# OPC UA Publisher providers

## 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 book is taken from Reference 1.

Some of the Appendix B in this book is cited from Reference 2.

## [Revision History]

| Version | Date | Description |
|---|---|---|
| 1.0.0 | 2024-3-18 | First edition. |
| 1.1.0 | 2024-12-16 | Update of internally used library. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Table of Contents

# 1. Introduction

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

OPC UA Publisher providers can issue OPC UA UDP messages (hereinafter referred to as messages) and send them to message-oriented middleware.

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

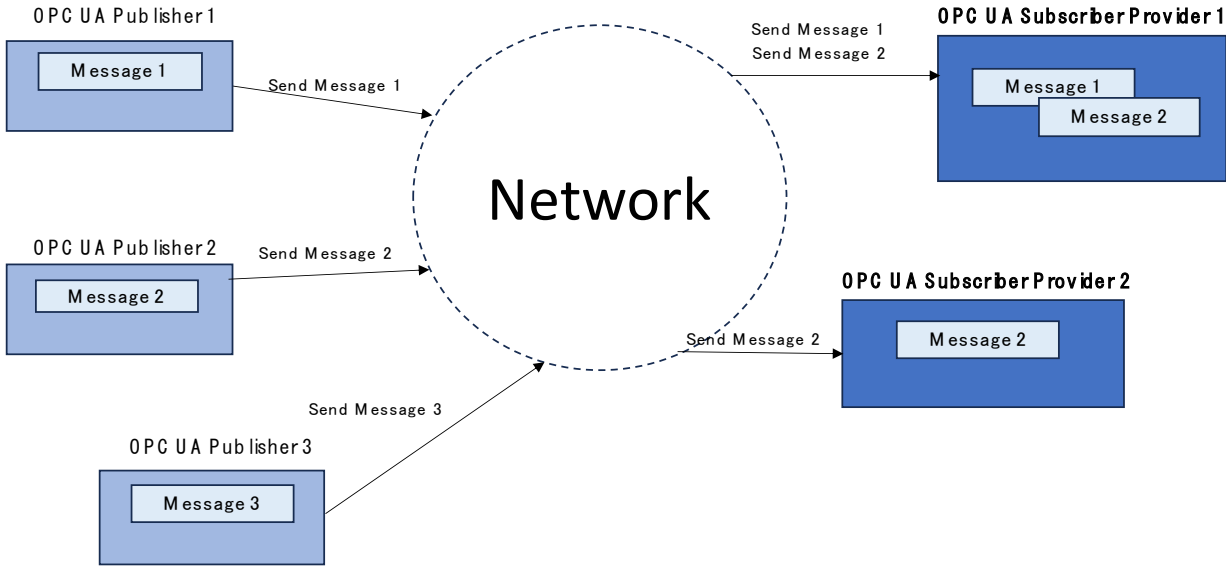


**Figure 1-1 OPC UA Publisher Providers Summary**

# 2. Provider Overview

## 2.1. Overview

OPC UA Publisher providers are CAO providers that provide messaging capabilities.

Its filetype is DLL(Dynamic Link Library) and is dynamically loaded at use from CAO. It is used to send a message to the message-oriented middleware and notify Subscriber of the message.

**Table 2-1 OPC UA Publisher Providers**

| | |
|---|---|
| File name | CaoProvOPCUAPublisher.dll |
| ProgID | CaoProv.OPCUA.Publisher |
| Registry registration | Regsvr32 CaoProvOPCUAPublisher.dll |
| Deleting registry entries | Regsvr32 /u CaoProvOPCUAPublisher.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>]

- For OPC UA Publisher 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 Publisher providers

OPC UA Publisher providers provide the following features:

➢ Sending a Message

- Messages can be sent periodically to Subscriber through the message-oriented middleware.
- Variable is used to send messages.

  You can create more than one Variable, each sending a separate messaging. You can use a PublisherId, WriterGroupId, DataSetWriterId to identify the message.
- Specify the period interval when sending messages.
  - Use PublishingInterval, KeyFrameCount to specify the period.
    - ✧ The period is determined by PublishingInterval×KeyFrameCount.

- ✧ If KeyFrameCount is 1, messages are continuously sent at PublishingInterval intervals.
- ✧ If KeyFrameCount is greater than or equal to 2, the first message is sent when PublishingInterval time has elapsed, and then the delta checking is performed every time PublishingInterval time has elapsed within the cyclic interval. If there is a difference, a message is sent even before the period interval elapses. If there is no difference, a message is sent continuously in the period interval. For details, see section 2.2.4.2.

- Attempting to send a message using an unsupported type results in an unsupported error.
  See 2.2.6 for supported types.
- Messages are not sent at the timing at which AddVariable was performed, but are started at the timing at which put_Value was performed. However, this does not apply if a WaitDataSetWriter or StartPublish was specified during AddVariable. For details, see 2.2.4.3 and 2.2.4.4.
- The maximum message length is 65535 bytes. This includes the lengths of IP headers, UDP headers, various headers of UADP messages, and options.

➢ Generating and sending meta-information (DataSetMetaData)

- OPC UA Publisher providers generate meta-information to ensure that OPC UA Subscriber gets the information correctly from the messages they send.
- OPC UA Publisher providers may receive a meta information request from OPC UA Subscriber, in which case the meta information is sent.
- The meta information retains the creation time as a version.

➢ Reading out the last set message

When the message is read (CaoVariable::get_Value), the data set as the last data for transmission is read.

- If no message has been set, VT_EMPTY is returned.

➢ Notes

- If you specify a KeyFrameCount other than 1, DataSetWriter associated with that WriteGroup is single, and no other DataSetWriter can be associated.

### 2.1.4. Flow of Transmission and Reception of OPC UA Publisher and OPC UA Subscriber Providers

Apart from OPC UA Publisher providers, the receiving OPC UA Subscriber providers (CaoProvOPCUASubscriber.dll) are separately implemented.

This section describes how to communicate with OPC UA Subscriber providers.

