0.2.14) |
| <port> | : | Specify the same connection-port number as Publisher.<br>(default: 4840) |

#### 2.2.1.3. NetworkInterface Optional

NetworkInterface optional parameter string is shown below.

"NetworkInterface[=<IP addressof NIC to be received>]"

| | | |
|---|---|---|
| <IP of NIC to be received> | : | IP of NIC to receive.<br>0.0.0.0 works for all NIC. (Default: 0.0.0.0) |

[5] Multicast addresses range from 224.0.0.0 to 239.255.255.255.
[6] A single PC cannot be unicast.
[7] DiscoveryUrl cannot be unicast.

### 2.2.2. CaoController::AddVariable method

AddVariable method of CaoController is a method that creates a variable object for Subscribe with each provider.

Subscriber providers repeatedly Request DataSetMetaData after creating a variable. After receiving a response, Subscribe is executed. Run UnSubscribe on Remove Variable.[8]

Format AddVariable(<bstrVariableName:VT_BSTR>[,<bstrOption:VT_BSTR>])

<bstrVariableName> : [in] Variable Name

<bstrOption> : [in] Option string

The following lists the option strings.

**Table 2-3 Optional String for CaoController::AddVariable**

| Option name[1] | Meaning[2] |
|---|---|
| PublisherIdDataType = <data type> | Required specifications.<br>Specifies the data type of PublisherId.<br>(Ref: 2.2.2.1) |
| PublisherId=<ID number/character string> | Required specifications.<br>Specify PublisherId.<br>(Ref: 2.2.2.1) |
| WriterGroupId[=<ID number >] | Optional information.<br>Specify WriterGroupId.<br>(default: 0)<br>(Ref: 2.2.2.1) |
| DataSetWriterId=<ID number > | Required specifications.<br>Specify DataSetWriterId.<br>0 cannot be entered.<br>(Ref 2.2.2.1) |
| RequestMetaDataInterval<br>[=<RequestMetaData sending interval >] | Optional information.<br>Specifies how often, in milliseconds, DataSetMetaData sends Request. |

[8] Since the connection is temporarily OFF during deletion, the message cannot be received by all Variable created from the same Controller until the deletion is completed.

| | |
|---|---|
| | (default: 1000)<br>(Ref: 2.2.2.2) |
| Max MaxNetworkMessageSize<br>[=<NetworkMessage size>] | Optional information.<br>Specifies the largest NetworkMessage to be received in bytes.<br>(default: 1500)<br>(Ref: 2.2.2.3) |
| MaxMessageCount[=<max message buffers>] | Optional information.<br>Specifies the maximum number of data items in the message buffer.<br>(default: 5)<br>(Ref: 2.2.2.4) |
| WaitReaderGroup[=<TRUE/FALSE>] | Optional information.<br>Specify ReaderGroup creation timing.<br>(Default: FALSE)<br>(Ref: 2.2.2.5) |
| StartSubscribe[=<TRUE/FALSE>] | Optional information.<br>Set whether Subscribe is started or not.<br>(Default: TRUE)<br>(Ref: 2.2.2.6) |
| PublishingInterval[=<Interval for receiving data >] | Optional information.<br>Specify the value of PublishingInterval set in the Publisher that is the source of the transmission.<br>(Default: 1000)<br>(Ref: 2.2.2.7) |

#### 2.2.2.1. Selection of receiving object by ID

Subscriber providers specify what to filter and receive on configured ID.

If the type and value do not correspond to PublisherId on Publisher, data cannot be received.

- PublisherIdDataType

Specifies the type of PublisherId by number.

**Table 2-4 Type numbers and their corresponding data types**

| No. | Data type | Input range |
|---|---|---|
| 1 | Byte | 0～255 |
| 2 | UInt16 | 0～65535 |
| 3 | UInt32 | 0～4294967295 |
| 4 | UInt64 | 0～18446744073709551615 |
| 5 | String | Unlimited[9] |

- PublisherId

  Specifies a range-value of the type specified in PublisherIdDataType option.

- WriterGroupId

  Specify a non-zero UInt16 range.

