ORiN2

Programming guide

Version 1.0.22.0

September 2, 2024

[Remarks]

Some items might not be installed depending on the package you have.

[Revision history]

Date	Version	Content
2005-05-09	1.0.0.0	First edition.
2006-08-21	1.0.1.0	Add the way to register CAO RCW into GAC under .NET framework 2.0.
2007-04-23	1.0.2.0	ORiN Installer registers CAO and CaoSQL RCW into GAC.
2008-07-16	1.0.3.0	Add some descriptions
2009-06-12	1.0.4.0	Add Import and Export commands to save settings.
2009-07-31	1.0.5.0	Add the way to register CAO RCW into GAC under .NET framework 2.0.
2010-02-10	1.0.6.0	Add description on LabVIEW
2010-06-08	1.0.7.0	Add the way to confirm an installation state of ORiN2 SDK.
2011-04-27	1.0.8.0	Add a sample code in C#.
2011-12-22	1.0.9.0	Add an error code.
2012-07-10	1.0.10.0	Add an error code.
2012-08-24	1.0.11.0	Modify an error code
2012-09-07	1.0.12.0	Modify an error code
2014-01-20	1.0.13.0	Add an error code. Add an integration into Visual Studio
2014-09-22	1.0.14.0	Add an error code.
2015-11-04	1.0.15.0	Add SysLog output of CAO log.
2016-12-12	1.0.16.0	Correction of errors.
2017-03-03	1.0.17.0	Add an error code.
2017-09-21	1.0.18.0	Add client creation using Java-COM bridge
2019-06-18	1.0.19.0	Add an error code.
2021-12-13		Fixes to the CAO call diagram in Java
2022-06-24	1.0.20.0	Add C# programming using ManagedCAO
2022-11-22	1.0.21.0	Correction of errors.
2024-09-02	1.0.22.0	Correction of errors.

Content

1. Introduction

ORiN is a middleware that offers a standard interface of various resources, like various Factory Automation (FA) equipment and databases, etc. including robots. By using ORiN, applications can be developed without depending on the manufacturer or the model type.

ORiN2 is composed of the following three basic technologies shown in Figure 1-1.

(1) CAO(Controller Access Object)

(2) CRD(Controller Resource Definition)

(3) CAP(Controller Access Protocol)

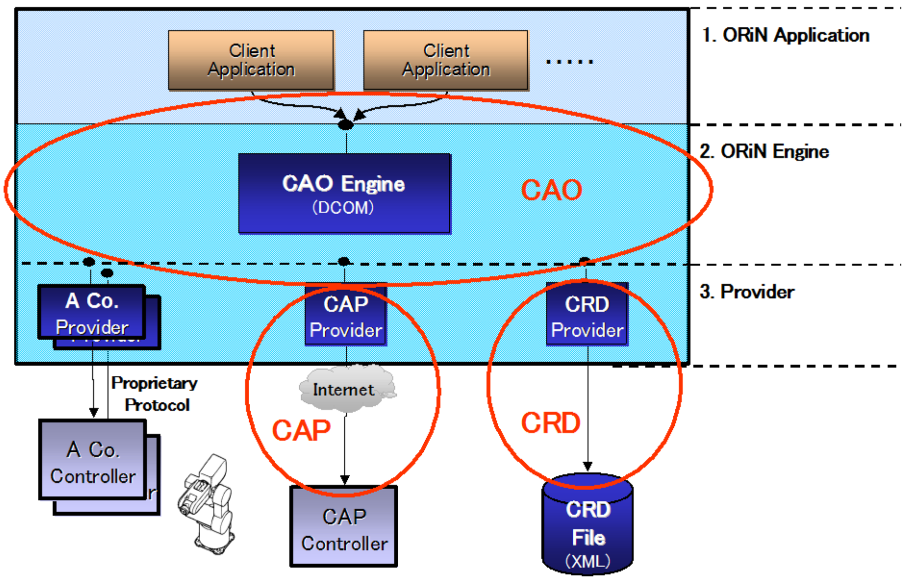


Figure 1-1 Three basic techniques of ORiN2

CAO is "Standard program interface" to access robots controller from client applications. CAO is developed based on the distributed object technology, and it is applied not only to the industrial robots but also PLC (Programmable Logic Controller) and the NC machine tool, etc. CAO has expanded the range of its application.

CRD is "Standard Data Schema" to share robot controller's resource information without depending on the robot manufacture, and uses XML files. CAO is a common API, and CRD is a common data expression. CRD is developed based on the XML technology, and can express various types of data, which differ in each robot manufacturer, by single CRD schema. Just like CAO, application field of the CRD basic data schema is not limited to the robot, but "Production information" etc. also can be expressed.

CAP is "Communication protocol for the Internet" to access CAO provider over the Internet. CAP is developed based on the SOAP (Simple Object Access Protocol) technology, and offers the function to access remote controllers without forcing ORiN application developer considering the Internet.

This book is a programming guide for these three basic technologies, CAO/CRD/CAP. Chapter 2 describes about the procedure for "Implementation of the CAO client". Chapter 3 is about the procedure to "Make of CRD file". Chapter 4 explains procedure for "Remote connection with CAP". Intended reader of each chapter is shown in Table 1-1. Each chapter explains necessary basic knowledge for the chapter in the first half, and then explains the implementation method in the latter half with concrete examples.

Table 1-1 Intended reader of this book

Chapter	Content	Intended Reader
Chapter 2 (p.8)	Implementation of CAO client	ORiN application developer
Chapter 3 (p.58)	Introduction to CRD implementation	Robot vender ORiN application developer
Chapter 4 (p. 68)	Remote connection by CAP	ORiN application developer

Chapter 5 explains the method of setting DCOM to use the CAO provider remotely. Chapter 6 explains how to use ORiN2 SDK tools.

2. Implementation of CAO client

2.1. Outline

CAO is developed based on the distributed object technology as described in the previous chapter. In ORiN2 SDK, DCOM (Distributed Component Object Model) of Microsoft Corporation is adopted as a distributed object technology. The DCOM based CAO can be used from various program languages such as C++, JAVA, and Visual Basic.

This chapter uses and explains Visual Basic6.0 (VB6), because the language is best for the introduction purpose. The first half of this chapter explains the object generation method as basic knowledge of VB6. The latter half of this chapter explains examples of variable object processing and event handling. The explanation uses the Blackboard provider, which is a standard provider included in ORiN2 SDK, and also includes example codes.

The Blackboard provider, which is used in the sample, offers functions to share the variable table between client applications, and it only has variable object. However, because the provider has functions of the event and the system variable, etc., readers can experience most of the general functions of the CAO provider.

The object model of CAO is shown as follows.

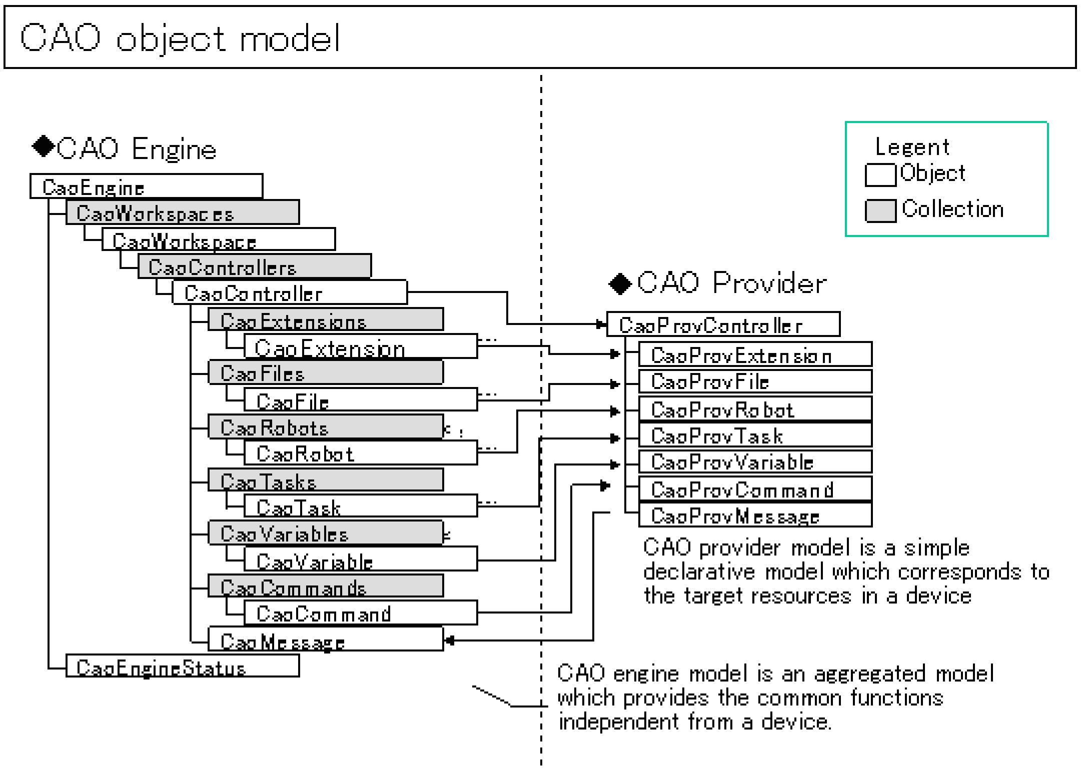


Figure2-1 Object model of CAO

2.2. Basic knowledge

2.2.1. Early Binding and Late Binding

CAO client has two types of object generation method, i.e. early binding and late binding[1]. Following explains the difference of these two methods, and how to generate objects.

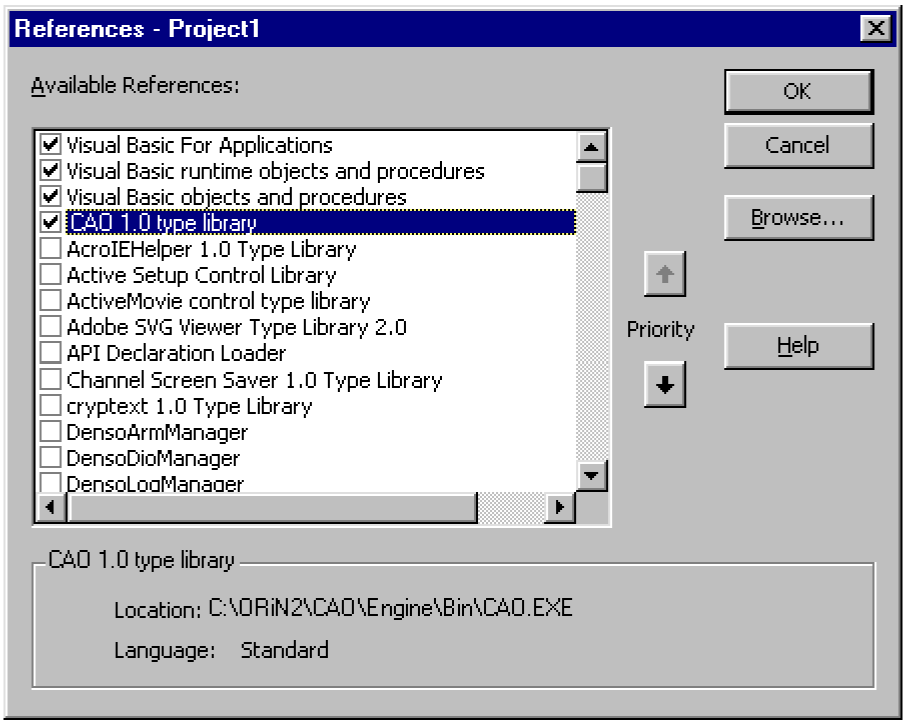
2.2.1.1. Early Binding

Early binding is a method of acquiring type information of the object at compiling time by referring to the type library. Client will keep type information and the information inquiry from client to the component is not necessary. As a result, the processing speed improves. However, as weak point, clients need to be recompiled if component with object information are changed, and it reduces flexibility.

Following procedures is how to use Early Binding.

(4) Select [Project]->[reference setting] in the menu bar of VB6.

(5) When the dialog is displayed as shown in Figure2-2, add the type library "CAO 1.0 type library".


References - Project1

Available References:

- [✓] Visual Basic For Applications
- [✓] Visual Basic runtime objects and procedures
- [✓] Visual Basic objects and procedures
- [✓] CAO 1.0 type library
- [ ] AcroIEHelper 1.0 Type Library
- [ ] Active Setup Control Library
- [ ] ActiveMovie control type library
- [ ] Adobe SVG Viewer Type Library 2.0
- [ ] API Declaration Loader
- [ ] Channel Screen Saver 1.0 Type Library
- [ ] cryptext 1.0 Type Library
- [ ] DensoArmManager
- [ ] DensoDioManager
- [ ] DensoLogManager

CAO 1.0 type library

Location: C:\ORiN2\CAO\Engine\Bin\CAO.EXE

Language: Standard

Figure2-2 Reference setting screen of VB

[1] [1]It uses and it explains the early binding in this book.

(6) The CaoEngine object can be generated with Early Binding by writing the following codes with VB6.

```
'CaoEngine type variable is made.
Dim caoEng As CaoEngine
'An instance is created with New keyword, and substituted by Set statement.
Set caoEng = New CaoEngine
```

Because CaoEngine is an object, Set statement is necessary for substitution[2]. The CaoEngine type can be called when the variable is declared, because the reference setting is finished in reference setting ob VB environment.

2.2.1.2. Late Binding

Unlike early binding, late binding doesn't need the type library. There is no object type inspection in compilation time, and the object type is dynamically inspected at the execution time. Therefore, the processing speed is slower than early binding because all process including type inspection is executed at runtime. However, recompilation of application after the change of component is not necessary, because it is completely independent of the component, and it is excellent in flexibility.

In rate binding, an object is created by using CreateObject. The definition of CreateObject is shown below.

```
CreateObject(class)
```

This function returns a pointer to an object specified with class. At this time, class is composed of “application-name.object-name."

To create an object with rate binding, write the following codes.

```
' An Object type variable is declared.
Dim caoEng As Object
' A CaoEngine object created with CreateObject() is substituted into caoEng.
Set caoEng = CreateObject(“CAO.CaoEngine” )
```

In this example, object type variable is created first. Then CaoEngine object of CAO.exe is created with CreateObject.

[2] In VB.NET, the Set statement is not used.

2.2.2. Creation and management of object

For the creation of COM object, New keyword or CreateObject function is used. The created object is substituted into a variable by using Set statement. These created object need to be deleted before the program ends. To delete the object, substitute Nothing to the variable[3].

For instance, following is a sample code of CaoEngine object to be created and deleted.

```
' CaoEngine type variable is declared.
Dim caoEng As CaoEngine
' Creation of object
Set caoEng = New CaoEngine
' Deletion of object
If Not caoEng Is Nothing Then
       Set caoEng = Nothing
End If
```

CAO client manages objects in the following way.

By using ORiN2 SDK, the objects shown in table Appendix A.1 CAO engine function list can be created. A sample program in this document uses a Blackboard provider, and it uses the following objects.

- CaoEngine
- CaoWorkspace
- CaoController
- CaoVariable

These object need to be created sequentially. Below is the order of creation.

(1) The CaoEngine object is created with New or CreateObject().
(2) The CaoWorkspace object is acquired from the CaoEngine object. (default workspace)
(3) The CaoController object is created from the CaoWorkspace object.
(4) The CaoVariable object is created from the CaoController object.

[3] It is necessary to call the function to open the object specifying it in VB.NET though the object can be annulled by substituting Nothing in VB6. Please refer to 2.5.3 f or details.

Section 2.3 describes about a concrete object generation method. The creation order is shown in Figure2-3.

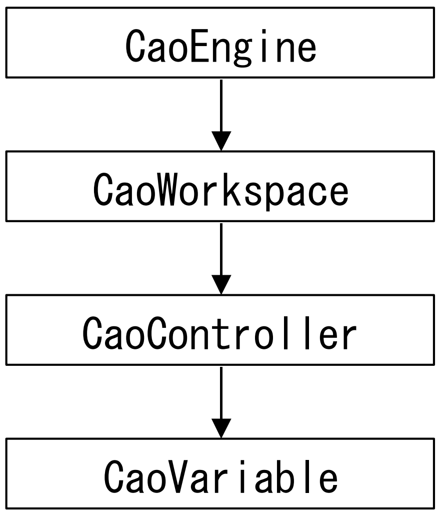


Figure2-3 Creation and the acquisition order of object

As mentioned previously, when the CAO client ends, it is necessary to delete all created objects. If the object is not normally deleted, CAO.exe[4] might not be terminated even after the CAO client ends. Although the CAO engine can be deleted only by substituting Nothing, other objects maintained by object collection, including CAO workspace, need to be deleted in the reverse order of creation and acquisition. Each object of CAO is created by Add… method. When objects are created, they are automatically registered in the collection. If an object is registered in a collection, they aren't deleted even if a client deletes the object. Remove method is necessary to remove a object from a collection.

