DataQueue provider

General-purpose data queue

Version 1.0.5

User's guide

September 3, 2021

【 remarks 】

This document is translated from Japanese into English by the machine translation.

【 revision history 】

Version	Date	Content
1.0.0	2017-10-08	First edition.
1.0.1	2017-12-08	For/..restoring.. file backup
1.0.2	2018-01-08	Indispensable manual backup/making list [a] function deletion and QFile parameters
1.0.3	2018-01-15	Added mode that does not save files, added QDir option.
1.0.4	2018-12-27	Add restriction on @QSNAP when QFile option is specified.
1.0.5	2020-02-27	Fixed error when allocating memory.
	2020-06-30	Added the relationship diagram between CAO object and queue. Corrected the description of AddExtension function.
	2021-05-24	Added notes to AutoBackup bit.
	2021-07-26	Corrected variable name to uppercase
	2021-09-03	Corrected the heading number of the Execute command

Contents

1. Introduction

This book is a user of DataQueue provider that offers the function to achieve queue processing guide.

The DataQueue provider can manage two or more cues internally. Each cue maintains data by the list structure of First-In First-[out] (FIFO), and offers the function to take [out] data sequentially if necessary.

This book explains mounting function of this DataQueue provider and variable and method.

2. Outline of provider

2.1. Outline

The queue processing offered in the DataQueue provider has been achieved by cue (Queue) structure.

This cue can be managed by the CaoExtension class of CAO(Controller Access Object) of ORiN2, and it operate it to the cue by the variable and the method that mounts in the CaoExtension class.

In the DataQueue provider, two or more cues can be managed at the same time by generating two or more CaoExtension objects through the object of CaoController. One CaoExtension object corresponds to one cue.

The Outline of the operation to the cue offered by the DataQueue provider is shown as follows.

The relationship between CAO objects and queues and the outline of operations for queues provided by the DataQueue provider are shown below.