  If 0 is specified, all WriterGroupId are accepted.

- DataSetWriterId

  Specify a non-zero UInt16 range.

#### 2.2.2.2. RequestMetaDataInterval Optional

Repeats Request at the set interval (milliseconds) until DataSetMetaData can be received from the time of variable-creation.

RequestMetaDataInterval[=<RequestMetaData sending interval >]

| | | | |
|---|---|---|---|
| <RequestMetaData interval> | sending | : | 1000～3600000 |

#### 2.2.2.3. MaxNetworkMessageSize Optional

Specifies the maximum size (in bytes) of data to be received.

If data larger than the set value is sent, it will not be received.[10]

MaxNetworkMessageSize[=<max. size of received data>]

| | | | |
|---|---|---|---|
| <Maximum size of data to be received> | | : | 1500～65535[11] |

[9] Limited to the maximum value of the return type size_t of Std::string.size(). The maximum value for this provider is 4294967295.
[10] No error notification is performed.
[11] This is actually a value including a header etc.

#### 2.2.2.4. MaxMessageCount Optional

Subscriber providers generate a message buffer for each receive target specified at CaoController::AddVariable and store the received data.

MaxMessageCount option specifies the largest amount of data in the message buffer to be generated.

If data exceeds the maximum number of data, the received data is discarded and an error message is notified. (Ref: 2.2.5.2)

MaxMessageCount[=<max. pre-read data>]
<Maximum number of data before reading> : 1～2147483647

#### 2.2.2.5. WaitReaderGroup Optional

Specifies the timing at which ReaderGroup is created.

If TRUE is specified, creation of ReaderGroup is placed in standby status.

Messages cannot be received (no Subscribe) during standby.

To create a standby ReaderGroup so that messages can be received, execute CreateReaderGroup command.

If you specify FALSE, ReaderGroup is created immediately.

However, when ReaderGroup is created, the connection is temporarily OFF, so messages cannot be received on all Variable created from the same Controller until the creation is completed.

#### 2.2.2.6. StartSubscribe Optional

Specifies whether Subscribe starts.

If TRUE is specified, Subscribe starts.

If FALSE is specified, Subscribe is not started.

It can also be specified by executing StartSubscribe command and StopSubscribe command, respectively.

#### 2.2.2.7. PublishingInterval Optional

Specify the sending interval set in Publisher and receive at the specified interval.

If the sending interval of Publisher does not match, the reception timing may be misaligned.

### 2.2.3. CaoController::Execute method

Executes the specified command generically.

Refer to 2.3.1 for the command names and details that can be used.

Format Execute(<bstrCommand:VT_BSTR>[,<vntParam:VARIANT>[,<pVal:VARIANT>]])

<bstrCmd> : [in] Command name

<vntParam> : [in] Parameters

<pVal> : [out] Acquired data

### 2.2.4. CaoVariable::get_Value property

Gets the received data from the message buffer.

If there is no received data, VT_EMPTY is returned.

The received data is stored in an array of VARIANT (VT_VARIANT | VT_ARRAY).

VT_VARIANT | VT_ARRAY

[0]PublisherIdDataType dependent : PublisherId

[1]VT_UI2 : WriterGroupId

[2]VT_UI2 : DataSetWriterId

[3]VT_DATE : Receive Timestamp

[4]Variable type dependency : Retrieved from NetworkMessage's Payload

**Restrictions**

Data types that are not listed in the table below cannot be acquired.

Arrays of two or more dimensions are not supported.