Following is an example. In this example, the program deletes objects in order of CaoVariable, CaoController, CaoWorkspace, and the CaoEngine object.

```
'caoVar is deleted from the Variable collection of caoCtrl.
If Not caoVar Is Nothing Then
       caoCtrl.Variables.remove caoVar.Index
       Set caoVar = Nothing
End If
'caoCtrl is deleted from the Controller collection of 'caoWS.
If Not caoCtrl Is Nothing Then
       caoWS.Controllers.remove caoCtrl.Index
       Set caoCtrl = Nothing
End If
'caoWS is deleted from the Workspace collection of 'caoEng.
If Not caoWS Is Nothing Then caoEng.Workspaces.Remove (caoWS.Index)
       Set caoWS = Nothing
End If
```

[4] CAO.exe is a file of the execute form of the CAO engine. When the CaoEngine object is generated in the client application, it starts automatically.

```
        If Not caoEng Is Nothing Then
              Set caoEng = Nothing
        End If
```

2.2.3. Asynchronous processing

Asynchronous processing that uses the event is supported in CAO. For instance, in the Blackboard provider which we will use as a sample, the event is generated when a variable is added or the value of variable is changed, and asynchronous processing can be achieved.

Different from synchronous processing, asynchronous processing does not return result by return value of property. When an event occurs, the event procedure of the variable that maintains an object is called. Therefore, for asynchronously processing, the client should implement process of receiving the result in the event procedure.

Following is the procedure to use asynchronous processing of CAO.[5]

(1) Add“WithEvents" to the declaration of the CaoController object. Following is an example of CaoController declaration with event procedure.

```
Private WithEvents caoCtrl As CaoController
```

(2) Add process to the event procedure. The event procedure name is the controller name followed by “_OnMessage". Following is an example of implementing the event procedure.

```
 Private Sub caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)
'Describe the content of processing here
 End Sub
```

Please refer to user's guide of each provider for details of the member variable implemented on CAOLib.ICaoMessage that is the argument of the event procedure. In the BlackBoard provider, which is used for the example, the name of the event type, the changing variable name and value, etc. are implemented.

2.2.4. Variable type used with COM

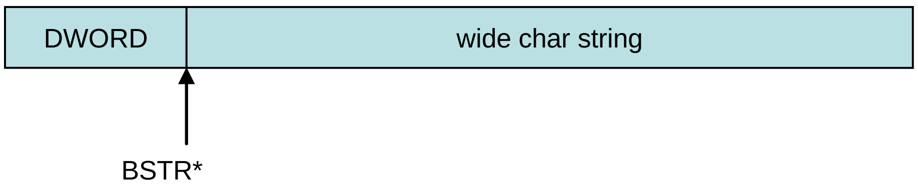
Four variable types, BSTR, VARIANT, SAFEARRAY, and HRESULT are characteristic variable types used in COM, and they are also often used in provider implementation.

2.2.4.1. BSTR

BSTR is a data type composed of a wide character string and string length in the DWORD type. The pointer of the BSTR type indicates the head of the character string as shown in Figure2-4.

Figure2-4 Structure of BSTR

[5] It is necessary to set the event procedure by the AddHandler method in VB.NET. Please refer to 2.5.4 for details.



To substitute value into BSTR, use SysAllocString() to secures the area of the character string. After the string is used, the area need to be released by SysFreeString(). Character string conversion macro A2BSTR also can be used. Following is an example of substituting character string "This is test." into BSTR.

```
// The BSTR type is declared.
BSTR bstrSample;
// Character string "This is test." is substituted.
// Because it is a wide character string, the character string is enclosed with L" ".
bstrSample = SysAllocString( L"This is test." );
// The area is released after it is used.
SysFreeString( bstrSample );
// Using character string conversion macro is also possible for substitution.
// (The memory is allocated in the macro.)
bstrSample = A2BSTR( "This is test." );
// The area is released after it is used.
SysFreeString( bstrSample );
```

BSTR also has a wrapper class named CComBSTR or _bstr_t. In these classes, operations like string substitution by "=" calls SysAllocString(), and destructor executes SysFreeString(). As a result, BSTR can be used without considering allocation or release of the area of the character string. Other overloaded operators are also prepared. For example, string addition is easily achieved by using operator "+=".

Following is an example of substituting character string "THIS IS TEST." into BSTR.

```
// The BSTR type is declared.
BSTR bstrVal;
// CComBSTR is declared by 16 characters.
CComBSTR str(16);
// The character string is substituted by "=".
str = L"This ";
// The character string is added by the Append method.
str.Append(L"is ");
// The character string is added by "+=".
str += L"test.";
// ToUpper method converts character string into the capital letters.
str.ToUpper();
// CComBSTR type is copied to the BSTR type.
bstrVal = str.Copy();
// CComBSTR is released.
delete str;
```

2.2.4.2. VARIANT

VARIANT is a structure that can treat various data types. The variable is composed of VARTYPE type variable VT to express data type, and a union of stored variable.

Following shows the details of a basic data type.

Table 2-1 Union of VARIANT type (part)

Type	Identifier	Member name	Explanation
BYTE	VT_UI1	bVal	Character(unsigned)
SHORT	VT_I2	iVal	short(signed)
LONG	VT_I4	lVal	long(signed)
FLOAT	VT_R4	fltVal	float(signed)
DOUBLE	VT_R8	dblVal	double(signed)
VARIANT_BOOL[6]	VT_BOOL	boolVal	Logical(passed as signed short)
SCODE	VT_ERROR	scode	Status code(HRESULT)
DATE	VT_DATE	date	Date(passed as double.)
BSTR	VT_BSTR	bstrVal	Character string type
IUnknown*	VT_UNKNOWN	punkVal	Interface pointer
IDispatch*	VT_DISPATCH	pdispVal	IDispatch interface pointer
SAFEARRAY*	VT_ARRAY	parray	Array type
	VT_EMPTY		no value

Among of them, VT_EMPTY shows that there is no value.

Using VARIANT type, these values also can be used for call by reference. In this case, logical AND of the identifier and VT_BYREF is used to show it is used for call by reference.

To substitute value to VARANT type, substitute identifier into vt at first, and then substitute value into union. A macro can be used for this substitution. Following is an example of substituting value 1000 of the long type into VARIANT.

```
// The VRIANTDATE type is declared, and initialized.
VARIANT varSample;
VariantInit( &varSample );
// The data type is set to long.
varSample.vt = VT_I4;
// The value of 1000 is substituted.
varSample.lVal = 1000;
```

Next is an example of substituting value 1000 of the long type into VARIANT using a macro.

```
// The VRIANT type is declared, and initialized.
VARIANT varSample;
VariantInit( &varSample );
// The data type is set to long.
V_VT( &varSample ) = VT_I4;
// The value of 1000 is substituted.
V_I4( &varSample ) = 1000;
```

As shown in above examples, newly declared VARIANT need to be initialized before using it. If VariantInit() is used for initialization, the identifier becomes VT_EMPTY to show that the value is not put in VARIANT.

[6] To substitute to VT_BOOL type, instead of TRUE, FALSE of BOOL type, VARIANT_TRUE and VARIANT_FALSE need to be used

BSTR type or SAFEARRAY type, which we will discuss later, allocated area need to be released using function VariantClear() before VARIANT is released. The function judges the necessity of releasing resources by the type of identifier. Following is an example of substituting value “This is test." of the BSTR type into VARIANT.

```
// The VRIANTDATE type is declared, and initialized.
VARIANT varSample;
VariantClear( &varSample );
// The data type is set to BSTR.
varSample.vt = VT_BSTR;
// The value,This is test., is substituted.
varSample.bstrVal = SysAllocString(L"This is test.");
```

When substituting VARIANT type into another VARIANT type, VariantCopy() should be used, because memory may be allocated. The function newly allocates memory area for copy destination and transfers data into there.

VARIANT type has wrapper classes named CComVariant or _variant_t. These classes call VariantInit() in constructer, and call VariantClear() in destructor. By using the class, the developer can use VARIANT without considering initialization or memory release of VARIANT. These wrapper classes also prepares overloaded operators etc., and operator “=” can directly substitute the integer type or the character string, etc. without considering the type or the memory allocation. Following is an example of substituting value 2234 of the long type using CcomVariant.

```
// The variable of the CcomVariant type is declared in the short int type,
// and value 1234 is substituted.
CComVariant var(1234,VT_I2);
// Value 2234 is substituted by using “=”.
var = 2234;
// VARTYPE is changed from VT_I2 to VT_I4.
var.ChangeType(VT_I4);
```

2.2.4.3. SAFEARRAY

SAFEARRAY is an array type mainly used in automation, and is a structure with fields of the dimension, the number of elements, and a pointer to data. In COM, SAFEARRAY is used to transfer array data between processes. SAFEARRAY is a structure, but it cannot be directly accessed and need to be accessed using functions.

The following is necessary procedure to access SAFEARRAY.

(1) Prepare SAFEARRAY.

(2) Access SAFEARRAY.

(3) Close access to SAFEARRAY.

The procedure is explained in detail.

(1) Declare a structure of SAFEARRAYBOUND type and define the structure of created array.

Following is the definition of SAFEARRAYBOUND.

```
typedef struct tagSAFEARRAYBOUND
{
       ULONG cElements;
       LONG lLbound;
} SAFEARRAYBOUND;
```

cElements shows the number of array elements and lLbound shows the lower bound value of the index. For instance, following is the declaration to define an array of 10 elements with index number starting from 0.

```
SAFEARRAYBOUND bound = { 10, 0 };
```

SAFEARRAYBOUND is used when SafeArrayCreate() function is called. SafeArrayCreate() is a function to make an instance of SAFEARRAY. Following is the definition of SafeArrayCreate().

```
SAFEARRAY* SafeArrayCreate( VARTYPE vt, UINT cDims, SAFEARRAYBOUND rgsabound );
```

In the argument, vt is the type of array, cDim is the dimension of array, and rgsabound is an pointer to the start address of SAFEARRAYBOUND array. With these arguments, SafeArrayCreate() creates an instance and returns a pointer to created SAFEARRAY. Following is an example of creating SAFEARRAY of the VT_I2 type with SafeArrayCreate().

```
SAFEARRAY pSa = SafeArrayCreate( VT_I2, 1, &bound );
```

(2) To access the data area of SAFEARRAY, prepare an accessing pointer that corresponds to the SAFEARRAY type. If SAFEARRAY is created with VT_I4, the pointer of the long type is prepared. Next, call SafeArrayAccessData() to access SAFEARRAY. The definition of SafeArrayAccessData() is shown below.

```
HRESULT SafeArrayAccessData( SAFEARRAY * psa, void HUGEP ** ppvData );
```

In the definition, psa is a pointer to accessed SAFEARRAY, and ppvData is an access pointer to the array data. As a result, array data becomes accessible. For instance, following shows how to accesses SAFEARRAY*pSa with long*lData.

```
SafeArrayAccessData( pSa, (void**)&lData );
```

(3) When processing to SAFEARRAY ends, it is necessary to call SafeArrayUnaccessData() and to close the access to SAFEARRAY. The definition of SafeArrayUnaccessData() is shown below.

```
HRESULT SafeArrayUnaccessData( SAFEARRAY* psa );
```

This function specifies the pointer to SAFEARRAY for the argument, and closes the access to SAFEARRAY specified by this argument. Following is an example of closing the access to SAFEARRAY*pSa.

```
SafeArrayUnaccessData( pSa );
```

When SAFEARRAY becomes unnecessary, SafeArrayDestroy() releases the area allocated in SAFEARRAY. The definition of SafeArrayDestroy() is shown below.

```
HRESULT SafeArrayDestroy( SAFEARRAY* psa );
```

The argument is a pointer to released SAFEARRAY.

Finally, an example of showing the whole of processing with SAFEARRAY is shown. In this example, the array of the short type is substituted into created SAFEARRAY.

```
// The array of the short type is prepared.
short sample[10] = { 1, 2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 };
// The pointer of SAFEARRAY is declared.                                (1)
SAFEARRAY* pSa;
// Because the instance of SAFEARRAY doesn't exist, create it.
SAFEARRAYBOUND bound = { 10 ,0 };
pSa = SafeArrayCreate( VT_I2, 1, &bound );
// Access SAFEARRAY.
short* iArray;                                                          (2)
SafeArrayAccessData( pSa, (void**)&iArray);
for( int i = 0; i < 10; i++){
        iArray[ i ] = sample[ i ];
}
SafeArrayUnaccessData( pSa );                                           (3)
```

The above example is copying an array of the short type into an array of SAFEARRAY.

Because the instance of SAFEARRAY doesn't exist, it is necessary to create it. Therefore, as shown at (1), a pointer (pSa) to created SAFEARRAY is prepared. Then, the structure of the array is defined by SAFEARRAYBOUND type. In this example, elements number is defined to 10 and the index lower bound value is defined as 0. With these definitions, SAFEARRAY is created by SafeArrayCreate(). Because the data stored in the example is an array of the short type, the type of SAFEARRAY becomes VT_I2.

Next, a pointer is prepared to access the data of SAFEARRAY. As shown at (2) of the list, pointer (iArray) of the short type is prepared because SAFEARRAY of VT_I2 is created from SafeArrayCreate() in this example. Then, the SafeArrayAccessData() function is called. The second argument is assumed to be (void**) and casted to match the function type.

While accessing to SAFEARRAY, a pointer to SAFEARRAY data (iArray) holds the first address of the array data. In this example, the data of short type array is copied to SAFEARRAY. After the copy ends, the access is closed with SafeArrayUnaccessData() as shown in (3) under the list.

If all array is not accessed together, but each element of array is accessed separately, use SafeArrayPutElement(). The definition of SafeArrayPutElement() is shown below.

```
HRESULT SafeArrayPutElement( SAFEARRAY * psa, long * rgIndices, void * pv );
```

Here, psa is a pointer to SAFEARRAY to be accessed, rgIndices is an index of SAFEARRAY to be accessed, and pv is the value to be set.

Following is an example of substituting short type array to created SAFEARRAY.

```
// The array of the short type is prepared.
short sample[10] = { 1, 2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 };
// The pointer of SAFEARRAY is declared.
SAFEARRAY* pSa;
// Because the instance of SAFEARRAY doesn't exist, create it.
SAFEARRAYBOUND bound = { 10 ,0 };
pSa = SafeArrayCreate( VT_I2, 1, &bound );
// Access SAFEARRAY.
for( int i = 0; i < 10; i++){
      SafeArrayPutElement( psa, &i, &sample[i] );
}
```

2.2.4.4. HRESULT

HRESULT is 32bit numerical data with a structure to store error code. Many kinds of the error of HRESULT are prepared by < winerr.h >. Some of the important erro codes are shown in Table 2-4.

To assign error code to HRESULT, substitute it. However, direct comparison of HRESULT to judge the success or failure of function execution is not desirable. Instead, use macros named SUCCEEDED() and FAILED(). If the argument HRESULT is normal termination, SUCCEEDED() return TRUE, and otherwise it returns FALSE. If argument HRESULT is abnormal termination, FAILED() returns TRUE, and otherwise, it returns FALSE.

In the following example, S_OK is substituted into HRESULT, and it is judged by FAILED().

```
// S_OK is substituted for hr of the HRESULT type.
HRESULT hr = S_OK;
// FAILED() judges whether hr is S_OK or not.
if( FAILED( hr ) )
{
// Error processing
}
return hr;
```

2.2.5. Notation of data

ORiN2 provides a method to express VARIANT type by the character string. Using the expression, even in the environment where VARIANT type is not supported, pseudo-VARIANT type can be used. This data description method is used in the transmission character string of RAC, or item value expression of CaoUPnP.

The format expresses the data type and the data row separated by comma.

< data type >,< data row >

Data type is expressed by a VARTYPE integer value. Following table shows the available data type and corresponding value.

Table 2-2 Available data type

Data type	Value	Meaning
VT_I2	2	Two byte integer type
VT_I4	3	Four byte integer type

VT_R4	4	Single precision floating point type
VT_R8	5	Double precision floating point type
VT_CY	6	Currency type
VT_DATE	7	Date type
VT_BSTR	8	Character string type
VT_BOOL	11	Boolean type
VT_VARIANT	12	VARIANT type
VT_UI1	17	Byte type
VT_ARRAY	8192	Array type

An array of data is expressed by logical AND of the data type of element and VT_ARRAY.

Data row is expressed in the character string. Array data is expressed using “,” (comma) delimiter.

Example1)	2,100	Type:	VT_I2	Value:	100
Example 2)	8,Sample	Type:	VT_BSTR	Value:	Sample
Example 3)	8194,100,200,300	Type:	VT_I2 ｜VTARRAY	Value:	100,200,300
Example 4)	8200,Sample,Test	Type:	VT_BSTR ｜VTARRAY	Value:	“Sample”,“Test”
Example 5)	8,Sample,Test	Type:	VT_BSTR	Value:	“Sample,Test”

2.2.6. Log output

CAO engine has log function, which records start and stop of CAO.exe, object creation and deletion, and other operations. Five types of log output, i.e., console, message box, event viewer, debugging viewer, and text file, can be selected as the output destination of the log. Use CaoConfig to set output. 6CaoConfig describes how to use the command.

Following is CAO engine log output timing.

・ When CAO engine starts and stops.
・ When objects such as CaoController and CaoVariable are created or deleted.
・ When a controller thread starts/ends or its operation starts/stops.
・ When log is recorded by a message event from CAO provider[7].

2.2.7. Error code

The source of ORiN2 error can be categorized in several modules. Following explains errors generated in CAO module.

[7] Please refer to "3.4.1. the log output by the message event" of 'CAO provider making guide' for the method of outputting the log by the message event of the CAO provider.