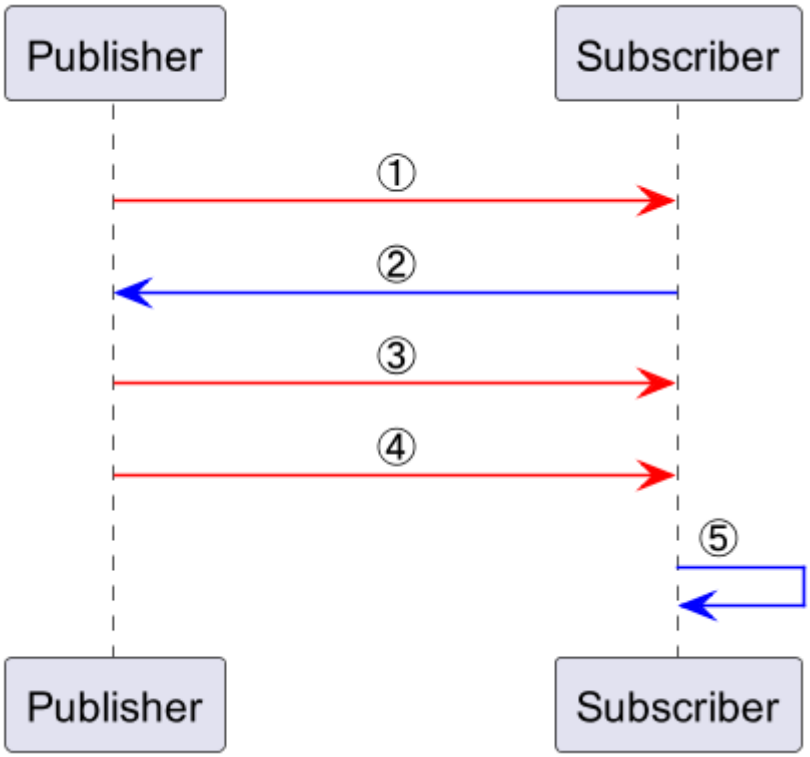


**Figure 2-1 Flow when Subscriber is already registered in PubsubConnection at startup**

①Send WriterData at Publisher startup

②To receive WriterData and request meta-data

③Send meta-data by receiving a request from Subscriber

④Sending a network message

⑤Use received metadata to convert messages

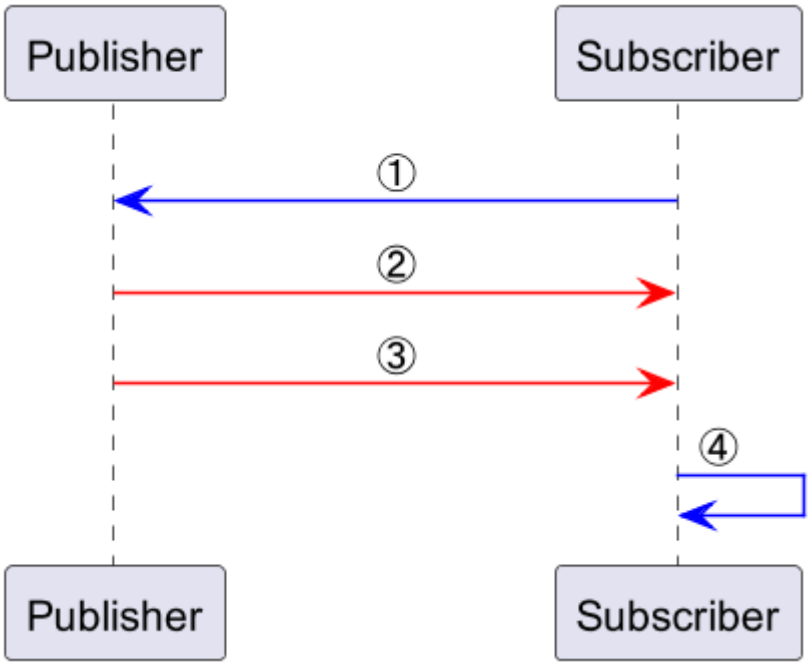


**Figure 2-2 Flow when registered in PubsubConnection before Subscriber**

①Requesting meta-data at Subscriber startup

②Send meta-data by receiving a request from Subscriber

③Sending a network message

④Use received metadata to convert messages

## 2.2. Method Properties

### 2.2.1. CaoWorkspace::AddController method

OPC UA Publisher 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.Publisher"
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 = <destination IP> | Required specification item.<br>Specify the connection parameters.<br>(Refer to 2.2.1.1) |
| DiscoveryUrl[=<DataSetMetaData's Request reply destination >] | Optional field.<br>IP addressing for Request reception and reply transmission of DataSetMetaData.<br>(Default: opc.udp://224.0.2.14:4840)[3]<br>(Refer to 2.2.1.2) |
| NetworkInterface<br>[=<IP addressof the destination NIC>] | Optional field.<br>Specifies the destination network interface.<br>(Default: 0.0.0.0)<br>(Refer to 2.2.1.3) |
| PublisherIdDataType = <data type> | Required specifications.<br>Specifies the data type of PublisherId.<br>(Refer to 2.2.1.4) |
| PublisherId=<ID number/character string> | Required specifications.<br>Specify PublisherId. |

[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.

| | |
|---|---|
| | (Refer to 2.2.1.5) |

#### 2.2.1.1. UADPUrl Optional

UADPUrl optional parameter string is shown below.

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

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>Specifies the address of the communication format to be used from the destination multicast address or unicast address.[5]<br>Example :"239.0.0.1" |
| <port> | : | Specifies the connection port number of the destination. (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 Request reception and reply transmission of DataSetMetaData.<br>Specifies a multicast address.[6]<br>(default: 224.0.2.14) |
| <port> | : | Specify the connection port number.<br>(default: 4840) |

#### 2.2.1.3. NetworkInterface Optional

NetworkInterface optional parameter string is shown below.

"NetworkInterface[=<IP addressof the destination NIC>]"

| | | |
|---|---|---|
| <IP of the destination NIC> | : | IP of the destination NIC.<br>0.0.0.0 works for all NIC. (Default: 0.0.0.0) |

[4] The "opc.udp" part connects "opc" and "udp" with a period.
[5] Multicast addresses range from 224.0.0.0 to 239.255.255.255.
[6] DiscoveryUrl cannot be unicast.

#### 2.2.1.4. PublisherIdDataType Optional

Specifies the type of PublisherId by number.

**Table 2-3 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[7] |

#### 2.2.1.5. PublisherId Optional

Specifies a value that is within the range of the type specified in PublisherIdDataType option. (See Table 2-3.)

If the same combination of PublisherIdDataType and PublisherId set in 2.2.1.4 is set, an error will result.

[7] Limited to the maximum value of the return type size_t of Std::string.size(). The maximum value for this provider is 4294967295.

### 2.2.2. CaoController::AddExtension method

AddExtension method of CaoController class uses CaoExtension object. It manages the information used to Publish with each provider in WriterGroup.