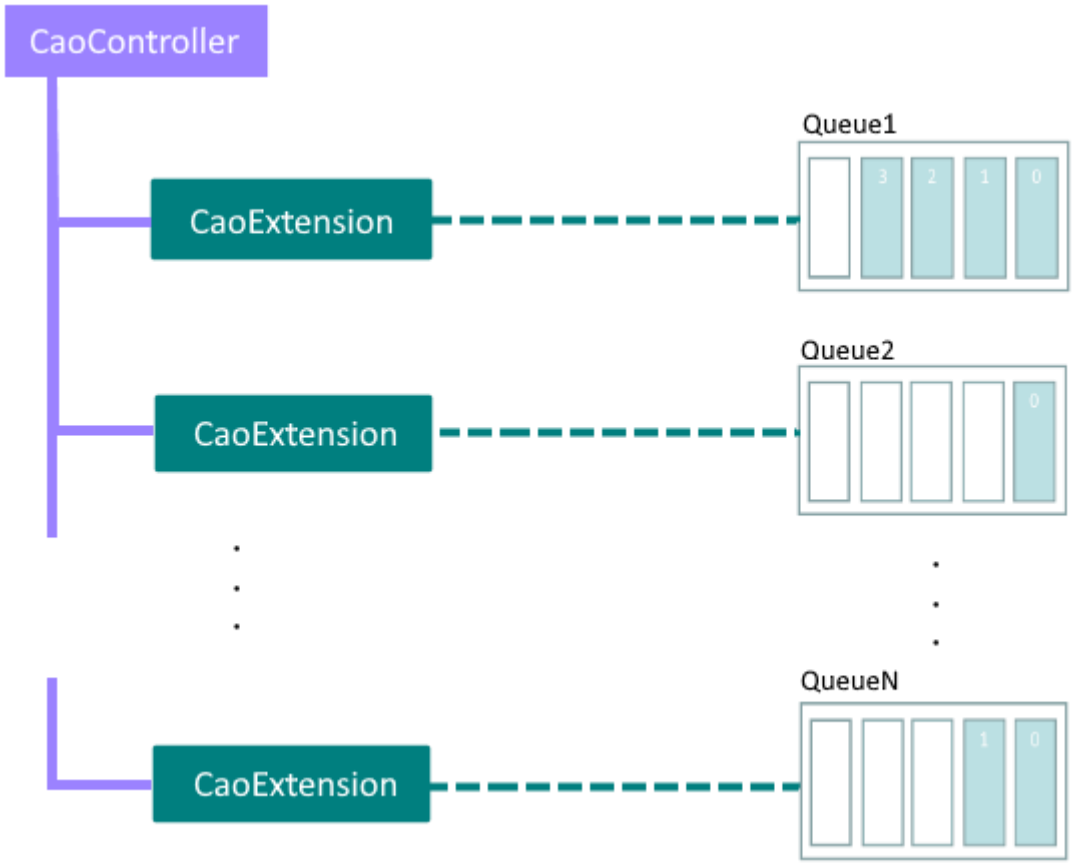


Figure 2-1 Relationship between CAO objects and queues

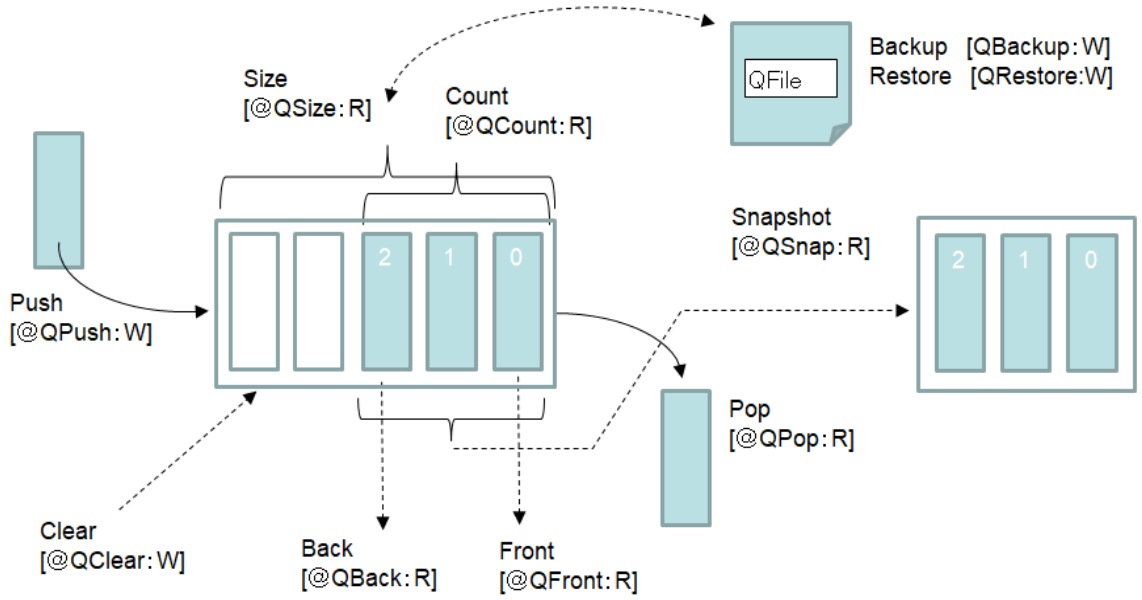


Figure2-1 Outline of the operation to the queue offered

2.2. Installation

The module of the DataQueue provider is composed of following DLL.

The installation work is unnecessary when installing it with the installer of ORiN2 SDK.

Please execute it as shown in Table 2-1 when you install it by hand power.

Table2-1DataQueue provider

File name	CaoProvDataQueue.dll
ProgID	CaoProv.DataQueue
[1]Registry registration	regsvr32 CaoProvDataQueue.dll
Blotting [out] of registry registration	regsvr32 /u CaoProvDataQueue.dll

2.3. Method property

2.3.1. CaoWorkspace::AddController method

A virtual controller name is specified in the DataQueue provider at AddController.

The argument specification of AddController is shown as follows.