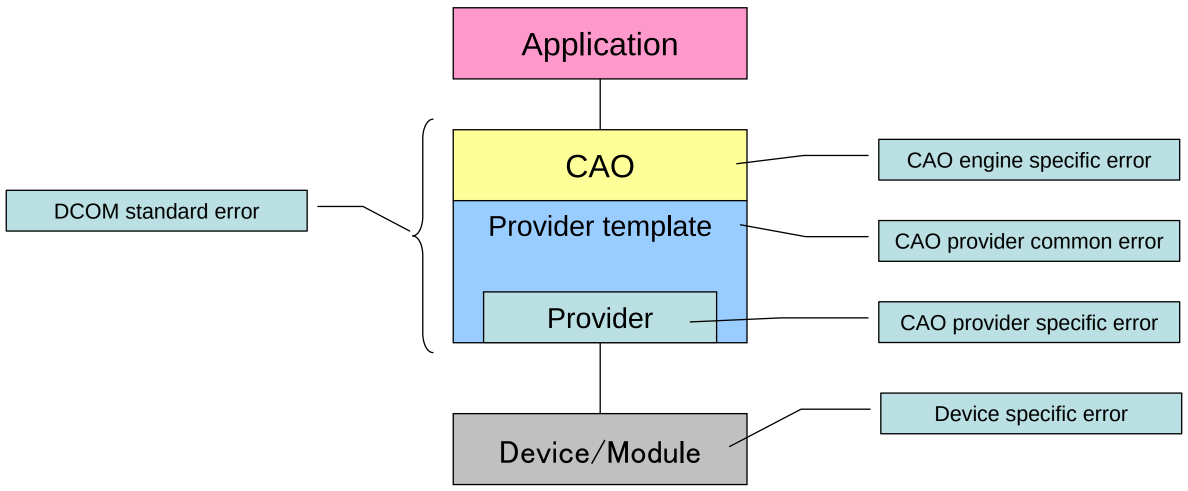


Figure 2-5 Errors generated in ORiN2

Table 2-3 ORiN2 error type

Error type	Error generation module	Explanation
DCOM standard error	-	General error in DCOM. For details, please refer to Table 2-4.
CAO engine specific error	CAO engine	CAO engine specific errors. For details, please refer to Table 2-5.
CAO provider common error	CAO provider template	Common error for CAO provider. For details, please refer to Table 2-6.
CAO provider specific error	CAO provider	Error code specifically defined for each CAO provider. For details, please refer to the users guide for each provider.
Device specific error	Device	Error code specifically defined for each device driver. For details, please refer to the manual for each device driver.

Table 2-4 DCOM common error (part)

Error name	Error code	Explanation
S_OK	0x00000000	Normal finish (0x0) with returning logical TRUE.
S_FALSE	0x00000001	Normal finish (0x1) with returning logical FALSE.

E_UNEXPECTED	0x8000FFFF	Catastrophic failure.
E_NOTIMPL	0x80004001	Not implemented.
E_OUTOFMEMORY	0x8007000E	Ran out of memory.
E_INVALIDARG	0x80070057	One or more arguments are invalid.
E_POINTER	0x80004003	Invalid pointer.
E_HANDLE	0x80070006	Invalid handle.
E_ABORT	0x80004004	Operation aborted.
E_FAIL	0x80004005	Unspecified error.
E_ACCESSDENIED	0x80070005	General access denied error.
E_WINDOWS_MASK	0x8007xxxx	Windows standard error codes (16-bit) are stored into xxxx (the least significant two bytes) Ex） Error code: 2（File not found） --> 0x80070002

Table 2-5 CAO engine specific error

Error name	Error code	Explanation
E_CAO_SEM_CREATE	0x80000200	It failed in the generation of the synchronous semaphore.
E_CAO_PROV_INVALID	0x80000201	The CAO access provider name is invalid.
E_CAO_COMPUTER_NAME	0x80000202	The computer name can not be acquired.
E_CAO_VARIANT_TYPE_NOSUPPORT	0x80000203	The VARIANT type which isn't supported was handed over.
E_CAO_OBJECT_NOTFOUND	0x80000204	A corresponding object isn't found out.
E_CAO_COLLECTION_REGISTERED	0x80000205	It is already registered on the collection.
E_CAO_THREAD_CREATE	0x80000207	It failed in the generation of the work thread.
E_CAO_REMOTE_ENGINE	0x80000208	Can not access the remote CAO server.
E_CAO_REMOTE_PROVIDER	0x80000209	Can not access the remote CAO's provider.
E_CAO_NOT_WRITABLE	0x8000020a	Can not write a data.
E_CAO_CMD_EXECUTE	0x8000020b	A command is under execution.
E_CAO_PROV_NO_LICENSE	0x8000020c	The specified provider is not licensed.
E_CAO_PRELOAD	0x8000020d	Failed CRD preload.
E_CAO_TEMP_REGISTERED	0x8000020e	Temporarily registered in the collection.
E_CAO_NO_PARENT	0x8000020f	There wa no parent object.

Table 2-6 CAO provider common error

Error name	Error code	Explanation
E_CRDIMPL	0x80000400	Improper CRD file
E_CAOP_SYSTEMNAME_INVALID	0x80000401	It is an invalid system variable name.
E_CAOP_SYSTEMTYPE_INVALID	0x80000402	It is an invalid system variable type.
E_CANCEL	0x80000403	The command was canceled.
E_CAOP_NOT_WRITABLE	0x80000404	It is read-only parameter.
E_TIMEOUT	0x80000900	Timeout
E_NO_LICENSE	0x80000901	No license.
E_NOT_CONNECTED	0x80000902	Connection is not established.
E_NOT_USE	0x80000903	It is not used.
E_INVALID_CMD_NAME	0x80000904	It is illegal command.
E_MAX_OBJECT	0x80000905	Exceeded the number of creatable object limit.
E_OVERLOADING	0x80000906	The setting is duplicated.
E_NONE_CONNECT_OPTION	0x80000907	Connection option is not found.
E_ALREADY_REGISTER	0x80000908	The name you have entered has already used.
E_TOO_MUCH_DATA	0x80000909	Data size is too large.
E_FAILED_ALLOC	0x8000090a	Failed to secure buffer.
E_MAX_CONNECT	0x8000090b	Exceeded the number of connection count limit.
E_CONVERT_CHAR_CODE	0x8000090c	Failed to convert character code.
~~E_WINDOWS_MASK~~	~~0x8090xxxx~~	~~Windows standard error codes (16-bit) are stored into xxxx (the least significant two bytes)~~ ~~Ex)~~ ~~Error code: 2（File not found）~~ ~~→ 0x80900002~~ (Notes) This error code was changed to 0x8007xxxx (see table 2-4)
E_WINSOCK_MASK	0x8091xxxx	Windows winsock error codes (16-bit) are stored into xxxx (the least significant two bytes) Ex) Error code: 10061（Connection denied）

		--> 0x8091274D

2.3. An installation state of ORiN2 SDK

Regarding the way to confirm an installation state of ORiN2 SDK, please refer to chapter 3.7 of "ORiN2 SDK User's Guide".

2.4. CAO client implementation

2.4.1. Open controller

To open controller, take the following procedure.

(1) Prepare a variable to maintain an object.
(2) Create a CaoEngine object.
(3) Acquire or generate a CaoWorkspace object.
(4) Create a CaoController object.

Following is detailed explanation. In this explanation, early binding is used, and New keyword is used to create an object.

(1) At first, declare a variable to store object. CaoEngine object and CaoWorkspace object are necessary to open controller. Then, AddController method will create CaoController object. Therefore, the variables corresponding to these three objects are prepared. Following is an example of declaring the variable of each object type as a private variable.

```
Private caoEng As CaoEngine 'Engin object
Private caoWs As CaoWorkspace 'CaoWorkSpace object
Private caoCtrl As CaoController 'Controlle object
```

(2) Next, create a CaoEngine object. The object is created by New keyword, similar to an example in 2.2.1.1 The object is substituted into variable using Set statement.

```
Set caoEng = New CaoEngine
```

(3) When CaoEngine object is created, a CaoWorkspaces object and a CaoWorkspace object are also created. To acquire the default object of CaoWorkspace, use CaoWorkspaces.Item(0). Following is an example of acquiring the default CaoWorkspace object.

```
Set caoWS = caoEng.CaoWorkspaces.Item(0)
```

(4) A CaoController object can be created by the AddController method of the CaoWorkspace object. The definition of the AddController method is shown here.

```
AddController(BSTR bstrController, BSTR bstrProvider, BSTR bstrMachine, BSTR bstrPara)
```

The arguments of this method are, the robot controller name, the provider name, the machine name, and a parameter. And, a CaoController object is returned as a return value. The provider name registered here can be acquired in an array by using the ProviderNames method of the CaoEngine object. Because the meaning of the parameter is different for each provider, please refer to the provider user's guide for details. Following is an example of using AddController method with the Blackboard provider.

```
' CaoCtrl and CaoWS are variables to maintain the object.
Set CaoCtrl = CaoWS.AddController(“bb1″,“CaoProv.Blackboard″, “″, “″)
' The third and the foruth argument can be omitted.
Set CaoCtrl = CaoWS.AddController(“bb1″,“CaoProv.Blackboard″)
```

2.4.2. Retrieve and set variable

A CaoVariable object needs to be created previously to acquire and to set the variable.

(1) The AddVaribale method of the CaoController object is used to create a CaoVariable object. The definition is shown as follows.

```
AddVariable(BSTR bstrName, BSTR bstrOption )
```

At this time, bstrName is a Variable name, and bstrOption is an option when the variable is retrieved. This method returns a CaoVariable object as a return value. Following is an example of retrieving a CaoVariable object of Variable name “Val” using AddVariable(),. Here, bstrOption can be omitted.

```
'CaoCtrl and CaoVar are variables to hold the object.
Set CaoVar = CaoCtrl.AddVariable(“Val”)
```

After the CaoVariable object is created, refer to the Value property of the CaoVariable object to set and to retrieve the value of the variable. Following is an example of setting value “1000" of Long type, and then retrieve the variable.

```
' CaoCtrl and CaoWS are variables to hold the object.
Dim iData as Integer
IData = 1000
' The value of iData is set to the variable.
CaoVar.Value = iData
' The value of the variable is retrieved and substituted to iData.
iData = CaoVar.Value
```

2.4.3. Retrieve and set system variable

Some providers may prepare peculiar system variable. The Blackboard provider used in the example has following system variables. Please refer to the user's guide of each provider for details.

・@COUNT	Number of user variables
・@CURRENT_TIME	Present time
・@VERSION	Version

Following is an example of retrieving present time as a system variable.

```
'Retrieve system variable @CURRENT_TIME
Set caoVar = caoCtrl.AddVariable("@CURRENT_TIME")
'Retrieve present time from variable object
Dim var as Date
var = caoVar.Value
```

2.4.4. Event processing

When the variable is added and the value of the variable is changed, the event is generated in the Blackboard provider as previously described.

To confirm the generation of the event, a simple sample program is made. First of all, please make the form with three text boxes with VB6, and describe the following codes.

List 2-1 **Form1.frm**

```
Dim caoEng As CaoEngine
Dim caoWs As CaoWorkspace
Dim WithEvents caoCtrl As CaoController

'Create CAO engine and connect to Blackboard provider
Private Sub Form_Load()
        Set caoEng = New CaoEngine
        Set caoWs = caoEng.Workspaces(0)
        Set caoCtrl = caoWs.AddController("BB1", "CaoProv.Blackboard")
End Sub


'Blackboard controller's event procedure
Private Sub caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)
        Text1.Text = pICaoMess.Description
        Text2.Text = pICaoMess.Destination
        Text3.Text = pICaoMess.Value
End Sub
```

Then, start the application, and confirm the generation of the event.

First of all, when you start the application, only default string is displayed in the text box as shown in Figure2-6.

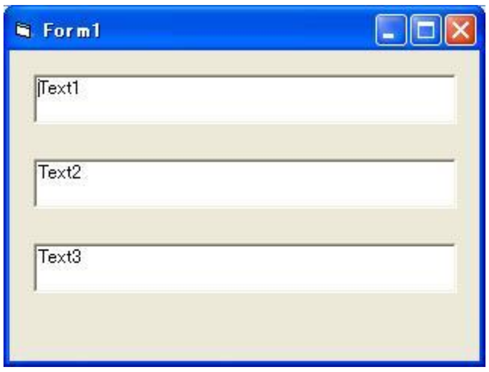
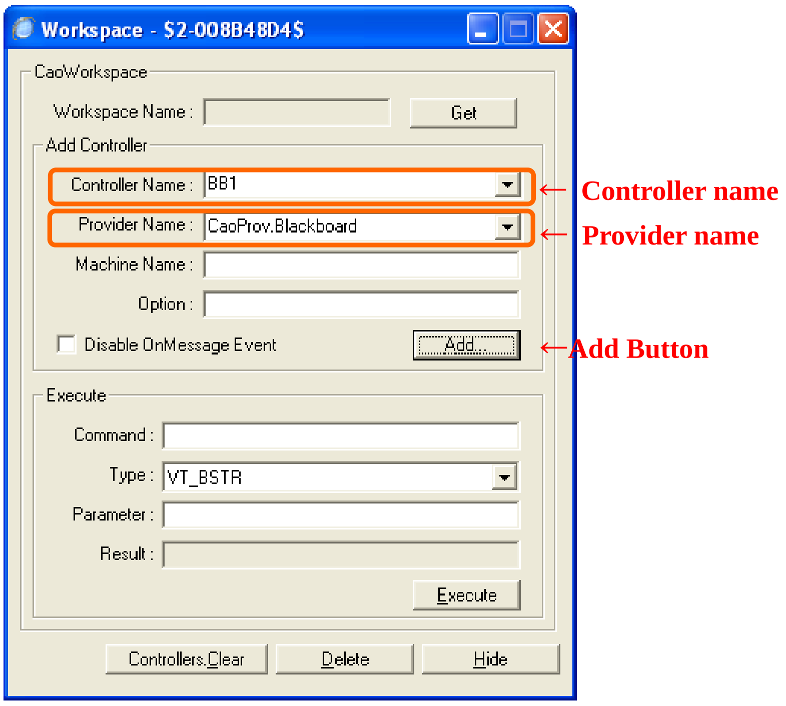


Figure2-6 Start of sample application

Next, an event is generated. CaoTester, which is included in ORiN2 SDK, is used to generate an event.

Please refer to the CaoTester user's guide for use of CaoTester.

Start CaoTester, and input controller name as ”BB1”, and select “CaoProv.Blackboard", as shown in Figure2-7 of the workspace child window. Then, press the [Add] button.


Workspace - $2-008B48D4$

CaoWorkspace

Workspace Name :

Add Controller

Controller Name : BB1

Provider Name : CaoProv.Blackboard

Machine Name :

Option :

Disable OnMessage Event

Execute

Command :

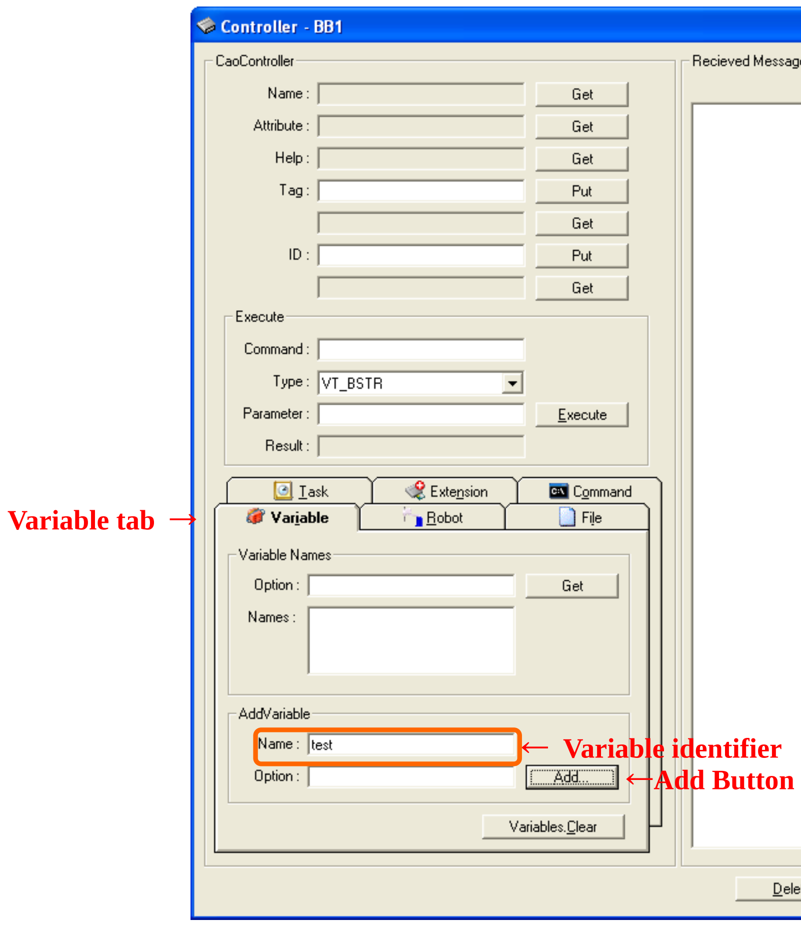
Type : VT_BSTR

Parameter :

Result :

Figure2-7 Connection to Blackboard provider

When you press the [Add] button, a child window is displayed as shown in Figure2-8. Select the Variable tab, input to the variable name (Name of AddVariable) as “test", and press the Add button.



