DataStore provider

General-purpose data sharing

Version 1.0.5

User's guide

October 31, 2018

【 Remarks 】

【 Revision history 】

Version	Date	Content
1.0.0.0	2006-02-23	First edition.
1.0.1.0	2006-12-15	Modified “2.2.4.DataBase preservation function” to “@VARS variable”.
1.0.2.0	2007-03-07	The fixed index function and the HELP writing function was added by the @VARS variable.
1.0.3.0	2008-02-06	Database option was added
1.0.4.0	2008-04-17	@LVARS_MAX variable was added
1.0.4.1	2010-02-10	Error code was added
1.0.4	2012-07-17	Document versioning rules was changed.
1.0.4	2018-05-21	Fix typo. ("PutMacro","GetMacro" → "PutVarsMacro","GetVarsMacro")
1.0.5	2018-10-31	Memory leak was corrected.

【 Hardware 】

Model	Version	Notes

Contents

1. Introduction

This book is a user’s guide of the DataStore provider that makes easier to share the data between ORiN2 applications.

The DataStore provider achieves the data sharing between the ORiN2 applications by storing the variable table internally, and sharing the variable table. If this provider is used, not only the data sharing between the applications in the same machine but also the data sharing with the application that is the operation with another machine can be easily achieved.

This book explains the function of this DataStore provider and the mounting method.

2. Outline of provider

2.1. Outline

CAO(Controller Access Object) of ORiN2 is an object model by whom the robot controller is abstracted. Therefore resource objects can be various types, such as tasks, variables, program files, and robots, though, the DataStore provider is a particular application provider with only "Variable" resources. However, if the controller name is different, a different value can be retained in the same variable name because the variable data is managed by each controller internally. [1]

The storable data type is VARIANT type which enable to store various data types including binary data of the array and the image data, etc. Therefore, this provider can share easily between two or more applications even complex data.

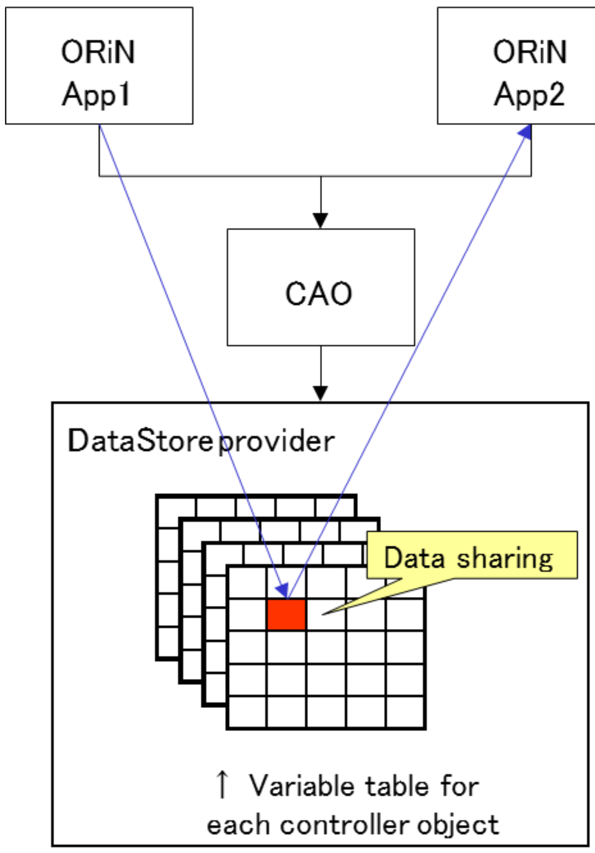

Figure 2-1 Outline

[1] This description is limited when executed by the same process space. In a different process space, the different value may be stored even in the same controller name and the same variable name.

Table 2-1 DataStore provider

File name	CaoProvData.dll
ProgID	CaoProv.DataStore
Registry registration[2]	regsvr32 CaoProvData.dll
Remove registry registration	regsvr32 /u CaoProvData.dll

MDAC(Microsoft Data Access Components) 2.7 is necessary for the DataStore provider.

[2] It is not necessary manual registration/deregistration when installed by ORiN SDK.

2.2. Outline of operation

2.2.1. Stack function

When variable name "@STACK" is used, the buffer where each controller can store two or more value of a variables is prepared. The buffer at this time becomes a First-In Last-Out type.

The buffer content is retained until the controller is deleted.