Format AddController (<bstrCtrlName:BSTR>, <bstrProvName:BSTR>, <bstrPcName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName >	:	controller name
< bstrProvName >	:	provider name. fixed value ="CaoProv.DataQueue "
<bstrPcName>	:	Execution machine name of in provider (unused)
<bstrOption>	:	optional character string (unused)

The example when the AddController method is executed is shown as follows.

```
AddController
(
  "DQ1",                                    // virtual controller name =    DQ1
  "CaoProv.DataQueue",
  "",
  ""
);
```

[1] It is not necessary registration/to blot out by hand power when installing it with ORiN SDK.

2.3.2. CaoController::AddExtension method

Each cue has been achieved with the CaoExtension object in the DataQueue provider.

To make the cue, AddExtension is executed for the CaoController object.

The argument specification of AddExtension is shown as follows.

Format AddExtension (<bstrName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName > : In cue name

An arbitrary name is specified.

<bstrOption> : In optional character string

The following options can be specified by switching off the comma district.

QSize : Specify the maximum number of data that can be stored in the queue. (Optional)

The value is an integer value and the range can be 0 to 2147483647.

By default, 0 is used as the default value.

[Maximum number of data when QSize is 0]

QFile option	Max size
Specified	2147483647
Unspecified	1000

Example)

QSize=100

QMode : Behavior corresponding to the operation of the cue is set. (It is possible to omit it.)

When omitting it, 0 is used as a default value.

The following value is bitting specified.

QMode =

[AutoBackup:4bit]
[SwapData:3bit]
[FullData:2bit]
[NoEmpty:1bit]
[NoOverFlow:0bit]
Please refer to 2.3.2.1 for details.

Example)
There is no error at QMode=1 // Push.
QMode=8 // Swap

QDir : Specify the directory to use when saving files in the queue.
When this parameter is omitted, the environment variable
The value of DATAQUEUE_PROVIDER_DIR is used.

QFile : Specify the prefix of the file name to be used when saving the file in the queue.
It is also possible to specify a subfolder by separating it with /.
This parameter is used when AutoBackup is executed and when data is saved to a file when queue size becomes large.
Required when AutoBackup is specified in QMode.
If not specified, data is not saved to the file.
If QFile is specified and QDir and the environment variable DATAQUEUE_PROVIDER_DIR are not set, an error occurs.
Example)

If you specify QDir = C: / Temp, QFile = Queue, files such as Queue.fro, Queue.bak, Queue.1 are generated under C: / Temp.
Please check the Precautions in 2.3.2.1.5 when using multiple DataQueue providers.
Restrictions:
When QFile is specified, the file is saved, so the maximum number of items to be acquired with @QSNAP and GetQueueSnapshot is 500.

The example when the AddExtension method is executed is shown as follows.

```
AddExtension
(
        // Queue name ＝ Q1
        "Q1",
        // No error at push, backup directory "C: /Temp", backup file prefix "Queue"
        "QMode=1,QDir=C:/Temp,QFile=Queue"
);
```

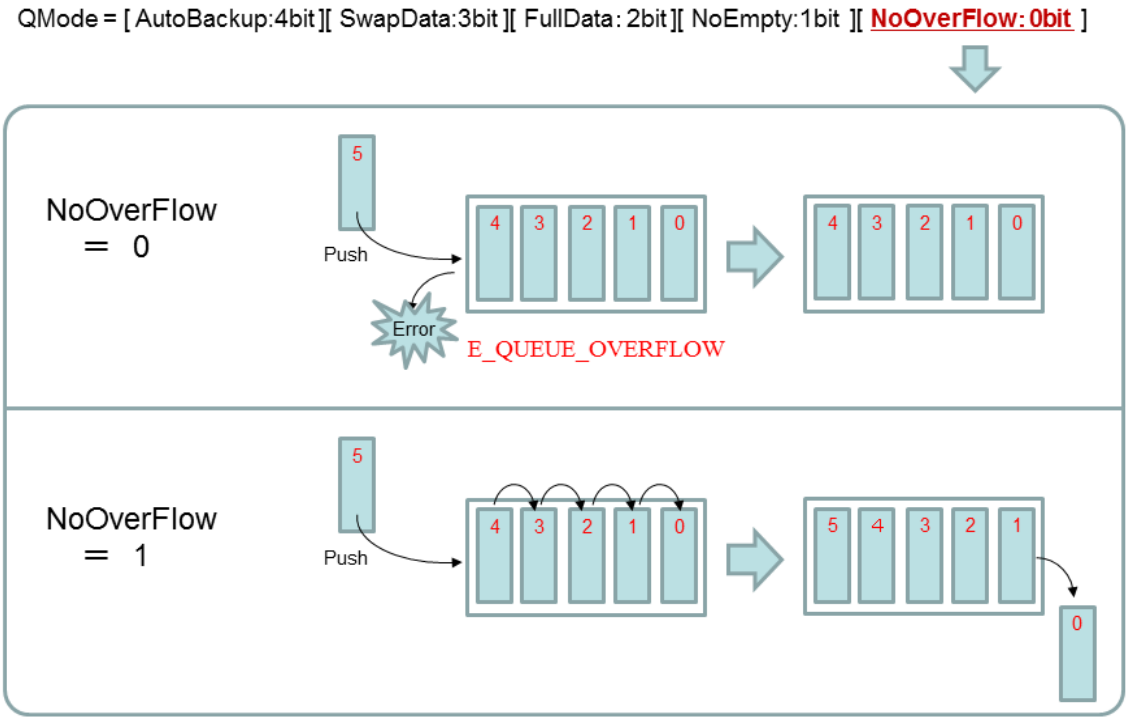
2.3.2.1. QMode is optional.