**Table 2-5 Supported data types**

| OPC UA built-in types | VARIANT Datatypes | Description |
|---|---|---|
| SByte | VT_I1 | 1-byte integer type |
| Int16 | VT_I2 | 2-byte integer type |
| Int32 | VT_I4 | 4-byte integer type |
| Int64 | VT_I8 | 8-byte integer type |
| Byte | VT_UI1 | 1-byte unsigned integer type |
| UInt16 | VT_UI2 | 2-byte unsigned integer type |
| UInt32 | VT_UI4 | 4-byte unsigned integer type |
| UInt64 | VT_UI8 | 8-byte unsigned integer type |
| Float | VT_R4 | 4-byte floating point type |
| Double | VT_R8 | 8-byte floating point type |
| DateTime | VT_DATE | Date Types |
| String | VT_BSTR | Character string type |

| | | |
|---|---|---|
| Boolean | VT_BOOL | Boolean type |
| ByteString | VT_UI1 \| VT_ARRAY | Byte string type |

### 2.2.5. CaoController::OnMessage Events

A CaoController class-of-event OnMessage occurs at the following times:

**Table 2-6 Message types**

| Message type | | Trigger |
|---|---|---|
| 0 | NetworkMessage reception | Occurs when a NetworkMessage is received. |
| 1 | Maximum number of message buffer data exceeded | If the message buffer is full at data reception, this error occurs when the received data cannot be stored. |
| 2 | Receive operating error Memory shortage | This error occurs when an error occurs due to a memory shortage during reception operation. |
| 3 | Receive operating error (other than out of memory) | Occurs when an error occurs during reception operation due to some reason other than memory shortage. |
| 4 | Receiving unsupported type-used NetworkMessage | Table 2-5 Supported data types |

#### 2.2.5.1. NetworkMessage reception

The data format obtained by NetworkMessage reception message is shown below.

Number : Message type (0)

Value : Data obtained from Payload of NetworkMessage (variable-type dependent)

DateTime : Receive Timestamp

Destination : AddVariable method <bstrVariableName>

Source : AddController Method UADPUrl Optional

Description : Empty string

#### 2.2.5.2. Maximum number of message buffer data exceeded

The data format obtained by the message buffer maximum data count over message is shown below.

Number : Message type (1)

Value : VT_EMPTY

DateTime : Receive Timestamp

Destination : AddVariable method <bstrVariableName>

Source : AddController Method UADPUrl Optional

Description : Empty string

#### 2.2.5.3. Receive operating error (insufficient memory)

The data format obtained by a receive operation error (insufficient memory) message is shown below.

Number : Message type (2)

Value : VT_EMPTY

DateTime : Time stamp

Destination : AddVariable method <bstrVariableName>

Source : AddController Method UADPUrl Optional

Description : Empty string

#### 2.2.5.4. Receive operating error (other than insufficient memory)

The following shows the data format obtained by a receive operation error (other than insufficient memory) message.

Number : Message type (3)

Value : Internal error code[12]

DateTime : Time stamp

Destination : AddVariable method <bstrVariableName>

Source : AddController Method UADPUrl Optional

Description : Error description information[13]

#### 2.2.5.5. Receiving unsupported type-used NetworkMessage

The data format obtained by NetworkMessage received messages of unsupported type is shown below. (Refer to Table 2-5 Supported Data Types)

Number : Message type (4)

Value : VT_EMPTY

DateTime : Time stamp

Destination : AddVariable method <bstrVariableName>

Source : AddController Method UADPUrl Optional

[12] Coding defined by SDK

[13] If there is no description information related to the internal error code, an empty string is stored.

Description : Empty string

## 2.3. Command list

### 2.3.1. CaoController class

**Table 2-7 CaoController::Execute Commands**

| Command | Function | Reference |
|---|---|---|
| CreateReaderGroup | Specify TRUE in 2.2.2.5 to create a ReaderGroup that is in standby status.<br>Since the connection is temporarily OFF when ReaderGroup is created, messages cannot be received in all Variable created from the same Controller until the creation is completed. | P.18 |
| StartSubscribe | Starts Subscribe at the specified Variable. | P.18 |
| StopSubscribe | Stops Subscribe at the specified Variable. | P.19 |
| GetMessageValue | Retrieves the received data from the specified Variable. | P.19 |

## CreateReaderGroup

| | |
|---|---|
| **Syntax** | Object.CreateReaderGroup() |
| **Argument** | None |
| **Return value** | None |
| **Description** | Specify TRUE in 2.2.2.5 to create a ReaderGroup that is in standby status.<br>Since the connection is temporarily OFF when ReaderGroup is created, messages cannot be received in all Variable created from the same Controller until the creation is completed. |