2.2.2. Queue function

When variable name "@QUEUE" is used, the buffer where each controller can store two or more value of a variables is prepared. The buffer at this time becomes a First-In First-Out type.

The buffer content is retained until the controller is deleted.

2.2.3. Random acquisition

The value when the value is acquired changes at random if variable name "@RANDOM" is used.

2.2.4. @VARS variable

Two or more values can be retained by changing the ID property when using the variable name "@VARS[<Fixed index number>]". < Fixed index number > is omittable, and cannot be changed dynamically by ID property when it is specified.

The controller name doesn't influence the value of "@VARS" variable. Therefore, the value can be shared as all controllers.

In the storage location of the value, there are a data base and a temporary variable. Only when the value is stored in the data base area, the content of data is preserved even if the DataStore provider is ended.

The size of the @VARS"variable area and the data base area can be changed by specifying the following character strings for Parameter of the DataStore provider by means of CaoConfig.

Table 2-2 Parameter character string of CaoConfig

Parameter	Meaning
DataBase=< Path of database file >	Specify the absolute path of the data base file. (Default: Default database) The data base file must copy and use the following files. <DataStore_Root>¥Bin¥datastore_master_en.mdb
ItemMax=< "@VARS"variable size>	Specify the size of ID property that is available in "@VARS"variable. (Default: 200)
ItemDBMax=< size of database	Specify the size of the database area of"@VARS"variable

area >	(default: 100)

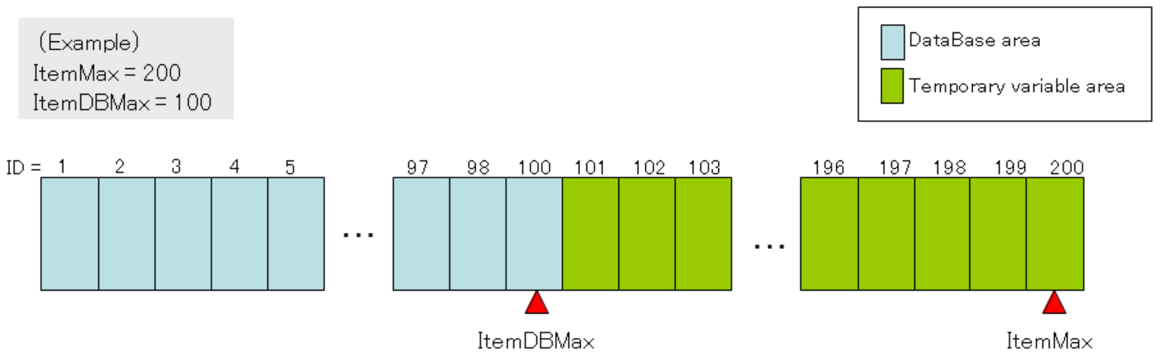

Figure 2-2 Structure of VARS variable

2.2.5. @LVARS variable

Two or more values can be retained by changing the ID property when using the variable name "@LVARS[<Fixed index number>]". < Fixed index number > is omittable, and cannot be changed dynamically by ID property when it is specified.

The value of "@LVARS" variable can be shared only when the controller name is identical.

Because the value is created on the memory space, the value is not preserved. When the object of the identical controller name is deleted, the stored value is cleared.

The size of the @LVARS"variable area can be changed by specifying the following character strings for Parameter of the DataStore provider by means of CaoConfig.

Table 2-3 Parameter character string of CaoConfig

Parameter	Meaning
LItemMax=<Size of the "@LVARS"variable >	Specify the size of ID property that is available in "@LVARS"variable (default: 200)

The function comparison between the "@LVARS" variable and the "@VARS" variable is shown as follows.

Table 2-4 "@LVARS" Variable and "@VARS" variable function comparison

	@LVARS	@VARS
Scope	Only the identical name controller	All controllers
Reserved area	Memory area only	Memory area and database area
Storage of value	Not stored	Database area only
Delete timing of value	When all controllers of the identical name are deleted	When the DataStore provider ends

2.3. Method and Property

2.3.1. CaoWorkspace::AddController method

In DataStore provider, a virtual controller name is specified at AddController execution. As mentioned earlier, the variable table is managed in each controller. The variable table is shared when the specified controller name already exists[3], and the variable table is made newly when not existing oppositely. Moreover, please note that the capital letter and the small letter are distinguished as for the controller name. When the controller object is deleted (Disconnect), the variable table is deleted. The once deleted data is not be restorable.