The behavior of the made cue can be set by bitting specifying the value for optional QMode of the AddExtension method.

The specification method in each bit of optional QMode is shown as follows.

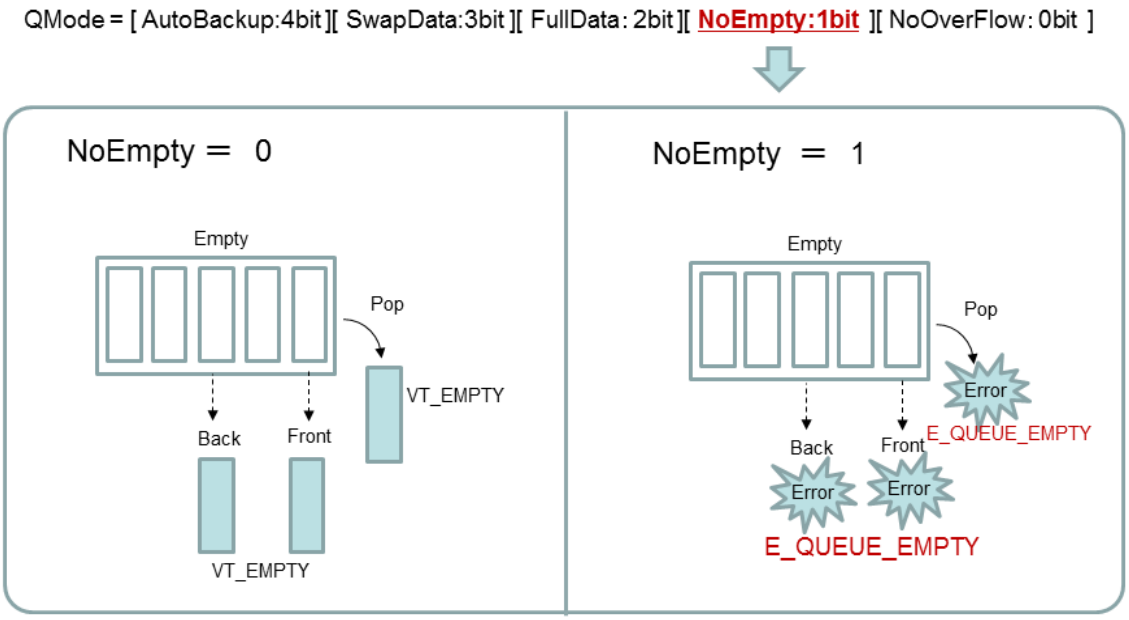
2.3.2.1.1. NoOverFlow bit

The specification of the NoOverFlow bit is specification whether assume the error in case of full when data is added to the cue. Data forward is annulled when adding it, and specified data is added backward when not assuming the error.