## StartSubscribe

| | |
|---|---|
| **Syntax** | Object.StartSubscribe(Data) |
| **Argument** | Data.VT=VT_BSTR<br>Data.bstrVal=VariableName |
| **Return value** | None |
| **Description** | Starts Subscribe at the specified Variable. |

## StopSubscribe

| | |
|---|---|
| **Syntax** | Object.StopSubscribe(Data) |
| **Argument** | Data.VT=VT_BSTR<br>Data.bstrVal=VariableName |
| **Return value** | None |
| **Description** | Stops Subscribe at the specified Variable. |

## GetMessageValue

| | |
|---|---|
| **Syntax** | pVal=object.GetMessageValue(Data) |
| **Argument** | Data.VT=VT_BSTR<br>Data.bstrVal=VariableName |
| **Return value** | pVal.VT=VT_VARIANT \| VT_ARRAY |
| **Description** | Reads one oldest received message from the receive buffers of the specified Variable.<br>If there is no received data, VT_EMPTY is returned.<br>The received data is stored in an array of VARIANT (VT_VARIANT \| VT_ARRAY). (Ref: 2.2.4) |

## 2.4. Variable List

### 2.4.1. CaoController class

**Table 2-8 CaoController Class User Variable List**

| Variable Name | Data type | Description | Attribute | |
|---|---|---|---|---|
| | | | Get | Put |
| Optional | VT_VARIANT \| VT_ARRAY | Run Subscribe when you create a variable. (Ref: 2.2.2)<br>Get_Value retrieves data from the message buffer. (Ref: 2.2.4) | ○ | - |

**Table 2-9 CaoController Class System Variable List**

| Variable Name | Data type | Description | Attribute | |
|---|---|---|---|---|
| | | | Get | Put |
| @VERSION | VT_BSTR | Provider version information. | ○ | - |

## 2.5. Error code

OPC UA Subscriber providers define the following unique error codes:

For ORiN2 common errors, see the error code section in ORiN2 Programming Guide.

**Table 2-10 Unique Error Codes**

| Error Name | Error Number | Description |
|---|---|---|
| E_FAILED_SUB_INVALID_PARAMS_PUBLISHERIDDATATYPE | 0x80100001 | The parameter (PublisherIdDataType option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_PUBLISHERID | 0x80100002 | The parameter (PublisherId option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_WRITERGROUPID | 0x80100003 | The parameter (WriterGroupId option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_DATASETWRITERID | 0x80100004 | The parameter (DataSetWriterId option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_REQUESTMETADATAINTERVAL | 0x80100005 | The parameter (RequestMetaDataInterval option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_MAXNETWORKMESSAGESIZE | 0x80100006 | The parameter (MaxNetworkMessageSize option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_MAXMESSAGECOUNT | 0x80100007 | The parameter (MaxMessageCount option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_WAITREADERGROUP | 0x80100008 | The parameter (WaitReaderGroup option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_STARTSUBSCRIBE | 0x80100009 | The parameter (StartSubscribe option) required for reception is |

| | | |
|---|---|---|
| | | invalid. |
| E_FAILED_SUB_STARTUP_SUBS | 0x8010000A | Subscriber providers encountered an error while preparing to receive. |
| E_FAILED_SUB_INVALID_PARAMS_UADPURL | 0x8010000B | The parameter (UADPUrl option) required for reception is invalid. |
| E_FAILED_SUB_INVALID_PARAMS_PUBLISHINGINTERVAL | 0x8010000C | The parameter (PublishingInterval option) required for reception is invalid. |

# 3. Appendix

## Appendix A. OPC UA PubSub Glossary

References 1 cite terms that appear below to address OPC UA PubSub.