After you create Extension, you create a Variable that Extension uses to refer to the information managed by.Extension to remove WriterGroup.[8]

Format AddExtension(<bstrExtensionName:VT_BSTR>[,<bstrOption:VT_BSTR>])

<bstrExtensionName> : [in] Variable Name

<bstrOption> : [in] Option string

The following lists the option strings.

**Table 2-4 Optional String for CaoController::AddExtension**

| Option name [1] | Meaning [2] |
|---|---|
| WriterGroupId=<ID number > | Required specifications.<br>Specify WriterGroupId.<br>(Refer to 2.2.2.1) |
| PublishingInterval[=<interval to send data>] | Optional Items<br>Used for periodic transmission<br>Specifies the interval in milliseconds between sending messages.<br>(default: 1000)<br>(Refer to 2.2.2.2) |
| KeepAliveTime<br>[=<KeepAliveMessage sending interval >] | Optional information.<br>Specifies the number of milliseconds between KeepAliveMessage transmissions.<br>(default: 10000)<br>(Refer to 2.2.2.3) |
| MaxNetworkMessageSize<br>[=<NetworkMessage maximum size>] | Optional information.<br>Specifies the largest NetworkMessage to be sent in bytes. |

[8] Because the connection is temporarily OFF during deletion, all Variable created from the same Controller are unable to send messages until the deletion is completed.

| | |
|---|---|
| | (default: 1400)<br>(Refer to 2.2.2.4) |
| WaitWriterGroup[=<TRUE/FALSE>] | Optional information.<br>Specify WriterGroup creation timing.<br>(Default: FALSE)<br>(Refer to 2.2.2.5) |
| AutoRebootWriterGroup[=<TRUE/FALSE>] | Optional information.<br>Specifies whether WriterGroup is automatically restarted when the meta-data is changed.<br>(Default: TRUE)<br>(Refer to 2.2.2.6) |

#### 2.2.2.1. WriterGroupId Optional

Specify a value within the specifiable range (1 to 65535).

If there is a duplicate in the same controller, an error occurs.

#### 2.2.2.2. PublishingInterval Optional

Specify this operand when sending data periodically.

The interval at which data is sent is determined according to the specified numeric value.

PublishingInterval[=<interval to send data>]

<Interval for sending data> : 1[9]～2147483647

See Appendix C for details on the type and timing of data to be transmitted.

#### 2.2.2.3. KeepAliveTime Optional

Specifying a value smaller than PublishingInterval option for this option results in an error.

If this option is not specified and PublishingInterval is larger than KeepAliveTime default value of 10000 milliseconds, the value of this option is automatically changed to the value equivalent to PublishingInterval.

When this option is specified, KeepAliveMessage is sent with the following timing/conditions.

[9] If you specify a short sending interval, messages may not be sent at the specified sending interval depending on the performance of OS.

| | |
|---|---|
| Timing | Timing when the time of the first PublishingInterval has elapsed after the time of KeepAliveTime has elapsed since the last time data was sent |
| Conditions | When there is no difference in data difference checking (DeltaFrameMessage is sent if there is a difference) |

KeepAliveTime[=<KeepAliveMessage sending interval >]

<Interval for sending KeepAliveMessage> : 1～2147483647

See Appendix C for details on the type and timing of data to be transmitted.

#### 2.2.2.4. MaxNetworkMessageSize Optional

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

Data larger than the set value is not sent.[10]

Data exceeding the maximum size is divided and sent.

MaxNetworkMessageSize[=<NetworkMessage maximum size >]

<Maximum size of data to send> : 1400[11]～65535[12]

#### 2.2.2.5. WaitWriterGroup Optional

Specifies the timing at which WriterGroup is created.

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

Messages cannot be sent (no Publish is made) during standby.

To create a standby WriterGroup so that messages can be sent, execute CreateWriterGroup command.

When put_Value is executed in the standby status, Publish is not started, and CreateWriterGroup is started with the send data set by put_Value at the timing of executing the command.

If you specify FALSE, WriterGroup is created immediately.

However, when WriterGroup is created, the connection is temporarily OFF, so the message cannot be sent on all Variable created from the same Controller until the creation is completed.[13]

#### 2.2.2.6. AutoRebootWriterGroup Optional

You may need to restart WriterGroup to notify Subscriber of DataSetMetaData changes when you change the type of outbound data. You can select this option to restart WriterGroup automatically or manually.

[10] No error notification is performed.
[11] If you specify a value that is less than the size of the data that cannot be partially divided, such as the configuration information to be sent when WriterData(Publish starts or the header portion of the message, you may not be able to send the data normally.
[12] This is actually a value including a header etc.
[13] Because the connectivity is temporarily turned OFF, the elapsed time of the send interval is reset.

When TRUE is selected, WriterGroup is automatically restarted when the type of the send data is changed to notify Subscriber of DataSetMetaData change.

If you select FALSE, Subscriber may not be able to receive any data because DataSetMetaData change notification is not sent to Subscriber, so to notify.Subscriber of DataSetMetaData change, execute RebootWriterGroup commandPlease restart WriterGroup.

However, when WriterGroup is restarted, the connection is temporarily OFF, so messages cannot be sent on all Variable created from the same Extension until the restart is completed.

When the type of the send data is changed, MetaData change notification message (see 2.2.8.2) is notified. Extention specified when.RebootWriterGroup command is executed can be retrieved from this message.

### 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] Execution result |

### 2.2.4. CaoExtension::AddVariable method

AddVariable method of CaoExtension is a method that creates a variable object for Publish with each provider.

OPC UA Publisher providers specify the data types and data to send after creating the variables. After executing the put_Value, they begin Publish. Run UnPublish on Remove Variable.[14]

If you add more than one Variable to a Extension, KeyFrameCount must be 1.

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

<bstrVariableName> : [in] Variable Name

<bstrOption> : [in] Option string

The following lists the option strings.

**Table 2-5 Optional String for CaoExtension::AddVariable**

| Option name [1] | Meaning [2] |
|---|---|
| DataSetWriterId=<ID number > | Required specifications.<br>Specify DataSetWriterId.<br>(Refer to 2.2.4.1) |
| KeyFrameCount<br>[=<Maximum number of times PublishingInterval will expire >] | Optional information.<br>Specifies the number of times a PublishingInterval can expire.<br>When this option is set to 2 or more, PublishingInterval sends messages only when the specified number of times has elapsed or the message has changed.<br>Use defaults if you want to associate more than one Variable with a single Extension.<br>(Default: 1)<br>(Refer to 2.2.4.2) |
| WaitDataSetWriter[=<TRUE/FALSE>] | Optional information. |

[14] Because the connection is temporarily OFF during deletion, all Variable created from the same Extension are unable to send messages until the deletion is completed.

| | |
|---|---|
| | Specify DataSetWriter creation timing.<br>(Default: FALSE)<br>(Refer to 2.2.4.3) |
| StartPublish[=<TRUE/FALSE>] | Optional Items<br>Specifies whether Publish starts.<br>(Default: TRUE)<br>(Refer to 2.2.4.4) |
| SendByteString[=<TRUE/FALSE>] | Optional Items<br>VT_UI1\| Specifies the VT_ARRAY transmission format.<br>(Default: FALSE)<br>(Refer to 2.2.4.5) |

#### 2.2.4.1. DataSetWriterId Optional

Specify a value within the specifiable range (1 to 65535).