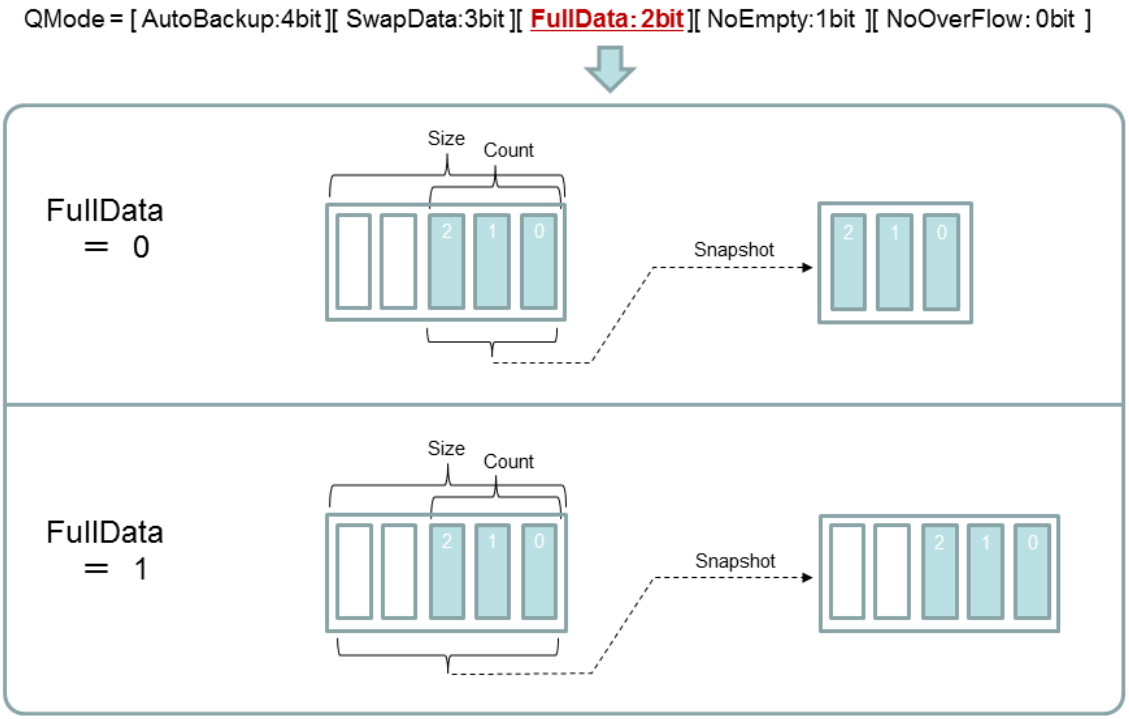
2.3.2.1.2. NoEmpty bit

The specification of the NoEmpty bit is specification whether assume the error when the cue is empty.