- **DataSetClass**
  It has a.DataSetClassId(GUID that declares the content of DataSet. (See References 1, 3.1.1.)
- **DataSetMetaData**
  You can have a.DataSetClassId(GUID that explains what DataSet is and what it means. (See References 1, 3.1.2.)
- **DataSetReader**
  An entity that receives DataSetMessage from message-oriented middleware. (See References 1, 3.1.3.)
- **DataSetWriter**
  An entity that creates a DataSetMessage from a DataSet and publishes it through message-oriented middleware, which has a unique ID(DataSetWriterId in.PublisherId. (See References 1, 3.1.4)
- **PublishedDataSet**
  The configuration of the application data that is issued as a DataSet. (See References 1, 3.1.5.)
- **SubscribedDataSet**
  This configuration dispatches the received DataSets. (See References 1, 3.1.7.)
- **NetworkMessage**
  Containers for storing DataSetMessages, which contain data that is shared between.DataSetMessages. (See References 1, 5.3.4)
- **DataSetMessage**
  It is created from a DataSet and consists of headers and DataSet encoded fields. (See References 1, 5.3.3)
- **DataSet**
  A list of name-value pairs representing a list of events or variable values. (See References 1, 5.2.1)
- **Publisher**
  A PubSub entity that sends NetworkMessage to message-oriented middleware and has a unique ID(PublisherId in.Message Oriented Middleware. (See References 1, 5.4.1.1)
- **Message Oriented Middleware**
  An infrastructure that supports sending and receiving NetworkMessage between distributed systems. (See References 1, 5.4.4.1)
- **PubSubConnection**
  Combination of protocol selection, protocol settings, and addressing information. (See References 1, 9.1.5.2)
- **ReaderGroup**
  A grouping of lists of DataSetReaders. (See References 1, 6.2.7.2.1)

## Appendix B. Introduction to OPC UA PubSub

Refer to References 1 and Reference 2 to introduce the outline of OPC UA PubSub.

OPC UA Publisher sends the data and OPC UA Subscriber receives the data.

- There is no need to manage connection information depending on the number.
- Multiple simultaneous data exchange is possible.
- OPC UA communication security is available.
  - ➢ This provider is not supported.