Type : VT_BSTR

Name : test

Figure2-8 Addition of variable object

When you press the [Add] button, a child variable window is displayed as shown in Figure2-9. Please input “VT_I4" to variable type, and “1234" to variable value, and press [Put] button.

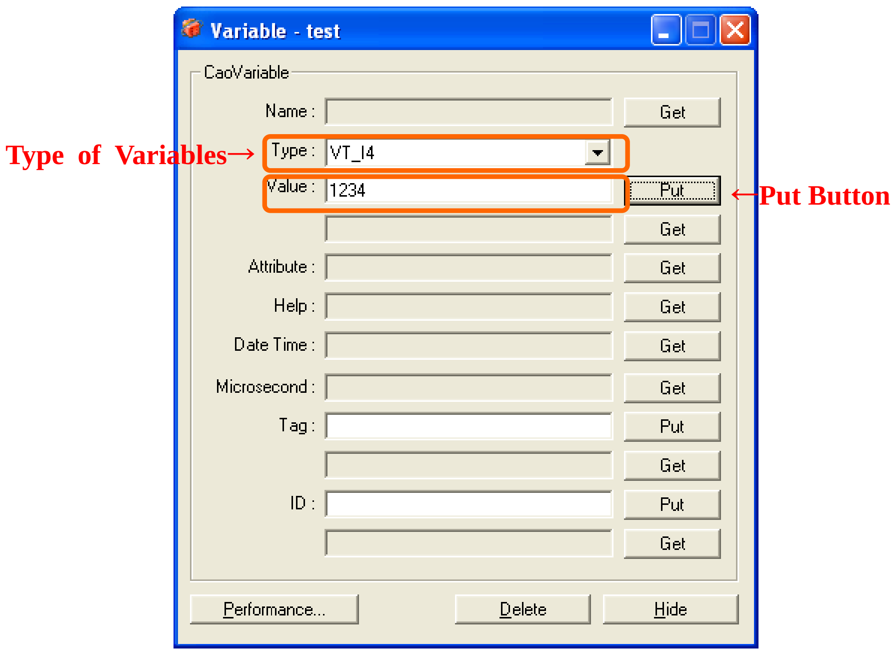
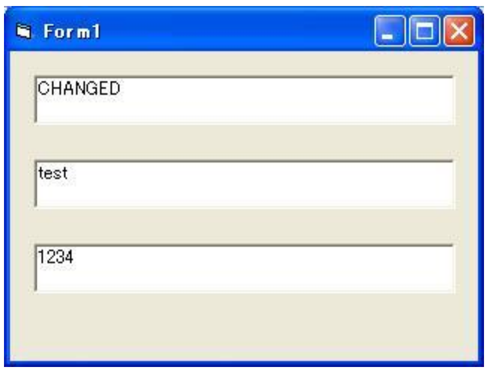


Figure2-9 Writing of variable

Now check the window of the sample program. As shown in Figure2-10, an event is generated, and the kind of the event, variable name, and variable value is displayed in the text box by the process of event procedure.



CHANGED

test

1234

Figure2-10 Generation of event

2.5. Developing client with VB.NET

2.5.1. RCW registration to GAC

VB.NET client accesses COM component through a module called Runtime Callable Wrapper (RCW). When the client application is created, the RCW module is automatically created automatically and locally from type library. Therefore, each VB.NET application will create its own RCW module. To avoid this situation, ORiN2 SDK prepares RCW module that can be registered in Global Assembly Cache (GAC).

There are three ways to register ORiN RCW to GAC.

(1) Automatic registration by ORiN2SDK installer.

Note that the automatic installation is possible only when the following conditions are fulfilled.

・ ORiN2SDK installer version is 2.1.2 or later.

・ Microsoft .NET Framework 2.0 or later is already installed.

(2) With Windows explorer, drop CaoRCW.dll into Windows¥assembly directory.

(3) Use global assembly cache tool（Gacutil.exe）8

Registration	:	ORiN2¥DotNet¥RCW> gacutil –i CaoRCW.dll
Deregistration	:	ORiN2¥DotNet¥RCW> gacutil –u CaoRCW

2.5.2. VB.NET project setting

To set RCW, select [Add reference] dialog -> [.NET] tab, and add the RCW module of CAO. The RCW module is stored in the following directory.

ORiN2¥DotNet¥RCW¥CaoRCW.dll

By adding RCW module, class variables of CAO engine can be declared. Following is an example of CAO engine variable declaration.

```
Dim caoEng As New ORiN2.interop.CAO.CaoEngine
```

2.5.3. Notes on object deletion

Because garbage collection is used in VB.NET, object is not released even if “Noting” is substituted into a variable. Therefore, when the object is deleted, the resource for the object need to be explicitly released using system.Runtime.InteropServices.Marshal.ReleaseComObject function. Following is an example of releasing controller object.

```
caoWS.Controllers.Remove(caoCtrl.Index)
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoCtrl)
caoCtrl = Nothing
```

System.Runtime.InteropServices.Marshal.ReleaseComObject is a function to decreases COM reference counter managed by garbage collector. If the reference count is n, ReleaseComObject need to be called n times before releasing the object.

Garbage collector also manages auto variables. Therefore, in the following case,

```
caoEng.Workspaces.Item(0).Controllers.Clear
```

8 Using gacutil.exe is possible only when.NET Framework 2.0 SDK or Visual Studio 2005 is already installed.

[.Workspaces] is also assigned to a temporary automatic variable and managed by garbage collector. The situation is same for [.Item(0)] and [.Controllers]. Therefore, reference counters for each of these automatic variables are incremented by one, and ReleaseComObject need to be called until reference counter for these automatic variables becomes 0 before releasing the variable.

For these reasons, managing automatic variable reference count is necessary. However, managing the reference count is troublesome, and may be omitted.

To avoid the reference count management problem, following example code declares local variables are explicitly. In this way, COM automatic variable assignment will not happen.

```
Dim caoEng As CaoEngine
Dim caoWss As CaoWorkspaces
Dim caoWs As CaoWorkspace
Dim caoCtrls As CaoControllers

caoEng = new CaoEngine
caoWss = caoEng.Workspaces
caoWs = caoWss.Item(0)
caoCtrls = caoWs.Controllers

Dim caoCtrl As CaoController
caoCtrl = caoWs.AddController("RC1", " CaoProv.Blackboard ", "", "")

caoCtrls.Clear
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoCtrl)
caoCtrl = Nothing

caoWss.Clear
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoCtrls)
caoCtrls = Nothing
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoWs)
caoWs = Nothing
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoWss)
caoWss = Nothing
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoEng)
caoEng = Nothing
```

2.5.4. Method of handling event

To handle an event, an event handler needs to be dynamically added and removed using AddHandler and the RemoveHandler key word. Following is an example of creation and deletion of event handler.

(1) A controller object is created and an event handler for the object is added.

```
caoCtrl = caoWs.AddController("BB1", "CaoProv.Blackboard ", "", "")
AddHandler caoCtrl.OnMessage, AddressOf caoCtrl_OnMessage
```

(2) Event handler process is defined.

```
Private Sub caoCtrl_OnMessage(ByVal pICaoMess As ORiN2.interop.CAO.CaoMessage)
       txtReceiveTime.Text = pICaoMess.DateTime
End Sub
```

(3) The event handler is removed before the controller object is deleted.

```
RemoveHandler caoCtrl.OnMessage, AddressOf caoCtrl_OnMessage
caoCtrls.Remove(caoCtrl.Index)
System.Runtime.InteropServices.Marshal.ReleaseComObject(caoCtrl)
caoCtrl = Nothing
```

2.6. Developing client with other languages

2.6.1. Client development with C++[9]

Following is the procedure to use CAO object with C++.

(1) Initialize DCOM

(2) Create CaoEngine object

(3) Execute method

(4) Release object

(5) Terminate DCOM

There are two methods to handle CAO object, i.e.

・ #Include directive

・ #Import directive

Details of each method are described below.

2.6.1.1. Using #include directive

To operate CAO object with C++, CAO header file (CAO.h) and UUID definition file (CAO_i.c) need to be included. These files are in "< ORIN2 root > ¥CAO¥Include".

```
#include "CAO.h"
#include "CAO_i.c"
```

Following is the actual procedure to operate CAO object.

(1) Initialize DCOM

Execute CoInitialize() before the CAO object is operated.

(2) Create CaoEngine object

Execute CoCreateInstance(). Following shows an example of arguments.

```
ICaoEngine* pEng;
hr = CoCreateInstance(CLSID_CaoEngine,
                              NULL,
                              CLSCTX_LOCAL_SERVER,
                              IID_ICaoEngine,
                              (void **)&pEng);
```

(3) Executie methods

Methods of CaoEngine are executed.

Following example executes AddController.

[9] Please refer to the following pages for a detailed explanation of DCOM.
“http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/jpdnguion/htm/msdn_drguion020298.asp”

```
// Retrieve CaoWorkspace collection
ICaoWorkspaces *pWss;
hr = pEng->get_Workspaces(&pWss);
if (FAILED(hr)) return hr;

// Retrieve CaoWorkspace
ICaoWorkspace *pWs;
hr = pWss->Item(CComVariant(0L), &pWs);
if (FAILED(hr)) return hr;

// Creation of CaoController
ICaoController *pCtrl;
hr = pWs->AddController(CComBSTR(L"TestCtrl"),
                                   CComBSTR(L"CaoProv.DataStore"),
                                   CComBSTR(L""),
                                   CComBSTR(L""),
                                   &pCtrl);
```

(4) Release object

Objects created or retrieved in a program must be released. To release object, execute Release(). The example shows an example of releasing object.

```
pCtrl->Release();
pWs->Release();
pWss->Release();
pEng->Release();
```

(5) Terminate DCOM

Execute CoUnitialize()before the program ends.

Following is a sample program to set and get variable

List 2-2 **CPPSample1.cpp**

```
#include "atlbase.h"
#include "CAO.h"
#include "CAO_i.c"

HRESULT func1();

int main(int argc, char* argv[])
{
       CoInitialize(0);

       HRESULT hr = func1();

       CoUninitialize();
       return 0;
}

HRESULT func1()
{
       HRESULT hr = S_OK;
       ICaoEngine* pEng = NULL;
       ICaoWorkspaces *pWss = NULL;
       ICaoWorkspace *pWs = NULL;
       ICaoController *pCtrl = NULL;
       ICaoVariable *pVar = NULL;
       CComVariant vntVal;

// Create CaoEngine
       hr = CoCreateInstance(CLSID_CaoEngine,
                                NULL,
                                CLSCTX_LOCAL_SERVER,
                                IID_ICaoEngine,
                                (void **)&pEng);
        if (FAILED(hr)) {
                goto EndProc;
        }

        // Retrieve CaoWorkspace collection
        hr = pEng->get_Workspaces(&pWss);
        if (FAILED(hr)) {
                goto EndProc;
        }

        // Retrieve CaoWorkspace
        hr = pWss->Item(CComVariant(0L), &pWs);
        if (FAILED(hr)) {
                goto EndProc;
        }

        // Create CaoController
        hr = pWs->AddController(CComBSTR(L"TestCtrl"),
                                 CComBSTR(L"CaoProv.DataStore"),
                                 CComBSTR(L""),
                                 CComBSTR(L""),
                                 &pCtrl);
        if (FAILED(hr)) {
                goto EndProc;
        }

        // Create CaoVariable
        hr = pCtrl->AddVariable(CComBSTR(L"TestVal"), CComBSTR(L""), &pVar);
        if (FAILED(hr)) {
                goto EndProc;
        }

// Set and get value
        pVar->put_Value(CComVariant(L"sample"));
        pVar->get_Value(&vntVal);

        // Release object
EndProc:
        if (pVar) pVar->Release();
        if (pCtrl) pCtrl->Release();
        if (pWs) pWs->Release();
        if (pWss) pWss->Release();
        if (pEng) pEng->Release();
        return hr;
}
```

2.6.1.2. Using #import directive

Another way to operate CAO object with C++ is to import CAO.exe using #import directive.

```
#import "CAO.exe"
```

The name space of the CAO object is "CAOLib"[10].

By using #import directive, a smart pointer, which is represented by object name followed by "Ptr" can be

[10] The setting of the namespace can be set by the option of # import directive. Please refer to MSDN for details.

used. (Example: ICaoEnginePtr)

Following is the actural procedure to operate CAO object.

(1) Initialize DCOM

Execute CoInitialize()before the CAO object is operated.

(2) Create CaoEngine object

The object is created using IcaoEnginePtr, which is a smart pointer of CaoEngine object. Following is an example. If smart pointer is not used, create object in the same way as described in 2.6.1.1

```
CAOLib::ICaoEnginePtr pEng(__uuidof(CAOLib::CaoEngine));
```

(3) Execute method

Execute the method of CaoEngine which is created in (2).

Following is an example of executing AddController.

```
CAOLib::ICaoWorkspacesPtr pWss = pEng->GetWorkspaces();
CAOLib::ICaoWorkspacePtr pWs = pWss->Item(0L);
CAOLib::ICaoControllerPtr pCtrl = pWs->AddController(L"SampleCtrl",
                                                     L"CaoProv.DataStore",
                                                     L"",
                                                     L"");
```

(4) Release object

Objects held by smart pointers are automatically released. To release object explicitly, or to release object that is not held by a smart pointer, follow the similar procedure of 2.6.1.1.

(5) Terminate DCOM

Execute CoUnitialize()before the program ends.

Following sample program sets and retrieves the variables.

List 2-3 **CPPSample2.cpp**

```
#import "D:¥work¥Robot¥Repos¥ORiN2¥CAO¥Engine¥Bin¥CAO.EXE"

HRESULT func1();

int main(int argc, char* argv[])
{
        CoInitialize(0);

        HRESULT hr = func1();

        CoUninitialize();
        return 0;
}

HRESULT func1()
{
        try {
                CAOLib::ICaoEnginePtr pEng(__uuidof(CAOLib::CaoEngine));
                CAOLib::ICaoWorkspacesPtr pWss = pEng->GetWorkspaces();
                CAOLib::ICaoWorkspacePtr pWs = pWss->Item(0L);
```

```
                    CAOLib::ICaoControllerPtr pCtrl = pWs->AddController(L"SampleCtrl",
                                                              L"CaoProv.DataStore",
                                                              L"",
                                                              L"");
                    CAOLib::ICaoVariablePtr pVar = pCtrl->AddVariable(L"SampleVal", L"");

                    pVar->PutValue(_variant_t(L"sample"));
                    _variant_t vntTemp = pVar->GetValue();

            }
            catch (_com_error e) {
                    return e.Error();
            }
            return S_OK;
    }
```

2.6.2. Client development with C

When using C language, #include directive is used and the CAO object is operated. CAO header file (CAO.h) and UUID definition file (CAO_i.c) are included. These files are in "<ORIN2 root directory> ¥CAO¥Include".

```
#include "CAO.h"
#include "CAO_i.c"
```

The procedure for operating the object of CAO is similar to C++ described in 2.6
Following is a detailed procedure.

(1) Initialize DCOM

CoInitialize() is executed before the CAO object is operated.

(2) Create CaoEngine object

CoCreateInstance() is executed. Following is an example.

```
hr = CoCreateInstance(&CLSID_CaoSQLEngine,
                                    NULL,
                                    CLSCTX_ALL,
                                    &IID_ICaoSQLEngine,
                                    (void **)&pEng);
```

(3) Execute method

Execute methods of CaoEngine that created in (2). Methods are called and executed as following. This is an example of CaoWorkspace collection retrieval.

```
pEng->lpVtbl->get_Workspaces(pEng, &pWSs);
```

If you declare a pre-processor symbol named COBJMACROS before header files are included, macros corresponding to each method become available. Following example shows retrieval of CaoWorkspace collection using the macro

```
ICaoEngine_get_Workspaces(pEng, &pWss);
```

(4) Release object

The object that is generated or retrieved in the program should be released. Objects are released by executing Release(). Following example shows example of releasing object using macro.

```
ICaoEngine_Release(pEng)
```

(5) Terminate of DCOM

Before program ends, execute CoUnitialize() to terminate DCOM.

Following is a sample program to set and get variables.