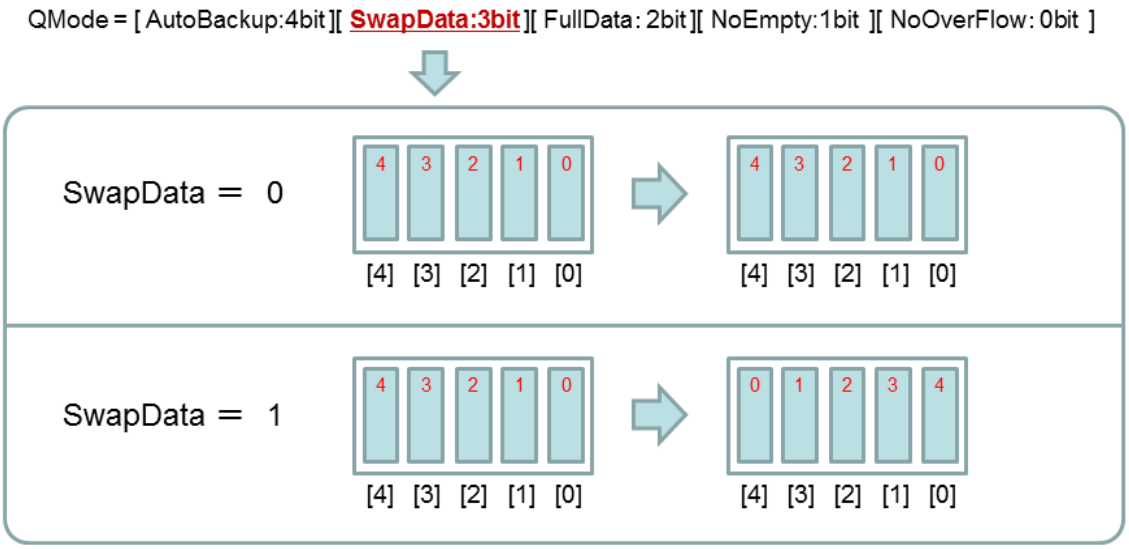
2.3.2.1.3. FullData bit

The specification of the FullData bit is specification whether to include the free space when all data stored in the cue is lumped together and acquired (Snapshot operation).



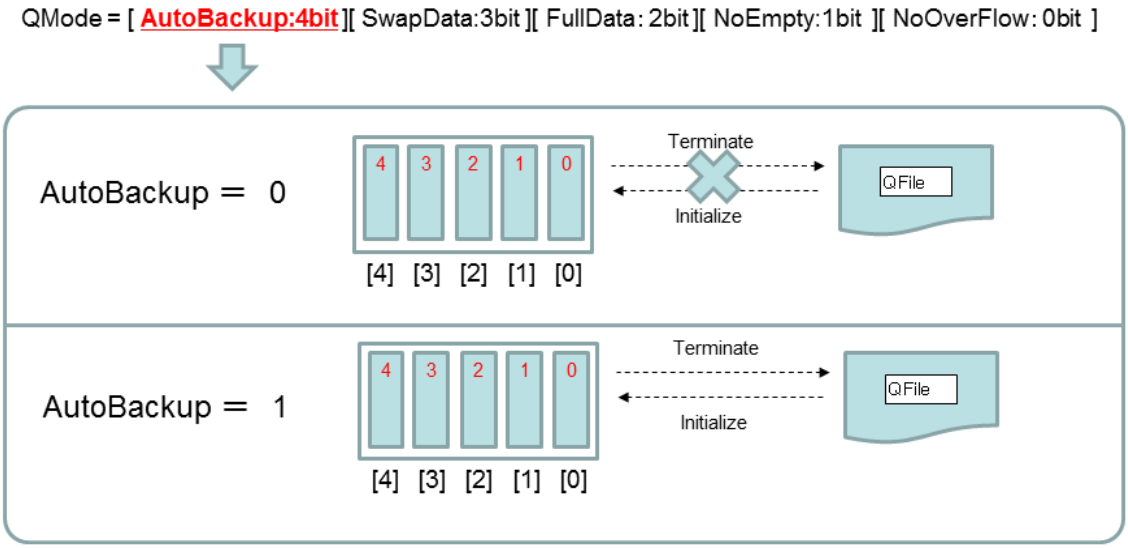
2.3.2.1.4. SwapData bit

The specification of the SwapData bit is specification whether to reverse the arrangement of the data string returned when all data stored in the cue is lumped together and acquired (Snapshot operation).



2.3.2.1.5. AutoBackup bit

The specification of the AutoBackup bit is specification whether to execute backup/restoring of data at initialization and the liberating time of Extension.



Precautions

If multiple DataQueue providers are used at the same time and the backup file specifications (QDir, QFile) point to the same one, the last saved one will be overwritten. At the time of restoration, only the queue for one DataQueue provider is restored.

CaoExtension::AddVariable method

It is possible to operate it to the cue by reading and writing the variable in the CaoExtension object to the managed cue.

The argument specification of AddVariable is shown as follows.

Format AddVariable (<bstrName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName > : In variable identifier

The following system variable identifier or user variable identifiers can be specified.

System variable : The following reservation character strings that start by '@' are specified.

@QMODE
@QSIZE,
@QCOUNT
@QCLEAR
@QSNAP
@QPUSH
@QPOP
@QBACK
@QFRONT

User variable : An arbitrary character string that doesn't start by '@' is specified. In this case, it is indispensable to specify Type of an optional character string.