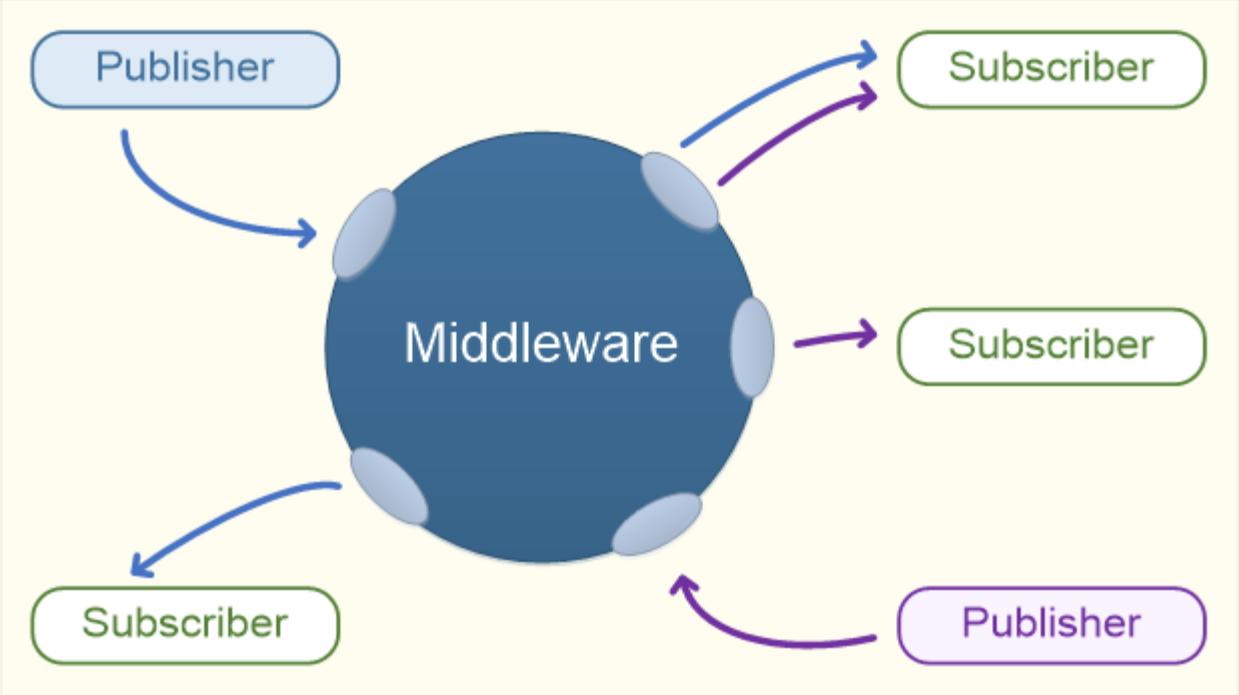


**Figure 3-1 Publish Subscribe Model Overview (See References 1)**

Transmitting data by Publisher has the following features:

- OPC UA UDP(UADP) High-speed multicast communication.
  - ➢ Unicast communication is also possible.
- Session-less communication using Broker (Message-Oriented Middleware).

Data reception by Subscriber has the following characteristics:

- You can specify the necessary data to receive.
  - ➢ This provider supports UADP multicast and UADP unicast.

OPC UA Publisher has a unique PublisherId in the networking. It also grants DataSetWriter,WriterGroup a unique DataSetWriterId,WriterGroupId for Publisher, which.DataSetWriter encode into DataSetMessage from DataSetMetaData and collected data. A WriterGroup representing a set of one or more DataSetWriter is encoded in NetworkMessage from the associated DataSetMessage, and.Publisher sends this NetworkMessage.

OPC UA Subscriber filters OPC UA Publisher {PublisherId, [WriterGroupId,] DataSetWriterId} for NetworkMessage, and.OPC UA Subscriber obtains DataSetMetaData needed to process NetworkMessage from OPC UA Publisher.

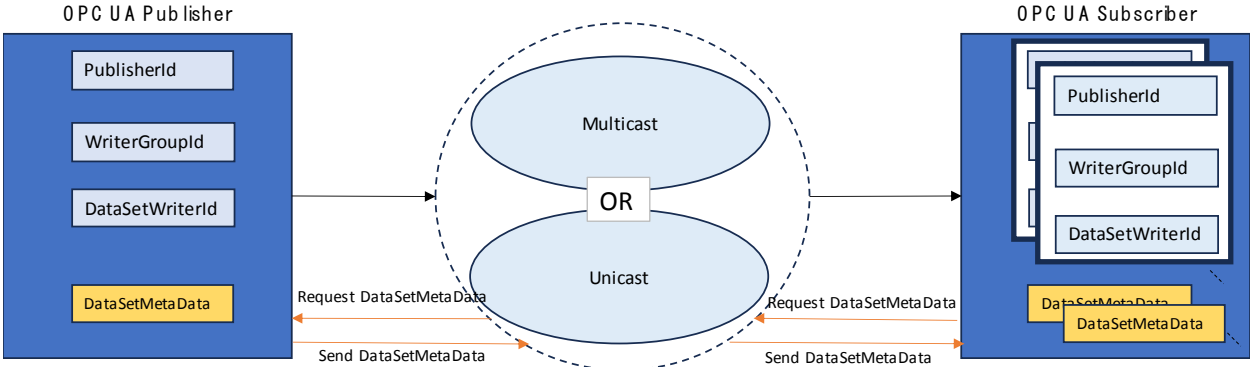


**Figure 3-2 Information required for NetworkMessage filtering and processing**

## Appendix C. References


- Reference 1:
  - OPC Foundation. OPC 10000-14: UA Part 14:PubSub v1.04. OPC Foundation.2018-02-06
  - https://reference.opcfoundation.org/Core/Part14/v104/docs
- Reference 2:
  - Torakazu satomura, OPC Foundation. Introduction and demonstration of OPC UA Pub/Sub. OPC Foundation.2018-12-14
  - https://jp.opcfoundation.org/wp-content/uploads/sites/2/2018/12/5_Satomura_PubSub.pdf