If there is a duplicate in the same controller, an error occurs.

#### 2.2.4.2. KeyFrameCount Optional

Specifies the number of times a PublishingInterval can expire.

When this option is specified, the transmission period is set to the following timing.

However, the first message is sent at the timing when the first PublishInterval has elapsed.

Transmit Interval = PublishingInterval × KeyFrameCount

When this option is set to 2 or more, if PublishingInterval period elapses within the send cycle, data difference checking is performed, and if there is a difference, only the difference data is sent as a DeltaFrameMessage.[15]

If you want to associate more than one Variable with a single Extension, use 1 (the default) for this option.

KeyFrameCount[=<Maximum number of times PublishingInterval will expire>]

< Max number of times : 1～4294967295

PublishingInterval expires >

See Appendix C for details on the type and timing of data to be transmitted.

#### 2.2.4.3. WaitDataSetWriter Optional

Specifies the timing at which DataSetWriter is created.

[15] Because DeltaFrameMessage requires additional indexes to send the changed data, the data can be larger than KeyFrameMessage, in which case an KeyFrameMessage is sent.

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

Messages cannot be sent (no Publish is made) during standby.

To create a standby DataSetWriter so that messages can be sent, execute CreateDataSetWriter command.

If put_Value is executed in the standby status, Publish is not started, and Publish is started with the send data set by put_Value at the timing when CreateDataSetWriter command is executed.

If you specify FALSE, DataSetWriter is created immediately.

However, when DataSetWriter is created, the connection is temporarily OFF, so the message cannot be sent on all Variable created from the same Extension until the creation is completed.[16]

#### 2.2.4.4. StartPublish Optional

Specifies whether Publish starts.

If TRUE is specified, Publish starts at the timing when the put_Value is performed.

If FALSE is specified, no Publish is performed.

You can also specify them by running StartPublish command and StopPublish command, respectively.

If put_Value is executed with this option set to FALSE, Publish will not be started, and Publish will be started with the send data set with put_Value at the timing when this option becomes TRUE.

#### 2.2.4.5. SendByteString Optional

Specifies whether to send as ByteString type or as array of VT_UI1 type when executing transmission with content including a particular type (VT_ARRAY | VT_UI1).

If TRUE is specified, it is sent as a ByteString type.

If FALSE is specified, it is sent as an array of VT_UI1.

### 2.2.5. CaoExtension::Execute method

Executes the specified command generically.

Refer to 2.3.2 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] Execution result |

[16] Because the connectivity is temporarily turned OFF, the elapsed time of the send interval is reset.

### 2.2.6. CaoVariable::put_Value property

Set the transmission data and start transmission.

From the set VARIANT type, convert the datatype to OPC UA built-in datatype, create the corresponding DataSetMetaData, and run Publish.

DataSetMetaData keeps a versioned hour when it was created.

If the data type has not changed compared to the last sent Publish data, DataSetMetaData is not created, only the value is updated, and Publish is executed.

**Restrictions**

Data types not listed in the table below cannot be set.

Arrays of two or more dimensions are not supported.

**Table 2-6 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 |

### 2.2.7. CaoVariable::get_Value property

Acquires the last sent Publish.

If no data is set, VT_EMPTY is returned.

### 2.2.8. CaoController::OnMessage Events

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

**Table 2-7 Message types**

| Message type | | Trigger |
|---|---|---|
| 0 | Error during transmission operation | Occurs when an error occurs during transmission operation for some reason. |
| 1 | MetaData change notification | You are notified when the type of data being sent changes. |

#### 2.2.8.1. Error during transmission operation

The data format obtained by error messages during transmission is shown below.

Number : Message type (0)

Value : Internal error code

DateTime : Time stamp

Destination : AddExtension method <bstrExtensionName>

Source : AddVariable method <bstrVariableName>

Description : Error description information[17]

#### 2.2.8.2. MetaData change notification

The format of MetaData change notification is shown below.

Number : Message type (1)

Value : AutoRebootWriterGroup Optional Settings (VT_BOOL)[18]

DateTime : Time stamp

Destination : AddExtension method <bstrExtensionName>

Source : AddVariable method <bstrVariableName>

Description : Metadata update notification information

[17] If there is no description information related to the internal error code, an empty string is stored.
[18] Setting specified for AutoRebootWriterGroup option during AddExtension

## 2.3. Command list

### 2.3.1. CaoController class-

**Table 2-8 List of CaoController::Execute Commands**

| Command | Function | Reference |
|---|---|---|
| CreateWriterGroup | Specify TRUE in 2.2.2.5 to create a WriterGroup that is in standby status.<br>Since the connection is temporarily OFF when WriterGroup is created, messages cannot be sent on all Variable created from the same Controller until the creation is completed. | P.22 |
| StartVariablesPublish | Starts Publish of all Variable associated with Extension. | P.22 |
| StopVariablesPublish | Publish of all Variable associated with Extension is stopped. | P.23 |
| RebootWriterGroup | Restarts WriterGroup corresponding to the specified Extension.<br>Specifying a WriteGroup that does not have a valid DataSetWriter results in an error.<br>When WriterGroup is restarted, the connection is temporarily OFF, so no messages can be sent on all Variable created from the same Extension until the restart is completed. | P.23 |

## CreateWriterGroup

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

## StartVariablesPublish

| | |
|---|---|
| **Syntax** | Object.StartVariablesPublish(Data) |
| **Argument** | Data.VT=VT_BSTR<br>Data.bstrVal=ExtensionName |
| **Return value** | None |
| **Description** | Starts Publish on all Variable associated with the specified Extension. |

## StopVariablesPublish

| | |
|---|---|
| **Syntax** | Object.StopVariablesPublish(Data) |
| **Argument** | Data.VT=VT_BSTR<br>Data.bstrVal=ExtensionName |
| **Return value** | None |
| **Description** | Stops Publish on all Variable associated with the specified Extension. |