<bstrOption> : In optional character string

The following options can be specified by switching off the comma district.

Type : For the user variable to decide behavior, the system variable identifier is specified.

@QMODE
@QSIZE,
@QCOUNT
@QCLEAR
@QSNAP

	@QPUSH @QPOP @QBACK @QFRONT This option is ineffectual for the system variable. Example) Type=@QCOUNT

The example when the AddVariable method is executed is shown as follows.

```
AddVaraible
(
  "@QCOUNT",                        // System variable
  ""                          //
);

AddVaraible
(
  "MyQueueCount",           // User variable
  "Type=@QCOUNT"                    // @QCOUNT of system variable
);
```

The system variable mounted in the DataQueue provider. Please refer to 2.4.1.

2.3.3. CaoExtension::Execute method

The command of the CaoExtension object is executed.

The list of the command of the CaoExtension::Execute method is shown as follows.

Table2-2List of mounting command of CaoExtension::Execute method

Command	Function	Correspondence variable
GetQueueMode	The operational mode of the cue is acquired.	@QMODE
SetQueueMode	The operational mode of the cue is set.	@QMODE
GetQueueSize	The number of maximum data that can be stored in the cue is acquired.	@QSIZE
SetQueueSize	The number of maximum data that can be stored in the cue is set.	@QSIZE
GetQueueCount	The number of data stored in the cue is acquired.	@QCOUNT
ClearQueue	The cue is emptied.	@QCLEAR
GetQueueSnapshot	The batch acquires the data stored in the cue.	@QSNAP
PushQueue	Data is added at the end of the queue of the cue.	@QPUSH
PopQueue	Data that exists in the queue of the cue forward is acquired.	@QPOP
GetQueueBack	Refer to rear data that exists in the queue of the cue.	@QBACK
GetQueueFront	Refer to data that exists in the queue of the cue forward.	@QFRONT

2.3.3.1. CaoExtension::Execute("GetQueueMode") command

The operational mode of the cue is acquired.

Format GetQueueMode ()

Return value : [out] operational mode (VT_I4)

Please refer to 2.3.3.1 for the value of the operational mode.

2.3.3.2. CaoExtension::Execute("SetQueueMode") command

The operational mode of the cue is set.

Format SetQueueMode ([<Mode>])

< Mode >	:	[in] operational mode (VT_I4) Range of value:Arbitrary integral value The specified value is converted into the value in which the mask is done only by an effective bit. Please refer to 2.3.2.1 for the value of the operational mode.
Return value	:	[out] none

2.3.3.3. CaoExtension::Execute("GetQueueSize") command

The number of maximum data that can be stored in the cue is acquired.

Format GetQueueSize ()

Return value	:	Number of maximum data in which [out] can be stored (VT_I4)

2.3.3.4. CaoExtension::Execute("SetQueueSize") command

The number of maximum data that can be stored in the cue is set.

Format SetQueueSize ([<Size>])

< Size >	:	[in] Number of maximum data in which in can be stored (VT_I4) Range of value: 1 - 2147483647 When the stored number of data is several where specification can be stored, data forward is annulled existing.
Return value	:	[out] none

2.3.3.5. CaoExtension::Execute("GetQueueCount") command

The number of data stored in the cue is acquired.

Format GetQueueCount ()

Return value : [out] Number of [out] data (VT_I4)

2.3.3.6. CaoExtension::Execute("ClearQueue") command

The cue is emptied.

Format ClearQueue ()

Return value : [out] none

2.3.3.7. CaoExtension::Execute("GetQueueSnapshot") command

The batch acquires the data stored in the cue.
The data that can be acquired becomes only data progressed on the memory.

Format GetQueueSnapshot ()

Return value : [out] Data progressed on [out] memory
(VT_VARIANT|VT_ARRAY)

2.3.3.8. CaoExtension::Execute("PushQueue") command

Data is added at the end of the queue of the cue.

Format PushQueue ([<Data>])

< Data > : [in] additional data (VT_VARIANT)
The NoOverFlow bit of the QMode operational mode : by one.
Data forward is annulled for the maximum number that can be stored, and data is added to the tail.