List 2-4 **ImportSample.cpp**

```
#import "D:/work/Robot/Repos/ORiN2/CAO/Engine/Bin/CAO.EXE"

HRESULT func1();

int main(int argc, char* argv[])
{
        CoInitialize(0);

        HRESULT hr = func1();

        CoUninitialize();
        return 0;
}

HRESULT func1()
{
        try {
                CAOLib::ICaoEnginePtr pEng(__uuidof(CAOLib::CaoEngine));
                CAOLib::ICaoWorkspacesPtr pWss = pEng->GetWorkspaces();
                CAOLib::ICaoWorkspacePtr pWs = pWss->Item(0L);
                CAOLib::ICaoControllerPtr pCtrl = pWs->AddController(L"SampleCtrl",

        L"CaoProv.DataStore",
                                                                        L"",
                                                                        L"");
                CAOLib::ICaoVariablePtr pVar = pCtrl->AddVariable(L"SampleVal", L"");

                pVar->PutValue(_variant_t(L"sample"));
                _variant_t vntTemp = pVar->GetValue();

        }
        catch (_com_error e) {
                return e.Error();
        }
        return S_OK;
}
```

2.6.3. Client development with C#

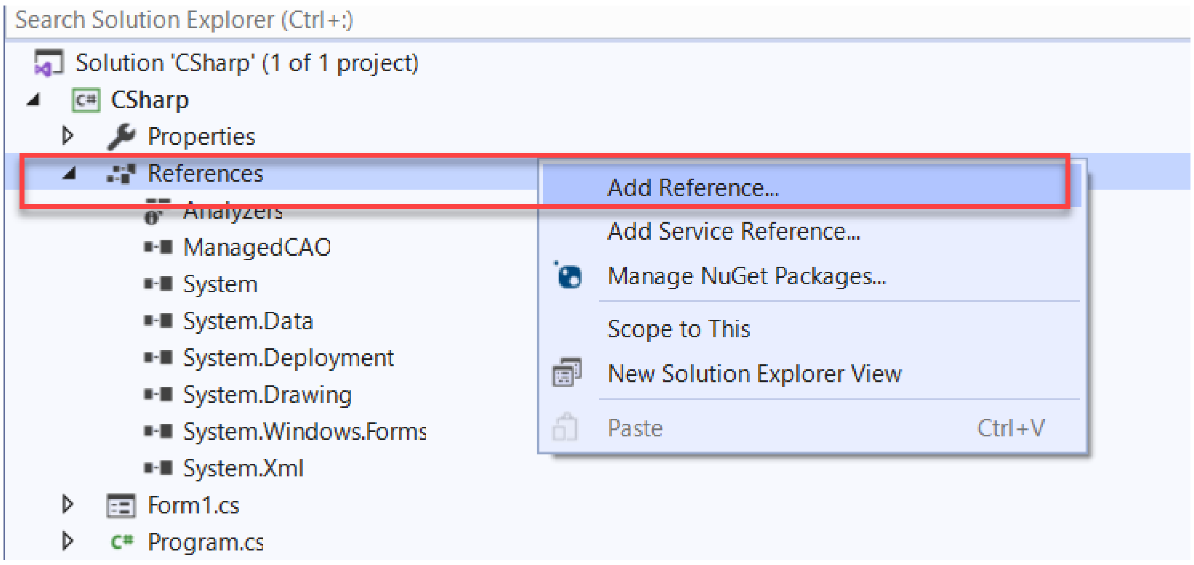
ORiN2 SDK provide two methods for creating C# client applications, ManagedCAO and RCW, but we currently recommend the more efficient ManagedCAO method. While explicit description of object release processing is essential with RCW, ManagedCAO does not require explicit description of that part, and is more efficient in that it is designed to automatically release objects in response to sudden termination of the client application, which has been an issue with RCW. ManagedCAO is more efficient in that it does not require an explicit description of this part.

2.6.3.1. Using ManagedCAO module

A C# client accesses the COM component via ManagedCAO, which wraps RCW.

2.6.3.1.1. Configuration when creating a new project

Add a reference to ManagedCAO to the C# client application project.



ManagedCAO corresponds to the following modules under the installation folder of ORiN2 SDK.
"<ORiN2 installed folder>/DotNet/ManagedCAO/Lib/ManagedCAO.dll"

By adding a reference to the ManagedCAO module, <ORiN2.ManagedCAO> will be recognized in the namespace, so please declare "using ORiN2.ManagedCAO;" as follows.

```
using System;
using System.Windows.Forms;
using ORiN2.ManagedCAO;
```

This allows class variables of the CAO engine to be declared as follows.

```
public partical class Form1 : Form
{
        private CCaoEngine _engine = null;
        private CCaoController _controller = null;
        private CCaoVariable _variable = null;
        public Form1()
        :
    }
```

2.6.3.1.2. Notes when deleting an object

In ManagedCAO, when the Dispose function of CCaoEngine is called, the Cao objects under it are automatically disposed of. Therefore, when initialization fails or when the application terminates, the CCaoEngine object should be released and disposed of.

In addition, if a message is received (event is acquired), cancel the message reception process before calling CCaoEngine's Dispose and call the Application.DoEvents function. This prevents CAO from terminating normally if the message object is not released.

The following is an example of a description defined as the ReleaseCaoObject function.

```
private void ReleaseCaoObject()
{

      if (_controller != null)
      {
          _controller.OnMessage -= new OnMessageEventHandler(_controller_OnMessage);
          Application.DoEvents();
       }

       if (_engine != null)
       {
           _engine.Dispose();
       }

       _variable = null;
       _controller = null;
       _engine = null;
}
```

The ReleaseCaoObject function defined above can be used to ensure that CAO terminates normally in the event of an exception. An example of its use is shown below.

```
private void Form1_Load(object sender, EventArgs e)
{
      try
      {
          _engine = new CCaoEngine();
          _controller = _engine.Workspaces[0].AddController( "Panel" ,  "CoProv.Dummy.Panel" );
          _variable = _controller.AddVariable( "Variable1" , null);

          _controller.OnMessage += new OnMessageEventHandler(_controller_OnMessage);
       }

       catch (Exception ex)
       {

           ReleaseCaoObject();
           this.Close();
       }
  }
```

2.6.3.1.3. How to call with thread-safe

In WindowsForms applications, access to the control from threads other than the UI thread is prohibited. Therefore, refer to the InvokeRequired property (true if the application is running outside the UI thread), and if outside the UI thread, process the control in the UI thread through the Invoke function (pass the process to the UI thread and wait for it to finish). In other cases, process in your own thread. In other cases, process in your own thread. An example is shown below.

```
void _controller_OnMessage(object sender, OnMessageEventArgs e)
{
    Action act = () =>
        {
         txtMessage.Text = string.Formart( “Number:{0}, Source:{1}, Value:{2}” ,
               e.Message.Number, e.Message.Source, e.Message.Value);
     };

    if (InvokeRequired)
    {
      Invoke(act);
    }
    else
    {
      act();
    }
}
```

The following sample code shows how to display a value of an OnMessage event into a text box.

List 2-5-1 **ManagedCaoSample.cs**

```
using System;
using System.Windows.Forms;
using ORiN2.ManagedCAO;

namespace Message
{
    public partial class frmMessage : Form
    {
        private CCaoEngine caoEng;
        private CCaoWorkspaces caoWss;
        private CCaoWorkspace caoWs;
        private CCaoControllers caoCtrls;
        private CCaoController caoCtrl;

        public frmMessage()
        {
            InitializeComponent();
        }

        private void frmMessage_Load(object sender, EventArgs e)
        {
            try
            {
                // CAO engine generation
                caoEng = new CCaoEngine();
                caoWss = caoEng.Workspaces;
                caoWs = caoWss[0];

                // Retrieving the controller collection
                caoCtrls = caoWs.Controllers;
```

```
                    // Connected to controller
                    caoCtrl = caoCtrls.Add("RC1", "CaoProv.Dummy", "", "");

                    // Register OnMessageEventHandler
                    caoCtrl.OnMessage += new OnMessageEventHandler(OnMessage);

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }

            private void frmMessage_FormClosed(object sender, FormClosedEventArgs e)
            {
                try
                {
                    // Releasing Controller Objects
                    if (caoCtrl != null)
                    {
                        caoCtrls.Remove(caoCtrl.Name);
                        caoCtrl = null;
                    }
                    caoCtrls = null;
                    caoWs = null;
                    caoWss = null;
                    caoEng = null;
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }

            private void cmdExecute_Click(object sender, EventArgs e)
            {
                try
                {
                    caoCtrl.Execute("", "");
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }

            // OnMessage event
            private void OnMessage(object sender, OnMessageEventArgs e)
            {
                Action act = () => { textBox1.Text = e.Message.Value.ToString(); };

                if (InvokeRequired)
                {
                    Invoke(act);
                }
                else
                {
                    act();
                }
            }
        }
    }
```

2.6.3.2. Using RCW module

A C# client program can access CAO (COM component) via RCW like a VB.NET client. The configuration is the same as 2.5.

2.6.3.2.1. Registration of RCW to GAC

The way of RCW registration is the same as VB.NET. Please refer to 2.5.1 for details.

2.6.3.2.2. Configuration when creating a new project

The way to add to "References" is the same as VB.NET. Please refer to 2.5.2 for details.

After adding the RCW module, the following class variable declaration of CaoEngine can be available.

private ORiN2.interop.CAO.CaoEngine caoEng;;

To simplify the above declaration, you can use "ORiN2.interop.CAO" as a name space. A way to declare the name space is as follows.

using ORiN2.interop.CAO;

2.6.3.2.3. Notes when deleting an object

A "Release" processing will not be called immediately even if a variable was set to null, because C# provides an automatic (delayed) garbage collection mechanism like a VB.NET. If it needs to delete an object immediately, you have to call "System.Runtime.InteropServices.Marshal.ReleaseComObject" function like VB.NET. Please refer to 2.5.3 for details.

2.6.3.2.4. How to receive an event

The way to receive an event with C# is as follows.

(1) Create a event handler

```
private void OnMessage(CaoMessage pICaoMsg)
{
        MessageBox.Show(pICaoMsg.Value.ToString());
}
```

(2) Register the event handler

```
caoCtrl = caoCtrls.Add("RC1", "CaoProv.Dummy", "", "");
caoCtrl.OnMessage += new _ICaoControllerEvents_OnMessageEventHandler(OnMessage);
```

2.6.3.2.5. How to call with thread-safe

Please note that an OnMessage event will be called back by another thread (not main thread) like VB.NET. You have to make a thread-safe program as follows.

(1) Create an event processing function.

(2) Create a delegate to call the above function.

```
private delegate void SetTextCallback(string msg);
```

(3) Call the delegate with “System.Invoke” function in an OnMessageEventHandler.

```
private void OnMessage(CaoMessage pICaoMsg)
{
       // create a delegate with the event processing function.
       SetTextCallback SetMsg = new SetTextCallback(SetText);

       // execute the function via the delegate with “System.Invoke”.
       Invoke(SetMsg, pICaoMsg.Value.ToString());
}
```

The following sample code shows how to display a value of an OnMessage event into a text box.

List 2-6-2 RcwSample.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ORiN2.interop.CAO;

namespace Message
{
    public partial class frmMessage : Form
    {

        private delegate void SetTextCallback(string msg);  // a delegate for OnMessage event
processing.

        private CaoEngine caoEng;
        private CaoWorkspaces caoWss;
        private CaoWorkspace caoWs;
        private CaoControllers caoCtrls;
        private CaoController caoCtrl;

        public frmMessage()
        {
            InitializeComponent();
        }

        private void frmMessage_Load(object sender, EventArgs e)
        {
            try
            {
                // createa a CAO engine
                caoEng = new CaoEngine();
                caoWss = caoEng.Workspaces;
                caoWs = caoWss.Item(0);

                // retrive a CaoController collection
                caoCtrls = caoWs.Controllers;

                // connect to the controller
                caoCtrl = caoCtrls.Add("RC1", "CaoProv.Dummy", "", "");

                // register an OnMessageEventHandler
                caoCtrl.OnMessage += new
_ICaoControllerEvents_OnMessageEventHandler(OnMessage);

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void frmMessage_FormClosed(object sender, FormClosedEventArgs e)
        {
            try
            {
                // Release a CaoController object
                if (caoCtrl != null)
                {
                    caoCtrls.Remove(caoCtrl.Name);
                    System.Runtime.InteropServices.Marshal.ReleaseComObject(caoCtrl);
                    caoCtrl = null;

                }
                System.Runtime.InteropServices.Marshal.ReleaseComObject(caoCtrls);
                caoCtrls = null;
                System.Runtime.InteropServices.Marshal.ReleaseComObject(caoWs);
                caoWs = null;
                System.Runtime.InteropServices.Marshal.ReleaseComObject(caoWss);
                caoWss = null;
                System.Runtime.InteropServices.Marshal.ReleaseComObject(caoEng);
                caoEng = null;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void cmdExecute_Click(object sender, EventArgs e)
        {
            try
            {
                caoCtrl.Execute("", "");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        // OnMessageEventHandler
        private void OnMessage(CaoMessage pICaoMsg)
        {
            try
            {
                // call a method of another thread
                SetTextCallback SetMsg = new SetTextCallback(SetText);
                Invoke(SetMsg, pICaoMsg.Value.ToString());
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        // OnMessage processing function
        private void SetText(string msg)
        {
            textBox1.Text = msg;
        }

    }
```

```
            }
```

2.6.4. Client development with Delphi

2.6.4.1. Making preparation

To operate the object of CAO with Delphi, create a unit file from the type library, and use the file.

Creation and use the unit in the following procedures.

(1) Select “Project → read type library” menu.

(2) From the displayed library, select "CAO1.0 type library", and crick “Unit creation” button. (Figure2-11) The unit file is created at the specified directory with file name “CAOLib_TLB.pas".

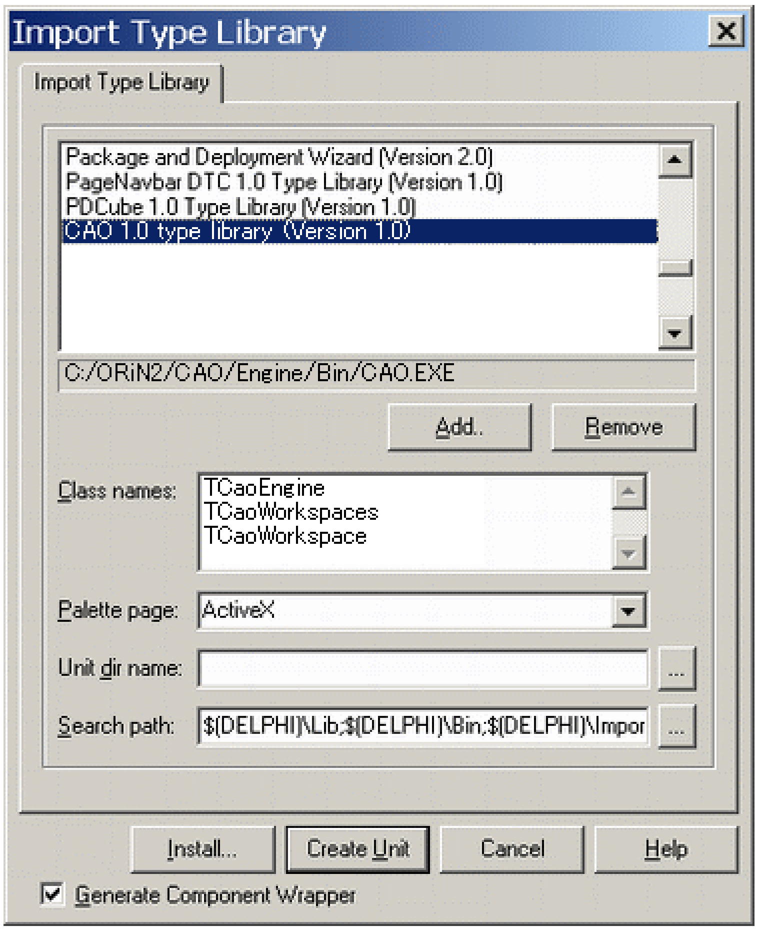

Import Type Library
Import Type Library
Package and Deployment Wizard (Version 2.0)
PageNavbar DTC 1.0 Type Library (Version 1.0)
PDCube 1.0 Type Library (Version 1.0)
CAO 1.0 type library (Version 1.0)
C:/ORiN2/CAO/Engine/Bin/CAO.EXE
Add..
Remove
Class names:
TCaoEngine
TCaoWorkspaces
TCaoWorkspace
Palette page:
ActiveX
Unit dir name:
Search path:
$(DELPHI)\Lib;$(DELPHI)\Bin;$(DELPHI)\Impor
Install...
Create Unit
Cancel
Help
Generate Component Wrapper

Figure2-11 Read type library screen

(3) Activate the source file that uses the CAO object, and select menu [File] -> [Use unit].

(4) Select ”CAOLib_TLB” on the unit selection screen and click OK button.

(5) Add “ComObj" in the uses paragraph of the source file.

CAO object will become available with these procedures.

2.6.4.2. Operation procedure

CAO object is operated in the following procedures.

(1) Create CaoEngine object

Retrieve UUID with ProgIDToClassID(), and create object with CreateComObject().

```
CaoObj := CreateComObject(ProgIDToClassID('CAO.CaoEngine'));
```

(2) Retrieve CaoEngine interface

Because the object generated in (1) has IUknown type interface, cast the object to the CaoEngine interface.

```
CaoEng := CaoObj as ICaoEngine;
```

(3) Operate with object

Use CaoEngine interface to operate CAO.