## RebootWriterGroup

| | |
|---|---|
| **Syntax** | Object.RebootWriterGroup(Data) |
| **Argument** | Data.VT=VT_BSTR<br>Data.bstrVal=ExtensionName |
| **Return value** | None |
| **Description** | Restarts WriterGroup corresponding to the specified Extension.<br>Specifying a WriteGroup that does not have a valid DataSetWriter results in an error.<br>When WriterGroup is restarted, the connection is temporarily OFF, so no messages can be sent on all Variable created from the same Extension until the restart is completed. |

### 2.3.2. CaoExtension class-

**Table 2-9 List of CaoExtension::Execute Commands**

| Command | Function | Reference |
|---|---|---|
| CreateDataSetWriter | In 2.2.4.3, specify TRUE to create a DataSetWriter that is in standby status.<br>Since the connection is temporarily OFF when DataSetWriter is created, messages cannot be sent on all Variable created from the same Extension until the creation is completed. | P. 24 |
| StartPublish | Starts Publish at the specified Variable. | P.24 |
| StopPublish | Stops Publish at the specified Variable. | P.24 |

## CreateDataSetWriter

| | |
|---|---|
| **Syntax** | Object.CreateDataSetWriter() |
| **Argument** | None |
| **Return value** | None |
| **Description** | In 2.2.4.3, specify TRUE to create a DataSetWriter that is in standby status.<br>Since the connection is temporarily OFF when DataSetWriter is created, messages cannot be sent on all Variable created from the same Extension until the creation is completed. |

## StartPublish

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

## StopPublish

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

## 2.4. Variable List

### 2.4.1. CaoController class-

**Table 2-10 List of CaoController Class System Variable**

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

### 2.4.2. CaoExtension class-

**Table 2-11 List of CaoExtension Class User Variable**

| Variable name | Data type | Description | Attribute | |
|---|---|---|---|---|
| | | | Get | Put |
| Optional | Data types that can be specified in put_Value (see 2.2.6) | After the variable is created, the data is created and sent with the data type and value specified at the timing of the put_Value. (Refer to 2.2.4) (Refer to 2.2.6) | ○ | ○ |

## 2.5. Error code

OPC UA Publisher providers define the following unique error codes:

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

**Table 2-12 Unique Error Codes21**

| Error Name | Error number | Description |
|---|---|---|
| E_FAILED_PUB_INVALID_PARAMS_UADPURL | 0x80100001 | The parameter (UADPUrl option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_PUBLISHERID DATATYPE | 0x80100002 | The parameter (PublisherIdDataType option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_PUBLISHERID | 0x80100003 | The parameter (PublisherId option) required for sending is invalid. |
| E_FAILED_PUB_DUPLICATE_PARAMS_PUBLISHER | 0x80100004 | The combination of |

| | | |
|---|---|---|
| ID | | PublisherIdDataType option and PublisherId option is duplicated. |
| E_FAILED_PUB_INVALID_PARAMS_WRITERGROUPID | 0x80100005 | The parameter (WriterGroupId option) required for sending is invalid. |
| E_FAILED_PUB_DUPLICATE_PARAMS_WRITERGROUPID | 0x80100006 | WriterGroupId options are duplicated. |
| E_FAILED_PUB_INVALID_PARAMS_PUBLISHINGINTERVAL | 0x80100007 | The parameter (PublishingInterval option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_KEYFRAMECOUNT | 0x80100008 | The parameter (KeyFrameCount option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_KEEPALIVETIME | 0x80100009 | The parameter (KeepAliveTime option) required for sending is invalid. |
| E_FAILED_PUB_KEEPALIVETIME_UNDER_PUBLISHINGINTERVAL | 0x8010000A | The parameter (KeepAliveTime option) required for the send process is smaller than PublishingInterval option. |
| E_FAILED_PUB_INVALID_PARAMS_MAXNETWORKMESSAGESIZE | 0x8010000B | The parameter (MaxNetworkMessageSize option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_WAITWRITERGROUP | 0x8010000C | The parameter (WaitWriterGroup option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_DATASETWRITERID | 0x8010000D | The parameter (DataSetWriterId option) required for sending is invalid. |
| E_FAILED_PUB_DUPLICATE_PARAMS_DATASETWRITERID | 0x8010000E | DataSetWriterId options are duplicated. |
| E_FAILED_PUB_INVALID_PARAMS_WAITDATASETWRITER | 0x8010000F | The parameter (WaitDataSetWriter option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_PARAMS_STARTPUBLISH | 0x80100010 | The parameter (StartPublish option) required for sending is invalid. |

| | | |
|---|---|---|
| E_FAILED_PUB_INVALID_PARAMS_SENDBYTESTRING | 0x80100011 | The parameter (SendByteString option) required for sending is invalid. |
| E_FAILED_PUB_INVALID_VALUEDATATYPE | 0x80100012 | The data type is not supported. |
| E_FAILED_PUB_CONVERT_DATA | 0x80100013 | Failed to convert the data to be sent. |
| E_FAILED_PUB_STARTUP_PUBS | 0x80100014 | Failed to set the information for sending.<br>Check AddController and AddExtension,AddVariable optional settings. |
| E_FAILED_PUB_LINK_DATASETWRITER | 0x80100015 | An attempt was made to add a KeyFrameCount to a Variable in a Extension with more than one Variable.<br>Or, an attempt was made to add a Variable to a Extension that already has a Variable with a KeyFrameCount of 2 or greater. |
| E_FAILED_PUB_INVALID_PARAMS_AUTOREBOOTWRITERGROUP | 0x80100016 | The parameter (AutoRebootWriterGroup option) required for sending is invalid. |
| E_FAILED_PUB_NO_VALID_DATASETWRITER_EXISTS | 0x80100017 | The command was executed for a WriteGroup that does not have a valid DataSetWriter. |

# 3. Appendix

## Appendix A. OPC UA PubSub Glossary

The terms that appear in the following OPC UA PubSub are cited from Reference 1.