The argument specification of AddController is shown as follows.

```
AddController
(
    "< controller name >"        // Virtual controller name
    "CaoProv.DataStore"          // Provider name. Fixed.
    "< machine name >"           // Execution machine name of provider4
    "< option >"                 // Option character string
)
```

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

```
AddController
(
    "App1",                      // virtual controller name =  App1
    "CaoProv. DataStore",
    "",                          // executes by CAO engine process.
    ""
);
```

(Naming convention of virtual controller name)

There is a possibility of competing when each ORiN application names the virtual controller name arbitrary. To avoid that problem, we recommend to follow the rule below.

[3] The collection function of CAO prohibits specifying the identical controller name in the same work space. Please use another workspace when using the same controller name.

[4] As mentioned earlier, if the <machine name> is not correct in each applications, the variable table cannot be shared even though specifying the same <controller name> because the DataStore provider is loaded to the different process space.

(1) < Application name > is used. < Application name > here indicates the one that all the file names except the extension of an executable file written by the capital letter. For instance, the application name of “CaoSQL.exe" becomes “CAOSQL".

(2) For the application which creates two or more instances, "< Application name> < Sequential number >" is used. < Sequential number > is recommended to begin from one. For example, “CAOSQL.1", “CAOSQL.2” ... etc.

2.3.2. CaoController::AddVariable method

The AddVariable method of this CaoController class is a method for the access to the variable in a general meaning of CAO. In the DataStore provider, the variable is registered in the variable table by this method. The specified variable is registered in the management table of the CaoController object that offers this method. The variable is shared when the specified variable name already exists, and the variable is created newly when not existing. Moreover, please note that the capital letter and the small letter are distinguished as well as the controller name as for the variable name. The value of a variable is VARIANT type which is capable of storing not only simple variable but also array data, etc.

The argument specification of AddVariable is shown as follows.

```
AddVariable
(
    ″< variable name >″   // Variable name registered in variable table.
    ″< option >″                   // (unused)
)
```

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

```
AddVariable
(
    “ABC″,                // Specify ABC variable.
    “″
)
```

Because the DataStore provider is originally assuming the use to share data, a source application of data sharing must open the variable list to specify the stored data.

Moreover, both the user variable and the system variable can be specified for the variable name. When the system variable is specified, put “@" initial letter of the variable name. The variable name which does not have “@” in initial is treated as a user variable.

Please refer to 2.4.1 for the system variable mounted in the DataStore provider.

2.3.3. CaoController::Execute method

This method executes command. Please refer to Table 2-5 for the parameter necessary for the execution of the command and the result of acquisition.

Table 2-5 List of mounting command of CaoController::Execute method

Command	Parameter	Return value	Operation
PutHelp <Index>, <Help>	<Index:VT_I4>= Index number of @VARS variable <Help:VT_BSTR>= Help character string of @VARS variable	None	Set the help character string of the @VARS variable. Acquisition uses the Help property of the variable class.
PutVarsMacro <Index>, <Macro>	<Index:VT_I4>= Index number of @VARS variable <Macro:VT_BSTR>= Macro name of @VARS variable	None	Set the macro name of the @VARS variable. Numerical value and "¥# $: ." cannot be used for the macro name.
GetVatsMacro <Index>	<Index:VT_I4>= Index number of @VARS variable	<Macro:VT_BSTR>= Macro name of @VARS variable	Acquire the macro name of @VARS variable.

2.3.4. CaoController::get_VariableNames property

The list of the user variable added with AddVariable is acquired in the VARIANT type that stores the array of the character string type in the get_VariableNames property.

2.3.5. CaoVariable::put_ID property

This property is available only at variable name "@VARS". When this property is used at other variables, the error is returned.

Specify the ID of the data which is accessed by the database preservation function. Minimum value is 1 and the maximum value is the value of ItemMax option of AddController. When a value outside this range is specified, the error is returned.

2.3.6. CaoVariable::get_ID property

This property is available only at variable name "@VARS". When this property is used at other variables, the error is returned.

ID of the data currently being accessed by the database preservation function is acquired.

2.3.7. CaoVariable::get_Help property

This property is available only at variable name "@VARS". When this property is used at other variables, the error is returned.

Explanatory information is acquired from the data currently being accessed by the database preservation function.

2.3.8. CaoVariable::get_Value property

The value registered in the variable table is acquired in the VARIANT type. The value of the content of 2.4.1 is acquired for the system variable. The value set with 2.3.9 is acquired for the user variable.