```
CaoWss := CaoEng.Workspaces;
```

As a note, when CAO object variable is released, Release() is automatically executed. Therefore, if variable is released explicitly as in the other language, CAO may not end normally.

Following is a sample to set and get the value of a variable.

List 2-7 Unit1.pas

```
unit Unit1;

interface

uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls, ComObj;

type
 TForm1 = class(TForm)
 Button1: TButton;
 Edit1: TEdit;
 Edit2: TEdit;
 Label1: TLabel;
 Label2: TLabel;
 procedure Button1Click(Sender: TObject);
 private
{ Private declaration }
 public
{ Public declaration }
 end;

var
 Form1: TForm1;
```

```
implementation

uses CAOLib_TLB;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
 CaoObj: IUnknown;
 CaoEng: ICaoEngine;
 CaoCtrl: ICaoController;
 CaoVar: ICaoVariable;

begin
 Try
{ create CaoEngine }
 CaoObj := CreateComObject(ProgIDToClassID('CAO.CaoEngine'));
 CaoEng := CaoObj as ICaoEngine;

{ create CaoController }
 CaoCtrl := CaoEng.Workspaces.Item(0).AddController('', 'CaoProv.DataStore', '', '');

{ create CaoVariable }
 CaoVar := CaoCtrl.AddVariable('Test', '');

{ set value }
 CaoVar.Value := Edit1.text;

{ get value }
 Edit2.Text := CaoVar.Value;
 Except
{ error processing }
 ShowMessage('Error');
 End;
end;

end.
```

2.6.5. Creating client in Java

It is impossible to call by COM from Java. Therefore, to use CAO from Java, there are the following methods.

・ JNI native method library.[11]

・ Java-COM bridge library.

[11] CaoSQL provides a library that wraps interface for Java.

2.6.5.1. CAO call from JNI's native method library

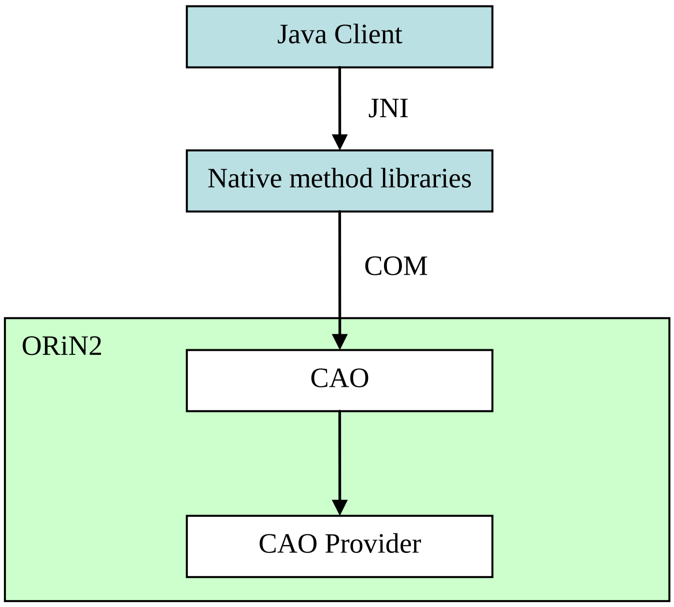


Figure 2-12 Calling CAO by Java

The following shows the creation procedure of a native method library and a sample client of Java. Words in brackets in the end of each procedure indicates a language used.

(1) Declaring native method (Java)

Declare a native method that has functions where a client program requires. The native method declared here does not need to be a CAO interface-wrapped method.

To declare a native method, put “native” keyword in the method declaration.

```
Example) public native int AddVariable(String VarName);
```

(2) Creating a client program that uses the native method (Java)

Create a class using declared native method, and then compile it.

To compile, run the following command by using command prompt.

```
Example)  javac CaoJNI.java
```

(3) Creating a C++ header of the native method (Java)

Create a header file that is included from the compiled file by the native code

To create, run the following command by using command prompt.

```
Example)  javah -jni CaoJNI
```

(4) Including JNI header file (C++)

Include a created header file and a JNI header file.

```
Example)  #include <jni.h>
          #include “CaoJNI.h”
```

(5) Implementing a native method library (C++)

Set a CAO object to use by the procedure of 2.6.1.

Implement the native method declared in the created header file.

(6) Compiling the native method library (C++)

Compile the implemented native command library.

To compile, run the following command by the command prompt. For about the compile options, change depending on the environment.

```
Example) cl /IC:¥jdk1.3.1¥include /IC:¥jdk1.3.1¥include¥win32 /IC:¥ORiN2¥CAO¥Include
                                                             /LD CaoJNI.cpp /FeCaoJNI.dll
```

(7) Executing a sample client (Java)

Execute a sample client. A file of the native method library needs to exist in the folder written by an environmental variable “CLASSPATH”.

To execute the client, run the following command.

```
Example)  java CaoJNI
```

The following sample source shows a source code created by the procedure written above.

List 2-8 **CaoJNI.java**

```
import java.io.*;

public class CaoJNI {

        /* --- Cao Native Interfases --- */
        public native int CreateCaoEngine();
        public native int AddController(String CtrlName, String ProvName, String Machine, String
Param);
        public native int AddVariable(String VarName);
        public native String GetValue();
        public native void PutValue(String strVal);
        public native void FreeMemory();

        // Constructor
        public void CaoJNI() {
        }

        // Initializing CaoJNI class
        public void init() {
                System.loadLibrary("CaoJNI");
        }

        public static void main (String args[]) {

                CaoJNI cao= new CaoJNI();
                cao.init();

                int iRet;
                // Creating CaoEngine
                iRet = cao.CreateCaoEngine();
                if (iRet != 0) {
                        System.out.println("Can't make CaoEngine!");
                        return;
                }

                iRet = cao.AddController("Sample", "CaoProv.DataStore", "", "");
                if (iRet != 0) {
                        System.out.println("Error on OpenController: ErrorCode(" + 
Integer.toHexString(iRet) + ")");
                        cao.FreeMemory();
                        return;
                }

                iRet = cao.AddVariable("S1");
                if (iRet != 0) {
                        System.out.println("Error on GetVariable: ErrorCode(" + 
Integer.toHexString(iRet) + ")");
                        cao.FreeMemory();
                        return;
                }

                cao.PutValue("Sample Data");
                System.out.println("PutValue Succeeded");

                String strRet = cao.GetValue();
                System.out.println(strRet);

                cao.FreeMemory();
        }
}
```

List 2-9 **CaoJNI.cpp**

```
#include <jni.h>
#include "CaoJNI.h"
#include <atlbase.h>

#include "CAO.h"
#include "CAO_i.c"

BSTR jstring2BSTR(JNIEnv *env, jstring jstr);

ICaoEngine* g_pICaoEng;
ICaoController* g_pICaoCtrl;
ICaoVariable* g_pICaoVar;

// CreateCaoEngine
JNIEXPORT jint JNICALL Java_CaoJNI_CreateCaoEngine
  (JNIEnv *env, jobject me)
{
        HRESULT hr;
        hr = CoInitialize( NULL );

        /* Create CaoEngine Object */
        hr = CoCreateInstance(CLSID_CaoEngine, NULL, CLSCTX_LOCAL_SERVER, IID_ICaoEngine, 
(void**)&g_pICaoEng);
        if (SUCCEEDED(hr)) {
                printf("Make CaoEngine Succeeded!¥n");
        }

        return hr;
}
```

```
// AddController
JNIEXPORT jint JNICALL Java_CaoJNI_AddController
  (JNIEnv *env, jobject me, jstring strCtrl, jstring strProv, jstring strMachine, jstring
strParam)
{
       if (g_pICaoCtrl) {
               return E_FAIL;              // Controller has already opened
       }

       HRESULT hr;

       CComBSTR bstrProv, bstrMachine, bstrCtrl, bstrParam;
       bstrProv = jstring2BSTR(env, strProv);
       bstrMachine = jstring2BSTR(env, strMachine);
       bstrCtrl = jstring2BSTR(env, strCtrl);
       bstrParam = jstring2BSTR(env, strParam);

       /* Get CaoWorkspace Object */
       CComPtr<ICaoWorkspaces> pICaoWSs;
       CComPtr<ICaoWorkspace> pICaoWS;
       hr = g_pICaoEng->get_Workspaces(&pICaoWSs);
       if (FAILED(hr)) {
               return hr;
       }
       hr = pICaoWSs->Item(CComVariant(0L), &pICaoWS);
       if (FAILED(hr)) {
               return hr;
       }

       /* Create CaoController Object */
       hr = pICaoWS->AddController(bstrCtrl, bstrProv, bstrMachine, bstrParam, &g_pICaoCtrl);
       if (SUCCEEDED(hr)) {
               printf("OpenController Succeeded!¥n");
       }

       return hr;
}

// AddVariable
JNIEXPORT jint JNICALL Java_CaoJNI_AddVariable
  (JNIEnv *env , jobject me, jstring strVar)
{
       if (g_pICaoVar) {
               g_pICaoVar->Release();
       }

       HRESULT hr;

       CComBSTR bstrVar, bstrOption;
       bstrVar = jstring2BSTR(env, strVar);
       hr = g_pICaoCtrl->AddVariable(bstrVar, bstrOption, &g_pICaoVar);
       SysFreeString(bstrVar);

       if (SUCCEEDED(hr)) {
               printf("GetVariable Succeeded!¥n");
       }

       return hr;
}

// GetValue
JNIEXPORT jstring JNICALL Java_CaoJNI_GetValue
  (JNIEnv *env, jobject me)
{
       if (!g_pICaoVar) {
```

```
                        return NULL;  // Variable is not found
                }

                // Obtaining a value
                CComVariant vntValue;
                HRESULT hr = g_pICaoVar->get_Value(&vntValue);
                if (FAILED(hr)) {
                        return NULL;
                }

                // Convert obtained value to character string
                hr = vntValue.ChangeType(VT_BSTR);
                if (FAILED(hr)) {
                        return NULL;
                }

                return env->NewString(vntValue.bstrVal, SysStringLen(vntValue.bstrVal));
        }

        // PutValue
        JNIEXPORT void JNICALL Java_CaoJNI_PutValue
          (JNIEnv *env, jobject me, jstring strVal)
        {
                if (!g_pICaoVar) {
                        return;                        // Variable is not found
                }

                HRESULT hr;
                CComBSTR bstrVal;
                bstrVal = jstring2BSTR(env, strVal);

                hr = g_pICaoVar->put_Value(CComVariant(bstrVal));
                return;
        }

        // FreeMemory
        JNIEXPORT void JNICALL Java_CaoJNI_FreeMemory
          (JNIEnv *env , jobject me)
        {
                if (g_pICaoVar) {
                        g_pICaoVar->Release();
                        printf("CaoVariable Deleted¥n");
                }
                if (g_pICaoCtrl) {
                        g_pICaoCtrl->Release();
                        printf("CaoController Closed¥n");
                }
                if (g_pICaoEng) {
                        g_pICaoEng->Release();
                        printf("CaoEngine Terminated¥n");
                }

                CoUninitialize();

                return;
        }

        // Character string conversion(Java→BSTR)
        BSTR jstring2BSTR(JNIEnv *env, jstring jstr)
        {

            if (jstr==NULL) {
                        return NULL;
                }

            const WCHAR* wc;
            wc = env->GetStringChars(jstr, NULL);
         int iLen;
            iLen = env->GetStringLength(jstr);
            BSTR bstrString;
            bstrString = SysAllocStringLen(wc, iLen);

            env->ReleaseStringChars(jstr, wc);

         return bstrString;

    }
```

2.6.5.2. Call CAO with Java-COM Bridge

You can call CAO by using Java-COM bridge.

Java-COM bridge is a library for calling COM from Java programs, and several kinds are provided as OSS. (Example: j-interop, COM 4 J etc.)

CAO is created using COM, so you can use CAO directly from Java by using Java-COM bridge.

Since the specific usage depends on the library, refer to the information of each library.

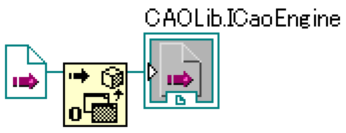
2.6.6. Creating client in LabVIEW

To operate CAO object from LabVIEW, create a VI file functioning as ActiveX client.

Following is the procedure to create and use ActiveX client VI.

Note: LabVIEW ver8.6 is used for window images of the following procedure.

(1) With automation open function, open reference to server.



At automation Refnum input, select [Select ActiveX class]-[Reference], and choose “CAO1.0 Type Library Version 1.0”, and specify [CaoEngine].

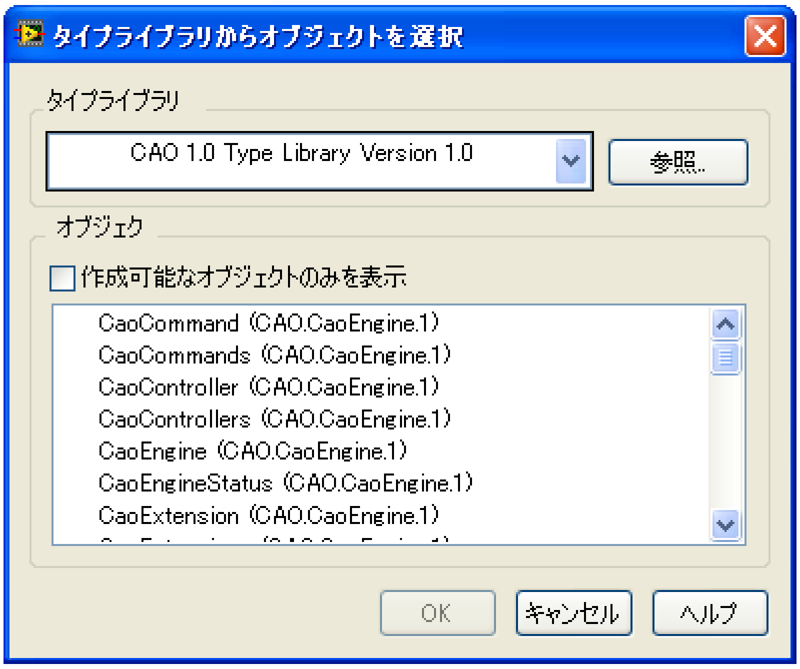
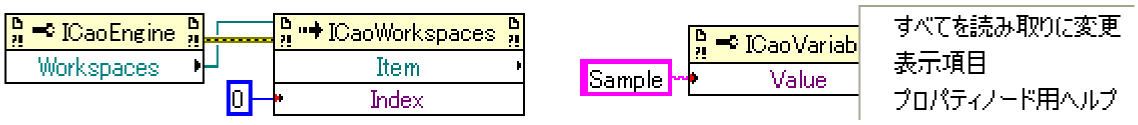

タイプライブラリからオブジェクトを選択
タイプライブラリ
CAO 1.0 Type Library Version 1.0
参照...
オブジェクト
作成可能なオブジェクトのみを表示
CaoCommand (CAO.CaoEngine.1)
CaoCommands (CAO.CaoEngine.1)
CaoController (CAO.CaoEngine.1)
CaoControllers (CAO.CaoEngine.1)
CaoEngine (CAO.CaoEngine.1)
CaoEngineStatus (CAO.CaoEngine.1)
CaoExtension (CAO.CaoEngine.1)
OK
キャンセル
ヘルプ

Figure 2-13 Window to select object from type library

(2) Setup properties, and use methods.

Note: To use properties with the capability of “Read” and “Write”, change the attribute of the property node.

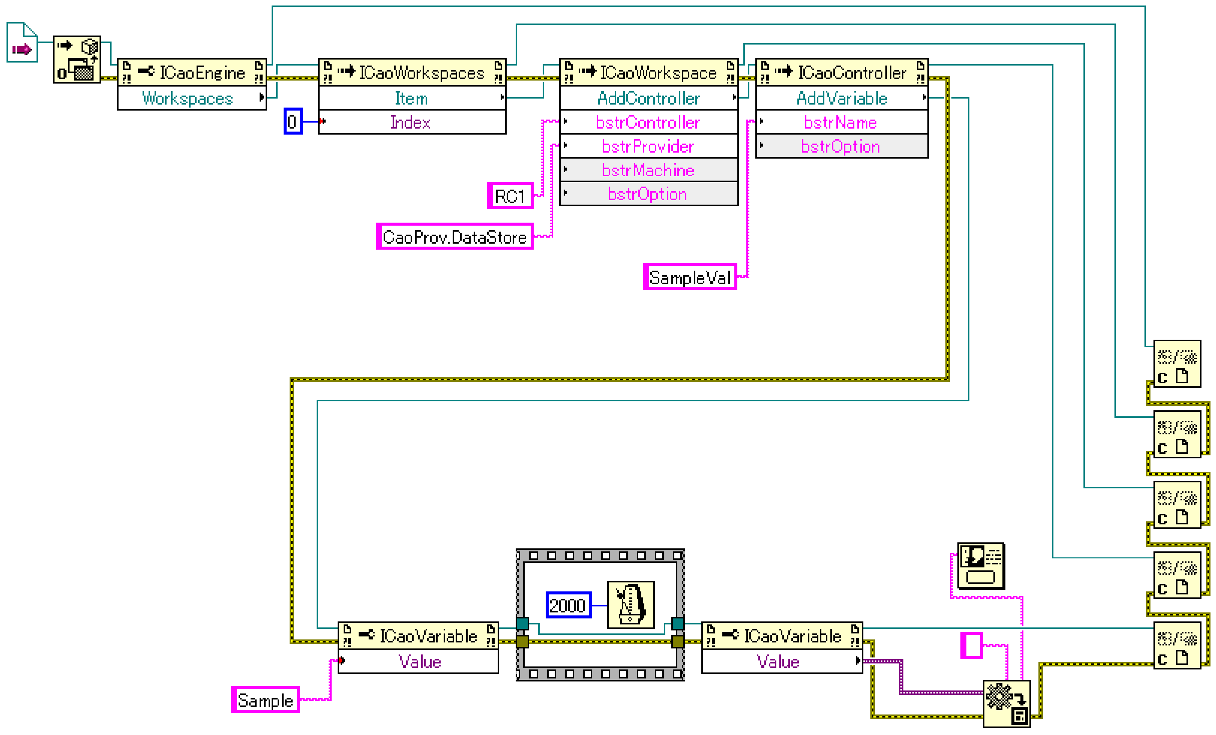


(3) Close reference

Close reference from a function that closes automation Refnum.

Below is an example of setting and getting variable value.

List 2-8	ImportSample.vi



2.7. Special functions of CAO

2.7.1. CRD switch function

According to the setting of providers, CAO engine can access to the CRD file when provider is accessed to get properties. Following is the provider set-up steps to use CRD switch function.

(1) Install CRD provider to the machine to run CAO engine.

(2) Set path to the CRD file to be referred in the registry information “CRDFile” of the provider that uses CRD switch function. CaoConfig can set this pass information. Please refer details in 6.

(3) In property that refers to the CRD file of the provider, return the value “E_CRDIMPL"[12].

Please not that object name of created controller needs to be as same as the CRD file.

2.7.2. Automatic object registration function

The function automatically register object to default workspace when CAO engine starts.

The automatically registered object has the same structure as the CRD file registered in the registry. If the registered file path is wrong, only the registerable object is registered.

CaoConfig can be used to register the path to the registry.

[12] When “E_CRDIMPL" is returned by the method, the value is returned to the client as it is.

It is convenient to use the XML editor to edit the CRD file. Following is one of free XML editor software.

Morphon XML-Editor 3.1.4: http://www.morphon.com/

Following is a sample of CRD file.

List 2-10 CRDSample.xml

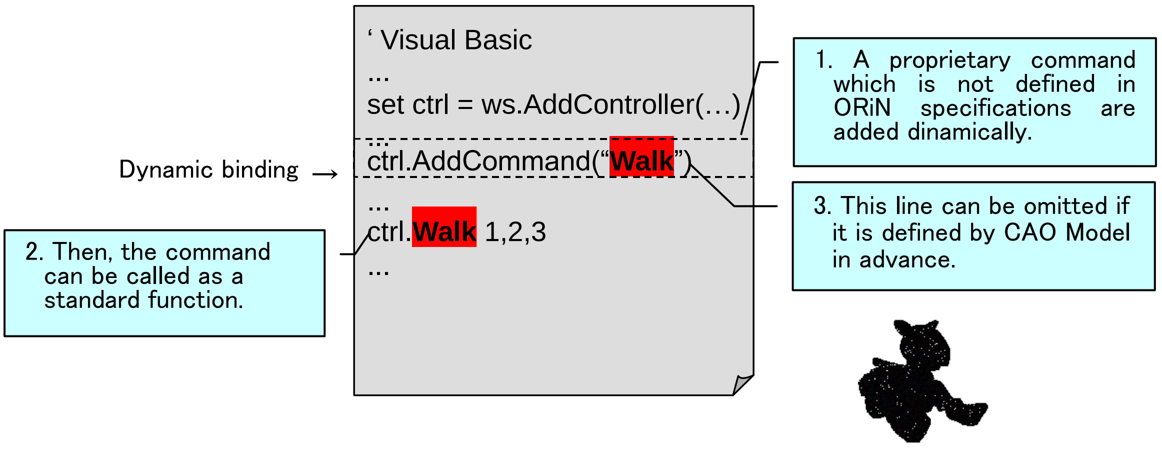
```
<?xml version="1.0" encoding="Shift_JIS"?>
<CRD xsi:schemaLocation="http://www.orin.jp/CRD/CRDSchema CRDSchema.xsd"
       xmlns="http://www.orin.jp/CRD/CRDSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