- **DataSetClass**
  It has a.DataSetClassId(GUID that declares the content of DataSet. (See Reference 1 1, 3.1.1.)
- **DataSetMetaData**
  You can have a.DataSetClassId(GUID that explains what DataSet is and what it means. (See Reference 1 1, 3.1.2.)
- **DataSetReader**
  An entity that receives DataSetMessage from message-oriented middleware. (See Reference 1 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 Reference 1 1, 3.1.4)
- **PublishedDataSet**
  The configuration of the application data that is issued as a DataSet. (See Reference 1 1, 3.1.5.)
- **SubscribedDataSet**
  This configuration dispatches the received DataSets. (See Reference 1 1, 3.1.7.)
- **NetworkMessage**
  Containers for storing DataSetMessages, which contain data that is shared between.DataSetMessages. (see Reference 1 5.3.4)
- **DataSetMessage**
  It is created from a DataSet and consists of headers and DataSet encoded fields. (see Reference 1 5.3.3)
- **DataSet**
  A list of name-value pairs representing a list of events or variable values. (see Reference 1 5.2.1)
- **Publisher**
  A PubSub entity that sends NetworkMessage to message-oriented middleware. (see Reference 1 5.4.1.1)
- **Subscriber**
  A consumer of NetWorkMessages from message-oriented middleware, which can be a client of a.OPC UA, a server of a OPC UA, or an application for understanding the message structure of a OPC UAPubSub rather than a client or server. (see Reference 1 5.4.2.1)
- **Message Oriented Middleware**
  An infrastructure that supports sending and receiving NetworkMessage between distributed systems. (see Reference 1 5.4.4.1)
- **PubSubConnection**

Combination of protocol selection, protocol settings, and addressing information. (see Reference 1 9.1.5.2)

- **WriterGroup**

  A grouping of lists of DataSetWriters. (see Reference 1 6.2.5.6.1)

- **KeyFrameMessage**

  This message contains all the present values of PublishedDataSet. (see Reference 1 6.2.3.3)

- **DeltaFrameMessage**

  This message is only the difference from the last message sent. (see Reference 1 6.2.3.3)

- **KeepAliveMessage**

  This message informs Subscriber that Publisher is not lost if no data is sent for a long time. (see Reference 1 6.2.5.3)

# Appendix B. Introduction to OPC UA PubSub

Refer to Reference 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.

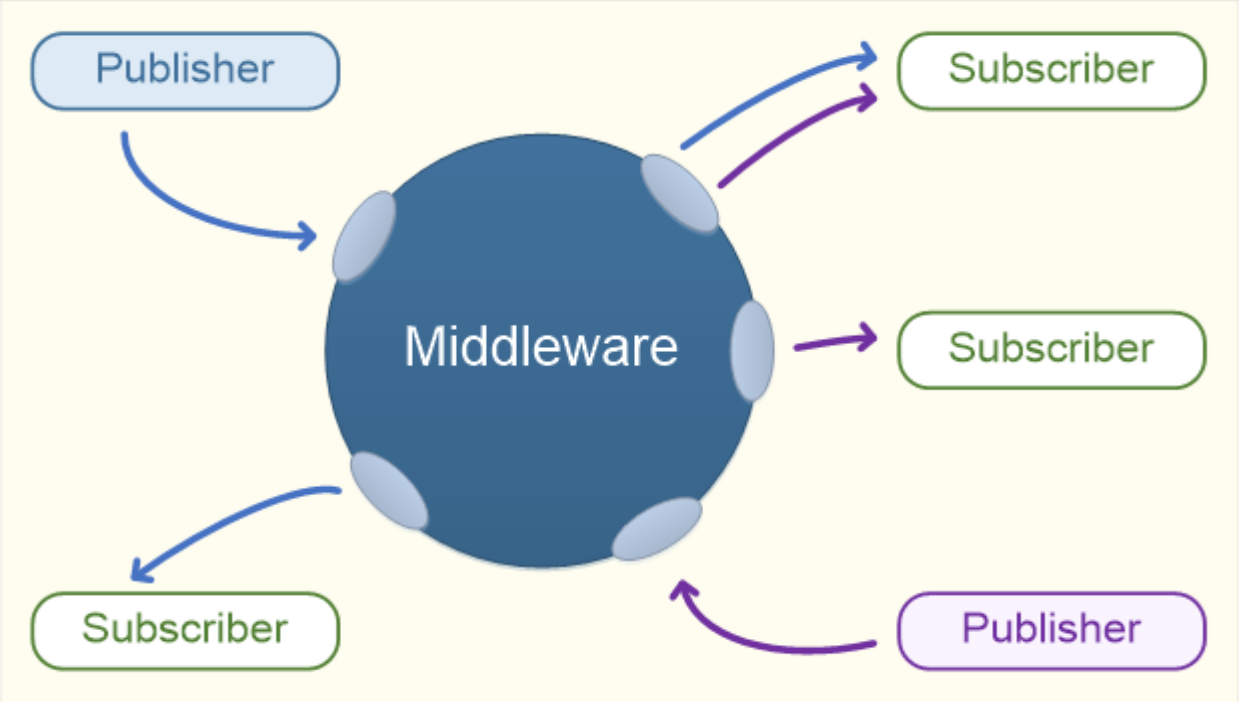


**Figure 3-1 Publish Subscribe Model Overview (see Reference 1)**

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.

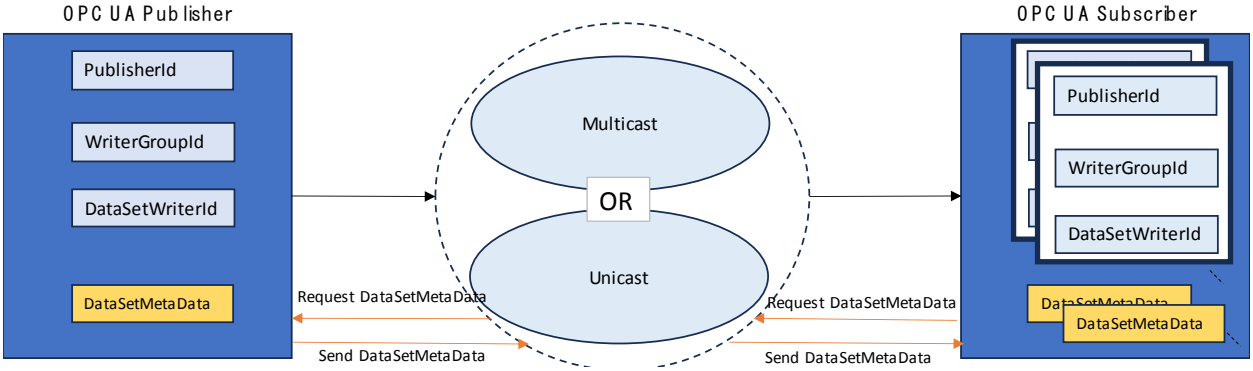


**Figure 3-2 Information required for NetworkMessage filtering and processing**

# Appendix C. Data Transmission Timing

The data transmission timing and the content of the data to be transmitted differ depending on the settings of "2.2.2.2.PublishingInterval option", "2.2.2.3.KeepAliveTime option" and "2.2.4.2.KeyFrameCount option".