Return value : [out] none

2.3.3.9. CaoExtension::Execute("PopQueue") command

Data that exists in the queue of the cue forward is acquired.

Format PopQueue ()

Return value : [out] data forward (VT_VARIANT)

2.3.3.10. CaoExtension::Execute("GetQueueBack") command

Refer to rear data that exists in the queue of the cue.

Format GetQueueBack ()

Return value : [out] rear data (VT_VARIANT)

2.3.3.11. CaoExtension::Execute("GetQueueFront") command

Refer to data that exists in the queue of the cue forward.

Format GetQueueFront ()

Return value : [out] data forward (VT_VARIANT)

2.3.4. CaoExtension::get_VariableNames property

In the get_VariableNames property, possible specification acquires the list of the system variable in the VARIANT type that stores the array of the character string type in AddVariable.

2.4. Variable list

2.4.1. CaoExtension class

In the DataQueue provider, only the system variable has been reserved. An arbitrary name can be used for the user variable.

Table2-3CaoExtension class system variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
@QMODE	VT_I4	The operational mode of the cue acquisition/is set. Range of value:Arbitrary integral value The specified value is converted into the value in which the mask is done only by an effective bit.	✓	✓
@QSIZE	VT_I4	The number of maximum data that can be stored in the cue acquisition/is set. Range of value: 2147483647 from one When the stored number of data is several where specification can be stored, data forward is annulled it is time when it sets it and existing.	✓	✓
@QCOUNT	VT_I4	The number of data stored in the cue is acquired.	✓	-
@QCLEAR	None	The cue is emptied.	-	✓
@QSNAP	VT_VARIANT| VT_ARRAY	The batch acquires all data maintained in the memory. However, maximum = 500 pieces when QFile option is specified. VT_EMPTY is restored when there is no data.	✓	-
@QPUSH	VT_VARIANT	Data is added at the end of the queue of the cue. Data forward is annulled for the maximum number where the NoOverFlow bit of the QMode operational mode can be stored by one, and data is added to the tail.	-	✓
@QPOP	VT_VARIANT	Data that exists in the queue of the cue forward is acquired. VT_EMPTY is restored when there is no data.	✓	-
@QBACK	VT_VARIANT	Refer to rear data that exists in the queue of the cue. VT_EMPTY is restored when there is no data.	✓	-

@QFRONT	VT_VARIANT	Refer to data that exists in the queue of the cue forward. VT_EMPTY is restored when there is no data.	✓	-

2.5. Error code

In the DataQueue provider, the following and peculiar the error code is defined. About the ORiN2 commonness error, Please refer to the chapter of the error code of "ORiN2 programming guide".

Table2-4Original error code list

Error name	Error number	Explanation
E_FAILED_CREATE	0x80100000	It failed in generation.
E_QUEUE_OVERFLOW	0x80100001	OVERFLOW was generated at PUSH.
E_QUEUE_EMPTY	0x80100002	The cue sometimes did the error generation of the sky (data none) to the data acquisition of POP, SNAPSHOT, BACK, and FRONT.

3. Sample program

Cue (Q1) is made as follows, the value is added on the other hand, and the sample taken [out] is shown.

List 3-1 SampleApp1.frm

```
Private Eng As CaoEngine
Private Ctrl As CaoController
Private Ext As CaoExtension
Private VarPush As CaoVariable
Private VarPop As CaoVariable

..'.. loading of form
Private Sub Form_Load()
  Set Eng = New CaoEngine

The variable object ..'.. is acquired.
  Set Ctrl = Eng.Workspaces(0).AddController("DQ", "CaoProv.DataQueue")
  Set Ext = Ctrl.AddExtension("Q1","QMode=1,QDir=C:/Temp,QFile=Queue")

  Set VarPush = Ext.AddVariable("@QPUSH")
  Set VarPop = Ext.AddVariable("@QPOP")

End Sub

It is 'PUSH in the cue as for the value.
Private Sub Command1_Click ()
  VarPush = Text1.Text
End Sub

It is 'Pop from the cue as for the value.
Private Sub Command2_Click ()
  Text2.Text = VarPop
End Sub
```