       <Help>CRD Sample Data</Help>
       <Version>1.0.0</Version>
       <Controller name="RC1" provider="CaoProv.XXX" machine="" option="Conn=eth:192.168.1.1">
                <Command name="Walk"/>
                <Variable name="Var1"/>
       </Controller>
</CRD>
```

2.7.3. Dynamic method addition function

The function dynamically adds command class as methods of controller class.

The controller class can execute the Execute method by specifying command name as method name.



' Visual Basic
...
set ctrl = ws.AddController(…)
...
ctrl.AddCommand("Walk")
...
ctrl.Walk 1,2,3
...
Dynamic binding →
1. A proprietary command which is not defined in ORiN specifications are added dinamically.
2. Then, the command can be called as a standard function.
3. This line can be omitted if it is defined by CAO Model in advance.

Figure2-14 Dynamic Binding

Following is a sample of dynamic method addition function.

```
'Create command
 Dim Cmd As CaoCommand
 Set Cmd = Ctrl.AddCommand("Shoot")

'Sample of executing in the command class
 Cmd.Parameters = Array(10,10,20)
 Cmd.Execute 0
```

```
'Sample of using dynamic method addition function
 Ctrl.Shoot 10, 10, 20
```

3. CRD programming guide

3.1. Outline

CRD (Controller Resource Definition) is a standard to describe various static resources of the FA device with XML (eXtensible Markup Language). Its data schema is called CRD schema, and a XML file described according to this CRD schema is called as CRD instance. CRD is a standard data schema to manage FA device information in a common format.

CRD is designed assuming following two usages.

(1) Device static data definition

(2) Device and whole system configuration definition

(3) Device capability definition

The first half of this chapter explains XML as basic knowledge, and the latter half explains concrete procedure of creating CRD.

3.2. Basic knowledge

3.2.1. What is XML?

XML is self-extensible standard generalized markup language (eXtensible Markup Language), and the manufacturer of the document can decide the rule to decide the document structure. (Therefore, it is named as “Self-extensible".) By utilizing the function that the creator of document can freely define its structure, highly flexible data, or data independent from specific document structure, can be widely exchanged using XML standard.

XML has the following features.

・ Data format has its meaning and content.
・ A hierarchical data structure definition is possible.
・ Tag can be logically described, and structured comprehensive document can be created.
・ Data format is extensible, and widely spreading as an industry standard.

Following is an simple example of XML document. In XML, data is expressed by using tag. Items from the beginning tag to the end tag are called as element. In the following example, tags like <ORiN> and <Author> is used to hold the information of ORiN or its author. The element can have another element as a child element, to express data structurally. In addition, XML can add expression like “Release=”2003” ”, to give additional information (attribute) to the elements.

```
<ORiN>
      <ORiN1 Release=“2003″>
              <Author>ORiN conference </Author>
              <EngineName>RAO</EngineName>
      </ORiN 1>
```

```
                <ORiN2 Release=“2005″>
                        <Author>ORiN conference </Author>
                        <EngineName>RAO</EngineName>
                </ORiN 2>
        </ORiN>
```

3.2.2. DOM

DOM (Document Object Model) is standard API to treat the XML document as an object.

XML document itself is a text file. However, when browsers or applications like CAO processed the file, they read out each elements described in XML as objects. If each applications differently handles XML object, it is very burdensome for developers. Therefore, W3C established DOM as a unified standard. Another XMP API other than DOM is SAX (Simple API for XML), and both of them are widely used.

DOM has several levels, and the higher level is the newer and more functional. Level-2 is the latest recommendation from W3C, and CRD provider also uses this level.

3.2.3. XML parser

The XML document is usually a text file. However, XML allows various ways of expression, and applications need to follow several procedures to read the file. General-purpose procedures are selected and implemented ad XML parser.

When the XML parser is called from an application, standard API is used. Standard API includes DOM and SAX, etc., which are explained before. DOM is an open standard, and several XML parsers compliant to the standard is released. On Windows environment, MSXML can be easily used with Visual C++ and Visual Basic, and MSXML is used in CRD provider etc. Several versions of MSXML has been released, and CRD provider uses MSXML4.0. When the CRD provider is used, you need to confirm the version of installed MSXML.

3.2.4. XML schema

The schema in XML is description of possible XML document structure. The schemes describes correct order of elements and attribute arrays. By describing the schema, XML parser can automatically check the correctness of XML document to some degree.

Typical XML Schema includes DTD, XML Schema, RELAX and XDR, and CRD uses XML Schema.

3.3. CRD file and CRD provider

Client applications that utilize CRD file mainly uses following two methods to access the file.

(1) Indirectly access through CAO engine.
(2) Direct access with XML parser.

Please refer a suitable reference book about method (2). For method (1), ORiN2 prepares CRD provider so that client application can access CRD file just as same way as accessing controller. By using this provider, application program can handle both data in actual controller and data expressed in CRD file in the same way.

CRD schema defines each element almost corresponds one-to-one to CAO interface. For example, when accessing a CRD file named “test.xml”, which defines “RC1” controller element and “Var1” variable element in the controller, CRD provider can access variable element “I4” in the CRD file in the following way.

```
Dim caoEng As New CaoEngine
Dim caoWS As CaoWorkspace
Dim caoCtrl As CaoController
Dim caoVar As CaoVariable

Set caoWS = CaoEng.CaoWorkspaces(0).
Set caoCtrl = caoWS.AddController("RC1", "CaoProv.CRD", "", "C:¥test.xml")
Set caoVar = caoCtrl.AddtVariable("Var1")
```

After CRD file is created, please try to use CRD provider to check and to change its design.

However, CRD provider can change data that is described as “W” in the R/W field of A.1 CAO engine function list.

For details of provider, please refer to “CRD provider users’ guide.”

3.4. Creating CRD file

3.4.1. Header and root element creation

CRD file is a XML file, and following description is placed at the top of the file for XML declaration.

```
<?xml version="1.0" encoding="Shift_JIS"?>
```

In the above declaration, encoding attribute need to match the character code 12 that is used in CRD file.

Next, specify the following as a root element of CRD document.

```
<CRD xmlns=http://www.orin.jp/CRD/CRDSchema
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:schemaLocation="http://www.orin.jp/CRD/CRDSchema CRDSchema.xsd">
```

In the element, specify the path to the scheme at the list line before "CRDSchema.xsd." If the schema does not exist at the specified path, CRD file structure is not checked.

In addition, the CRD root element can have following child elements.

3.4.2. Static data definition

Static data definition is to describe object elements and property elements of controller in XML file.

(1) Add object elements

Object element represents CAO object instance. Controller element is allocated below CRD root element, and objects below controller element are allocated as the same structure as CAO object tree. Following is an example of object element description.

```
<Controller name="RC1">
       <File name="pro1.pac">
                 …
       </File>
       <Variable name="I1">
                 …
       </Varriable>
</Controller>
```

All object elements require name element, the name element represents object name. In the above example, CaoController"RC1" has a CaoFile "pro1.pac" and CaoVariable "I1"

(2) Add property element to object element.

Add property element to describe static data of each object under object element. The property element depends on the type of object element. Please refer <ORiN2.1 Specifications Part 3: CRD> for details.

3.4.3. System configuration definition

System configuration definition is to add object elements of controller in XML file. Same as static data element, object element is allocated under CRD root element in tree structure.

Following is an example description.

```
<Controller name="RC5" option="Conn=eth:10.8.109.126" provider="CaoProv.DENSO.NetwoRC">
    <Robot name="Arm1"/>
    <Variable name="IO100"/>
</Controller>
```

The table below shows attributes of each object element.

Table 3-1 Object element and attribute

object element	required attribute	optional attribute
Controller	Name Provider	Machine Option
Command	Name	Option
Extension		
File		
Robot		
Task		
Variable		
Message	Number	-

3.4.4. Device capability definition

Device capability definition is to add object element and capability definition element to XML file.

(1) Add object element

Object element is allocated under CRD root element in tree structure, like static data definition. All object elements may have name attribute and key attribute optionally.

Name attribute shows object name, and an object without name attribute is regarded as an object without name specification.

The capability of an object with key attribute is recognized as the capability defined in the capability definition element with the same key name. An object without key attribute is regarded as using the default capability definition element.

Following is an example of object element.

```
<Controller>
  <File/>
  <Robot/>
  <Task/>
  <Variable name="@CURRENT_TIME" key="SYS_VAR_CURRENT_TIME"/>
</Controller>
```

The above example represents the following.

・ All controllers and the files, robots, task objects under the controllers use the default capability definition element.

・“@CURRENT_TIME” variable objects under all controllers refer a capability definition element with a key named “SYS_VAR_CURRENT_TIME”.

(2) Add capability element.

Capability definition element referred by each object is allocated under CRD root element. Capability element definition specifies information like implementation status of controller member or data type / value range of argument / return value.

Capability elements are described in the controller capability element. In this way, each controller can have the controller specific capability definition.

Each capability definition element can have ObjectKey element. This is to uniquely determine a capability specified by object element. Capability element without ObjectKey element is regarded as defining default capability.

Following is a capability definition example.

```
<Controller_Info>
  <Args>
    <Arg index="1">
      <HelpString>Controller Name</HelpString>
      <VarInfo>
        <DataInfo>
          <Min>0</Min>
          <Max>10</Max>
        </DataInfo>
      </VarInfo>
    </Arg>
    <Arg index="2"/>
  </Args>
  <Variable_Info>
    <ObjectKey>SYS_VAR_CURRENT_TIME</ObjectKey>
    <Put_Value_Info>
      <Args>
        <Arg index="1">
          <VarInfo type="VT_DATE"/>
        </Arg>
      </Args>
    </Put_Value_Info>
    <Get_Value_Info>
      <Result>
        <VarInfo type="VT_DATE"/>
      </Result>
    </Get_Value_Info>
  </Variable_Info>
</Controller_Info>
```

The above example explains the followings.

・The number of argument of AddController for default controller is two.

・The first argument of AddController for default controller is the controller name.

・The range of the first argument of AddController for default controller is 1 to 10.

・For object in the default controller, Put_Value() and Get_Value() is implemented for variables with definition of Key = “SYS_VAR_CURRENT_TIME”.

・ In the above mentioned variable, Put_Value() sets VT_DATA type value.
・ In the above mentioned variable, Get_Value() acquires VT_DATA type value.

3.5. Sample CRD file

For better understanding, three types of CRD usage examples are shown. These examples may be combined into one file, but these definitions are usually divided into several files according to the usage.

3.5.1. Static data definition example

List 3-1 **CRDSchema2.xsd**

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!--
    ********** [ORiN2] CRD Sample **********
-->
<CRD xsi:schemaLocation="http://www.orin.jp/CRD/CRDSchema ../../Schema/CRDSchema2.xsd"
     xmlns=http://www.orin.jp/CRD/CRDSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Help>CRD Test Data</Help>
  <Version>1.0.0</Version>
  <Controller name="RC1">
    <Attribute>0</Attribute>
    <Help>Sample Ctrl</Help>

    <!—In case of VT_I2 type -->
    <Variable name="Var1">
      <Value type="VT_I2">
        <iVal>10</iVal>
      </Value>
    </Variable>

    <!—In case of VT_I2 type one dimensional array -->
    <Variable name="Var2">
      <Value type="VT_ARRAY">
        <array type="VT_I2">
          <!—Array information -->
          <dimension>1</dimension>
          <arrayBound>
            <lBound>0</lBound>
            <elements>3</elements>
          </arrayBound>
          <!-- Data -->
          <arrayData>
            <iVal>0</iVal>
            <iVal>1</iVal>
            <iVal>2</iVal>
          </arrayData>
        </array>
      </Value>
    </Variable>
  </Controller>
</CRD>
```

3.5.2. System configuration definition example

List 3-2 **CRDSchema2.xsd**

```
<?xml version="1.0" encoding="UTF-8"?>
<CRD xmlns=http://www.orin.jp/CRD/CRDSchema
     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
     xsi:schemaLocation=" http://www.orin.jp/CRD/CRDSchema  ../../Schema/CRDSchema2.xsd">

  <Controller name="RC5" option="Conn=eth:10.8.109.126" provider="CaoProv.DENSO.NetwoRC">
    <Robot name="Arm1"/>
    <Variable name="IO100"/>
  </Controller>
</CRD>
```

3.5.3. Device capability definition example

List 3-3 **CRDSample_DC.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<CRD xsi:schemaLocation=http://www.orin.jp/CRD/CRDSchema ../../Schema/CRDSchema2.xsd
     xmlns=http://www.orin.jp/CRD/CRDSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Controller_Info>
    <Args>
      <Arg index="1">
        <HelpString>Controller name</HelpString>
        <VarInfo>
          <DataInfo>
            <Min>0</Min>
            <Max>10</Max>
          </DataInfo>
        </VarInfo>
      </Arg>
      <Arg index="2"/>
    </Args>
    <Get_VariableNames_Info>
      <Result>
        <VarInfo type="VT_ARRAY|VT_VARIANT"/>
      </Result>
    </Get_VariableNames_Info>
    <Execute_Info>
      <Args>
        <Arg index="1">
          <VarInfo>
            <DataInfo>
              <List>getAutoMode</List>
              <List>putAutoMode</List>
            </DataInfo>
          </VarInfo>
        </Arg>
        <Arg index="2">
          <VarInfo type="VT_EMPTY"/>
          <VarInfo type="VT_I2">
            <DataInfo>
              <List>10</List>
              <List>20</List>
            </DataInfo>
          </VarInfo>
          <VarInfo type="VT_I4">
            <DataInfo>
              <Min>-1</Min>
              <Max>100</Max>
            </DataInfo>
          </VarInfo>
        </Arg>
      </Args>
      <Result>
        <VarInfo type="VT_EMPTY"/>
        <VarInfo type="VT_I2"/>
        <VarInfo type="VT_I4"/>
      </Result>
    </Execute_Info>