The following timing is detailed for the transmission timing of various data.

| | |
|---|---|
| KeyFrameMessage | Timing when the sending interval (multiplication of PublishingInterval and KeyFrameCount) has elapsed |
| DeltaFrameMessage | When a difference is found by performing a differential check with the previous data at the timing when PublisherInterval has elapsed within KeyFrameMessage send interval |
| KeepAliveMessage | When KeepAliveTime time has elapsed since the last data was sent within KeyFrameMessage send cycle and the difference is not found after PublisherInterval time has elapsed by checking the difference with the previous data. |

The following figure shows an example pattern. Refer to the appropriate pattern in the figure.

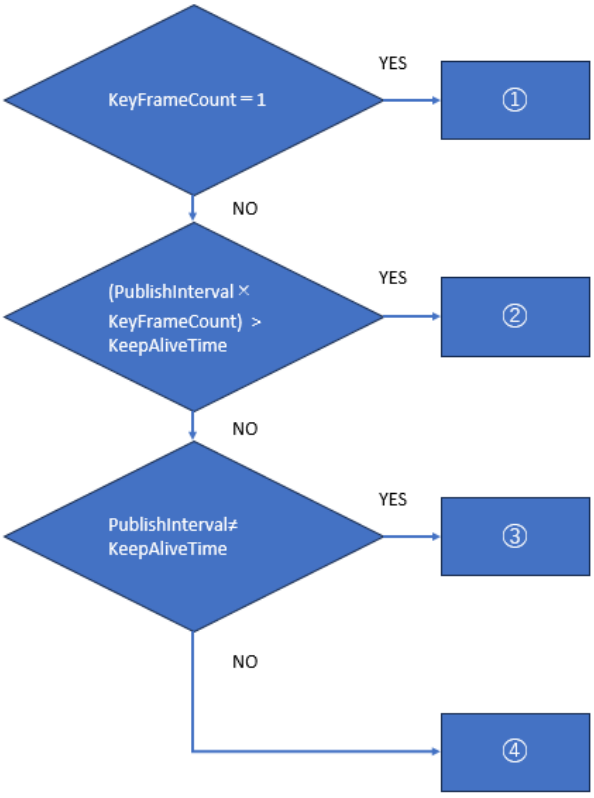


**Figure 3-3 Pattern flow**

- **Pattern①When the setting value of KeyFrameCount option is 1 (default value)**

With this pattern, no KeepAliveMessage or DeltaFrameMessage is sent, and the data is sent at each PublishingInterval.

| | |
|---|---|
| PublishInterval | 100ms |
| KeyFrameCount | 1(default) |
| KeepAliveTime | 10000ms(default) |

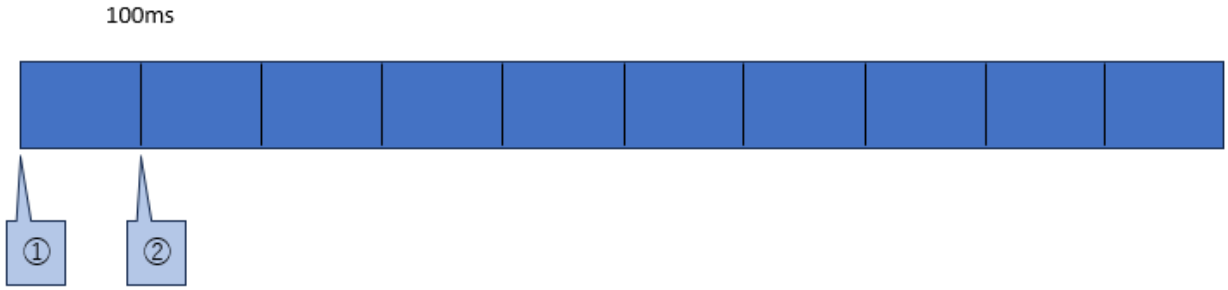


**Figure 3-4 Pattern diagram ①**

① Initial message transmission

② KeyFrameMessage is sent every 100 milliseconds

- **Pattern ②When KeyFrameCount option is set to 2 or more and KeepAliveTime option is not within the send interval**

For this pattern, the transmission period is set to the multiplication of PublishingInterval and KeyFrameCount options, and the difference between the previous transmission message and the transmission message is checked every time PublishingInterval elapses within the transmission period.

If the difference is confirmed, only the difference is sent as a DeltaFrameMessage.

Also, no KeepAliveMessage is sent for this pattern.

| | |
|---|---|
| PublishInterval | 100ms |
| KeyFrameCount | 5 |
| KeepAliveTime | 10000ms(default) |

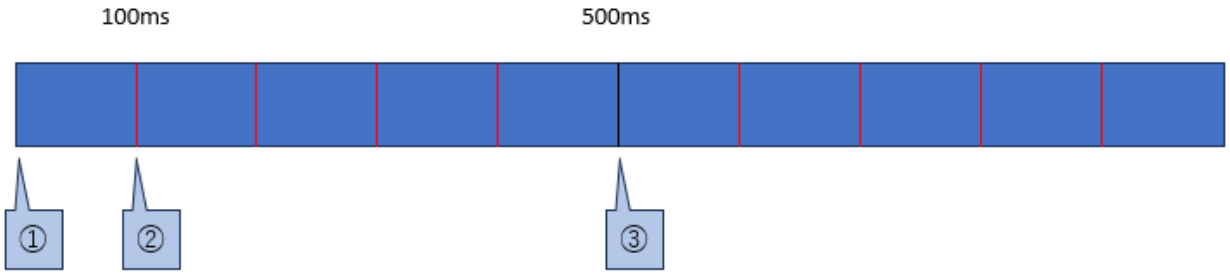


**Figure 3-5 Pattern Diagram ②**

① Initial message transmission.

② After every 100 milliseconds, the previous send data and the difference check are performed. If there is a difference, only the difference is sent as a DeltaFrameMessage. [15]

③ KeyFrameMessage is sent every 500 milliseconds.

※Note that if a KeyFrameMessage is sent in the middle, it will be counted again by 500 milliseconds.

- **Pattern ③When KeyFrameCount option is set to 2 or more and KeepAliveTime option is within the send interval**

For this pattern, the transmission period is set to the multiplication of PublishingInterval and KeyFrameCount options, and the difference between the previous transmission message and the transmission message is checked every time PublishingInterval elapses within the transmission period.

If the difference is confirmed, only the difference is sent as a DeltaFrameMessage.