2.3.9. CaoVariable::put_Value property

This property enables to register the value in the variable table by the VARIANT type. For the system variable, put_Value property is available only when "Put" column of 2.4.1 is checked (”√”).

2.3.10. CaoVariable::get_DateTime property

The date when the value was updated is returned. This property initialize to current time at the time of startup. This property is available only when the name of variable is “@VARS”. When this property is used at other variables, the error is returned.

2.3.11. CaoVariable::get_Microsecond property

The element at the micro second of the date to which the value is updated is returned. This property initialize to current time at the time of startup. This property is available only when the name of variable is “@VARS". When this property is used at other variables, the error is returned

2.4. Variable list

2.4.1. Controller class

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

Table 2-6 Controller class system variable list

Variable name	Data type	Explanation	Attribute	
			get	put
@COUNT	VT_I4	Return the number of user variables. The system variable is not included.	√	-
@CURRENT_TIME	VT_DATE	Local time of machine from which provider is executed.	√	-

Variable name	Data type	Explanation	Attribute	
			get	put
@QUEUE	VT_VARIANT	Queue type variable. One in one controller object. When emptying, VT_EMPTY is returned. The buffer is cleared and empties when VT_EMPTY is written.	√	√
@QUEUE_SIZE	VT_I4	A current number of elements of queue type variables (@U_QUEUE) is returned.	√	-
@RANDOM	VT_I4	Create random numbers.	√	-
@STACK	VT_VARIANT	Stack type variable. One in one controller object. When emptying, VT_EMPTY is returned. The buffer is cleared and empties when VT_EMPTY is written.	√	√
@STACK_SIZE	VT_I4	Return a current number of elements of stack type variables (@U_STACK) .	√	-
@STRESS	VT_I4	Create CPU load. The stress level is set by writing the value. Whenever reading, CPU load is created according to the stress level.	√	√
@TICK_COUNT	VT_I4	Return the elapsed time after system starting up by each millisecond (ms). GetTickCount function.	√	-
@VERSION	VT_BSTR	Version.	√	-
@VARS[.<Fixed index>]	VT_VARIANT	Common variable. < fixed index > is omittable. The index number or the macro name can be specified for < fixed index >. Acquire/set the value for the data which is specified in Variable : ID property. When < fixed index > is specified, the ID property cannot be set. When data is not set when acquiring, VT_EMPTY is acquired.	√	√
@VARS_MAX	VT_I4	Acquire the maximum value of ID that can be specified by the database access variable (@VARS)	√	-

Variable name	Data type	Explanation	Attribute	
			get	put
@LVARS[.<Fixed indexr>]	VT_VARIANT	Each controller's common variable. < fixed index number > is omittable Acquire/set the value for the data which is specified in Variable : ID property. When < fixed index number > is specified, the ID property cannot be set. When data is not set when acquiring, VT_EMPTY is acquired.	√	√
@LVARS_MAX	VT_I4	Acquire the maximum value of ID that can be specified with @LVARS. This value is correspond with the value of the LItemMax parameter.	√	-

2.5. Error code

In the DataStore provider, the peculiar error code is defined as follows. Please refer to the chapter of the error code of "ORiN2 Programming guide" for the ORiN2 commonness error.

Table 2-7 Peculiar error code list

Error name	Error number	Explanation
E_FAILED_CREATE	0x80100000	Creation failure.
E_DOUBLE_DEFINED	0x80100001	Already defined.

3. Sample program

The following code shows an example of how to set the variable value by App1.frm(1exe), and acquire the value by App2.frm (another exe)

List 3-1 SampleApp1.frm

```
Private Eng As CaoEngine
Private Ctrl As CaoController
Private Var As CaoVariable

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

    'Acquire the variable object.
    Set Ctrl = Eng.Workspaces(0).AddController("RC1", "CaoProv.DataStore")
    Set Var = Ctrl.AddVariable("Sample")

End Sub

'Set the value to the variable of Ctrl.
Private Sub Command1_Click ()
    Var = Text1.Text
End Sub
```

List 3-2 SampleApp2.frm

```
Private Eng As CaoEngine
Private Ctrl As CaoController
Private Var As CaoVariable

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

    'Acquire the variable object.
    Set Ctrl = Eng.Workspaces(0).AddController("RC1", "CaoProv.DataStore")
    Set Var = Ctrl.AddVariable("Sample")

End Sub

' The value is acquired from the variable of Ctrl2.
Private Sub Command2_Click ()
    Text2.Text = Var
End Sub
```