If there is no difference from the previous message during the differential check and KeepAliveTime has elapsed since the previous message was sent, a KeepAliveMessage is sent.

| | |
|---|---|
| PublishInterval | 100ms |
| KeyFrameCount | 5 |
| KeepAliveTime | 210ms |

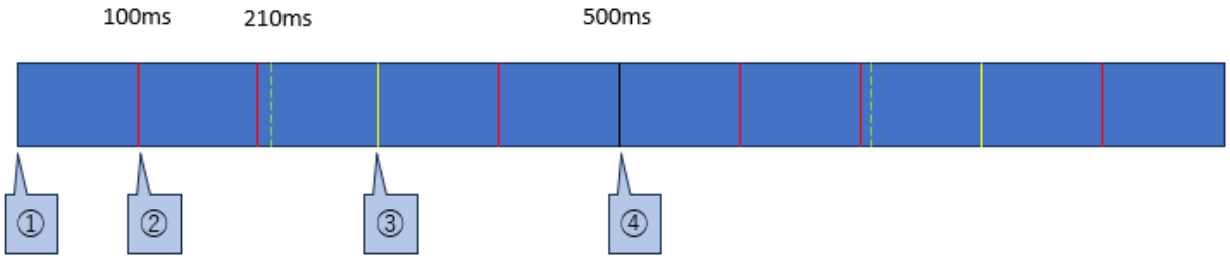


**Figure 3-6 Pattern Diagram ③**

① Initial message transmission

② Every 100 milliseconds, the previous send data and the difference check are performed. If there is a difference, only the difference is sent as a DeltaFrameMessage.
At this timing, KeepAliveTime period of 210 milliseconds has not elapsed, so no KeepAliveMessage is sent even if there is no difference.

③ After KeepAliveTime of 210 milliseconds (*) has elapsed since the previous data was sent, if there is no difference in the first differential check, a KeepAliveMessage is sent. (If there is a difference, a DeltaFrameMessage is sent.)
※Note that if a KeyFrameMessage or DeltaFrameMessage or KeepAliveMessage is sent, it will be counted again by 210 milliseconds.

④ Data is sent every 500 ms
※Note that if a KeyFrameMessage is sent in the middle, it will be counted again by 500 milliseconds.

- **Pattern ④ When KeyFrameCount option is set to 2 or more and KeepAliveTime option is equivalent to PublishingInterval option**

For this pattern, the transmission period is set to the multiplication of PublishingInterval and KeyFrameCount options, and the difference between the previous transmission message and the transmission message is checked every time PublishingInterval elapses within the transmission period.

If the difference is confirmed, only the difference is sent as a DeltaFrameMessage.

If there is no difference from the previous message during the differential check and KeepAliveTime has elapsed since the previous message was sent, a KeepAliveMessage is sent.

However, in this pattern, KeepAliveMessage may not be sent because the differential checking is completed before KeepAliveTime elapses due to the processing timing.

In such cases, it is judged whether to send KeepAliveMessage at the elapsed timing of the following PublishingInterval.

**Behavior when KeepAliveTime duration has elapsed**

| | |
|---|---|
| PublishInterval | 100ms(default) |
| KeyFrameCount | 5 |
| KeepAliveTime | 100ms |

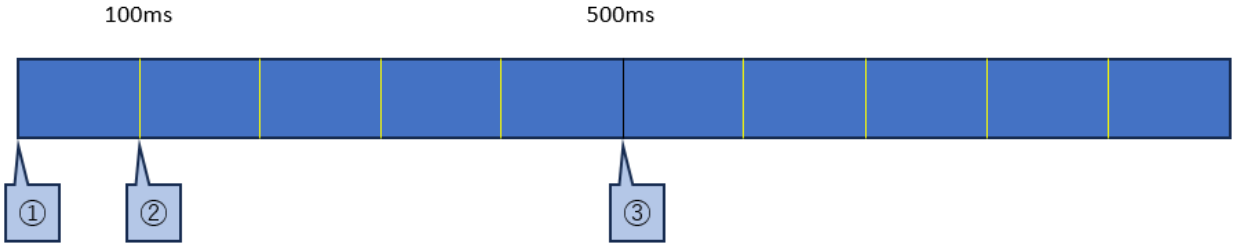


**Figure 3-7 Pattern Diagram ④131**

① Initial message transmission.

② After every 100 milliseconds, the previous send data and the difference check are performed. If there is a difference, only the difference is sent as a DeltaFrameMessage.
If there is no difference, a KeepAliveMessage is sent if KeepAliveTime period has elapsed since the last sent message.

③ Data is sent every 500 ms.
※Note that if a KeyFrameMessage is sent in the middle, it will be counted again by 500 milliseconds.

**Behavior when KeepAliveTime has not elapsed**

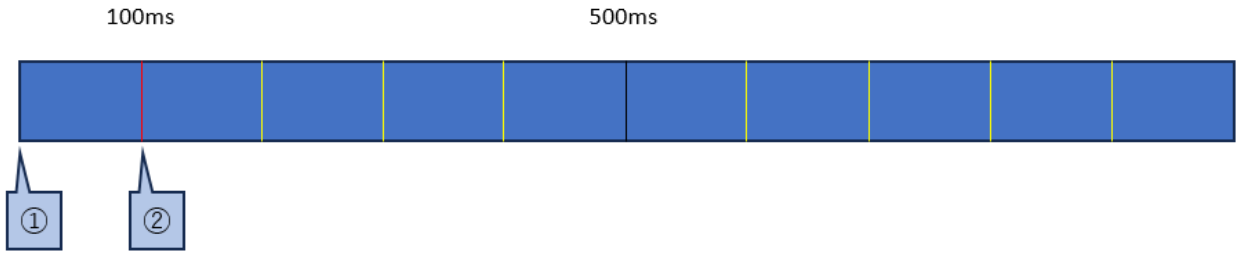


**Figure 3-8 Pattern Diagram ④-2**

① Initial message transmission.

② Difference checking is performed when PublishingInterval time of 100 milliseconds has elapsed since the previous message was sent, and if there is no difference, a KeepAliveMessage should be sent. However, if there is a difference of a few milliseconds in the message transmission timing, KeepAliveTime time may not have elapsed since the previous message was sent, and KeepAliveMessage may not be sent.
If there is no difference when the next PublishingInterval elapses, a KeepAliveMessage will be sent.

# Appendix D. References and 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.pdfhttps://jp.opcfoundation.org/wp-content/uploads/sites/2/2018/12/5_Satomura_PubSub.pdf