    <File_Info>
      <Get_Value_Info>
        <Result>
          <VarInfo type="VT_BSTR"/>
          <VarInfo type="VT_ARRAY|VT_UI1"/>
        </Result>
      </Get_Value_Info>
      <Get_VariableNames_Info>
        <Result>
          <VarInfo type="VT_ARRAY|VT_VARIANT"/>
        </Result>
      </Get_VariableNames_Info>
      <Copy_Info/>
      <Delete_Info/>
      <Move_Info/>
    </File_Info>

    <Robot_Info>
      <Halt_Info/>
      <Move_Info/>
      <Speed_Info/>
    </Robot_Info>

    <Task_Info>
      <Start_Info/>
      <Stop_Info/>
    </Task_Info>

    <Variable_Info/>

    <Message_Info>
      <Clear_Info/>
    </Message_Info>

    <Variable_Info>
      <ObjectKey>SYS_VAR_CURRENT_TIME</ObjectKey>
      <Put_Value_Info>
        <Args>
          <Arg index="1">
            <VarInfo type="VT_DATE"/>
          </Arg>
        </Args>
      </Put_Value_Info>
      <Get_Value_Info>
        <Result>
          <VarInfo type="VT_DATE"/>
        </Result>
      </Get_Value_Info>
    </Variable_Info>

  </Controller_Info>

  <Controller>
    <File/>
    <Robot/>
```

```
        <Task/>
        <Variable name="@CURRENT_TIME" key="SYS_VAR_CURRENT_TIME"/>
      </Controller>
    </CRD>
```

4. CAP programming guide

4.1. Outline

CAP is a communication protocol to access the CAO provider by way of the Internet. In ORiN2 SDK, the distributed object technology by way of the Internet has been achieved by using SOAP.

This chapter actually constructs the Web server, and explains the method of calling the provider using CAP. The composition of this chapter explains SOAP as basic knowledge in the first half, and explains the outline of operation of CAP. It explains the environmental construction method to use CAP in the latter half, and the sample program is made at the end.

4.2. Basic knowledge

4.2.1. SOAP

SOAP is a protocol to call the object on PC remotely by transmitting the message described by the XML format with HTTP.

There are CORBA and DCOM, etc. as a technology that calls the object on PC remotely. However, these technologies should do the setting that permits communicating to a specific port when communicating through the firewall to communicate by using an original port number. Because any security hole of DCOM, such as worm, exists these days, the setting that permits communicating to the port of DCOM is not generally done.

Then, to solve these problems, the technology named SOAP was developed. Because SOAP only specifies the message exchange structure between objects, existing HTTP and SMTP are used for the protocol for the communication. As a result, if HTTP or SMTP can be communicated, using the distributed object technology through the firewall becomes possible by using SOAP.

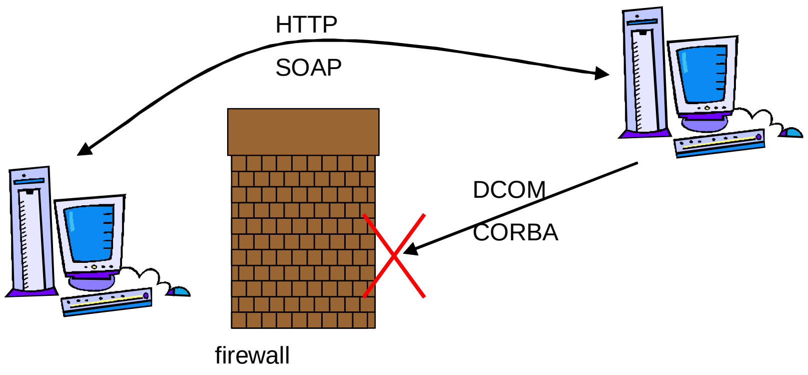


Figure4-1 Image of SOAP

4.3. CAP provider and CAP listener

It provides for the specification named CAP in ORiN2 SDK to access the CAO provider by way of the

Internet. The CAO provider can be remotely accessed by analyzing between the client and the server as the sending and receiving message based on the specification of this CAP.

However, the CAP listener is offered with the CAP provider in ORiN2 SDK because it takes time to make a sending and receiving message and analytical processing based on CAP.

Figure4-2 shows the operation outline of CAP provider and CAP listener. As this figure shows, CAP provider is a provider to generate and to transmit the message to access a remote provider. This message is transmitted to CAP listener to have opened Web server to the public. CAP listener is an interface to receive and analyze CAP-based message, and to call the COM component (CAO provider).

Thus, using the provider remotely just like other local providers become possible by the CAP provider listener's path.

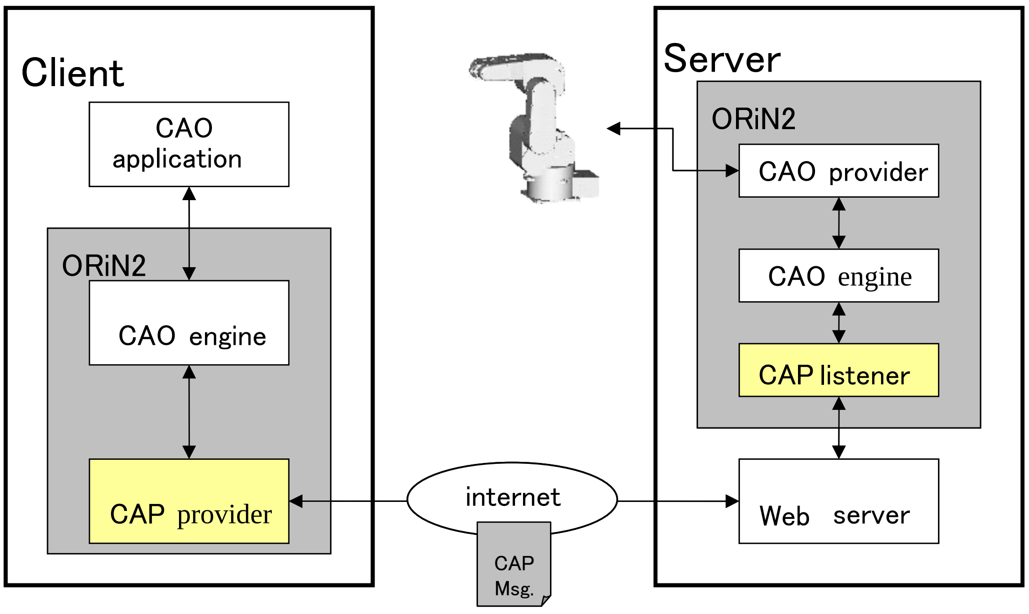


Figure4-2 Outline of CAP

4.4. Environmental construction

This chapter explains the environmental construction to use CAP method.

Hereafter, the side which calls a remote provider by using CAP provider is expressed “Client”, and the called side is expressed "Server".

ORiN2 SDK is assumed to be installed in both the server computer and client computer.

4.4.1. Setting of server side

(1) Installation of Web server

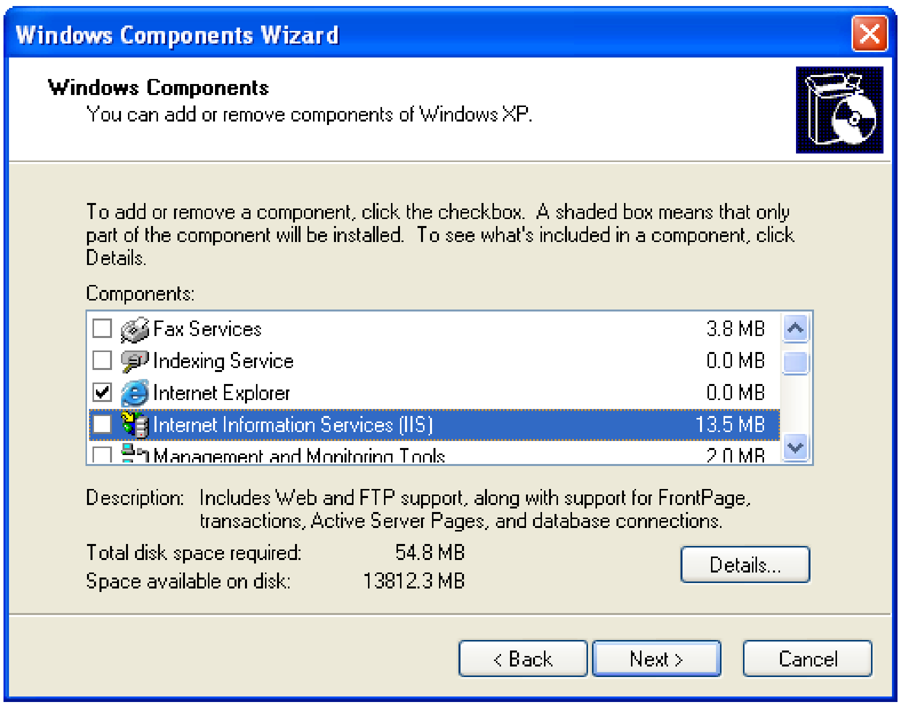
The message transmitted with SOAP as mentioned above starts the CAP listener with the Web server, and analyzes the message. Therefore, it is necessary to construct the Web server.

Web server IIS (Internet Information Service) is appended to Windows 2000 Server/Windows 2000 Professional/Windows XP Professional by the standard[13]. However, you need to install additionally because it is not installed in the default installation. (It is installed by the standard for Windows 2000 Server.)

This section explains the setting procedures with Windows XP Professional.

From the Start menu, click [Setting], click [Control panel], click [Add or Remove Programs], and then select [Add/Remove Windows Components].

Once the dialog box shown in Figure4-3 is displayed, select the checkbox of Internet Information Services (IIS).


Windows Components Wizard

Windows Components
You can add or remove components of Windows XP.

To add or remove a component, click the checkbox. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Components:

☐	Fax Services	3.8 MB
☐	Indexing Service	0.0 MB
☑	Internet Explorer	0.0 MB
☐	Internet Information Services (IIS)	13.5 MB
☐	Management and Monitoring Tools	2.0 MB

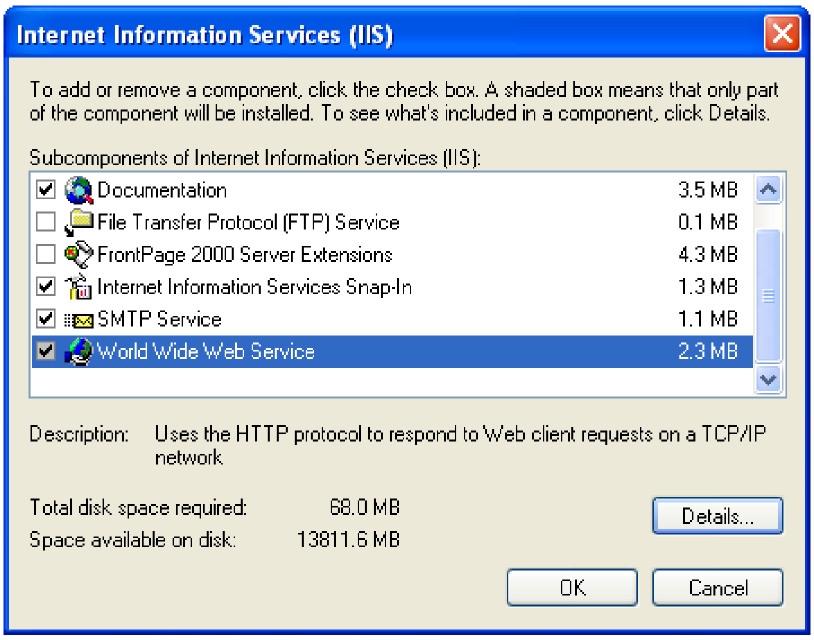
Description: Includes Web and FTP support, along with support for FrontPage, transactions, Active Server Pages, and database connections.

Total disk space required: 54.8 MB
Space available on disk: 13812.3 MB

[13] As for IIS of the Windows 2000 Professional/Windows XP Professional attachment, the function is limited compared with the one of Windows 2000 Server. Please use Windows 2000 Server when you want to provide real service. Moreover, please note that IIS is not attached to Windows XP Home Edition.

Figure4-3 Addition of Windows component

When Figure4-4 is displayed, double-click "World Wide Web Service", and then select the check box of all components. At that time, "Internet Information Service Snap-In" and "Common Component" will be automatically selected.


Internet Information Services (IIS)

To add or remove a component, click the check box. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Subcomponents of Internet Information Services (IIS):

☑	Documentation	3.5 MB
☐	File Transfer Protocol (FTP) Service	0.1 MB
☐	FrontPage 2000 Server Extensions	4.3 MB
☑	Internet Information Services Snap-In	1.3 MB
☑	SMTP Service	1.1 MB
☑	World Wide Web Service	2.3 MB

Description: Uses the HTTP protocol to respond to Web client requests on a TCP/IP network

Total disk space required: 68.0 MB
Space available on disk: 13811.6 MB

Figure4-4 Addition of the Internet information service

Select the components to install, and then click [Next] to start installation of the Windows components.
"Windows XP Professional installation disk" is requested while installing it.

(2) Installation of SOAP Toolkit

Install Microsoft SOAP Toolkit 3.0 in order to analyze the message that has been transmitted with SOAP. Double-click the downloaded soapsdk.exe, and then click the button according to the instruction, the installation is completed. You can leave the setting as the default.

Table 4-1 Soap Toolkit 3.0 download site

File name	URL
soapsdk.exe	http://www.microsoft.com/downloads/details.aspx?

	FamilyID=c943c0dd-ceec-4088-9753-86f052ec8450&DisplayLang=en

(3) Creating virtual directory

A virtual directory is created by using SOAPVDIR.CMD of Microsoft SOAP Toolkit 3.0. A virtual directory here is the one that the address of the HTTP access is allocated in a physical path.

SOAPVDIR.CMD is in the following directory.

< SOAP installation directory > ¥Binaries

To create a directory, enter the following commands in the command prompt.

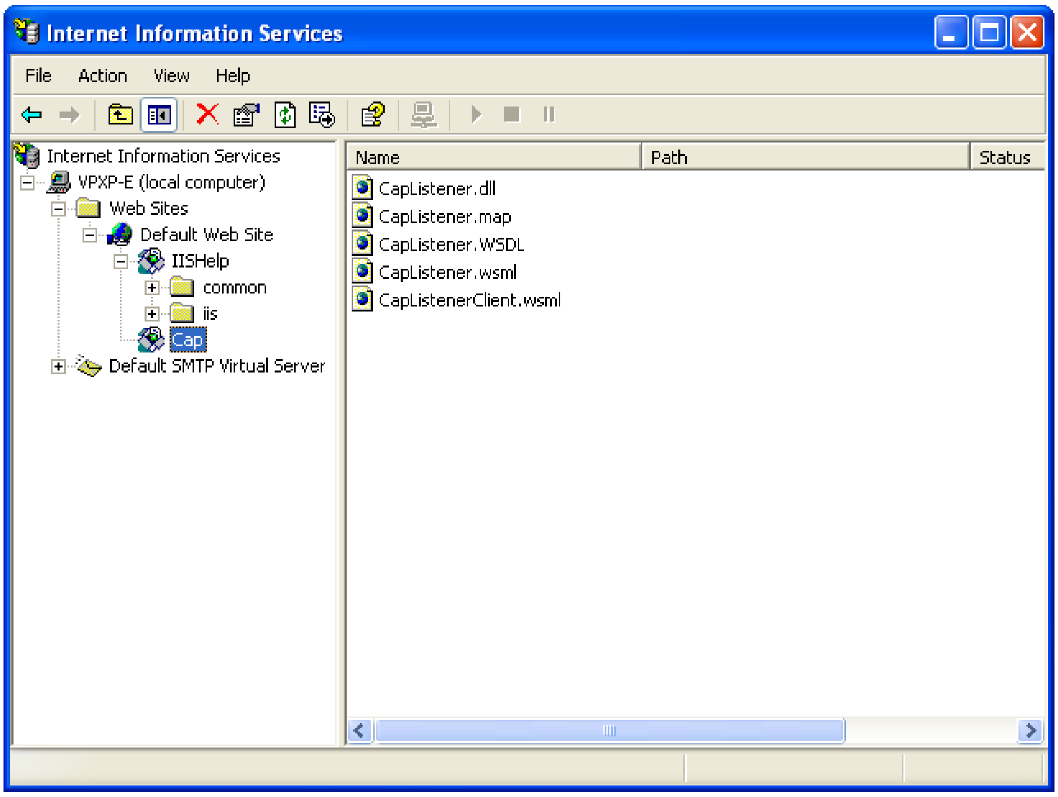
```
SOAPVDIR.CMD CREATE Cap < Bin directory of CapListener >
```

(4) Setting of IIS

Next, perform the settings necessary to start the CAP listener from ISS.

From the Start menu, click [Setting], click [Control panel], click [Administrative Tools], and then start [Internet Information Services].

To confirm the virtual directory [Cap] that you have just created, from the [Internet information service] window, click [Local computer], click [Web site], and then click [Default Web Site].



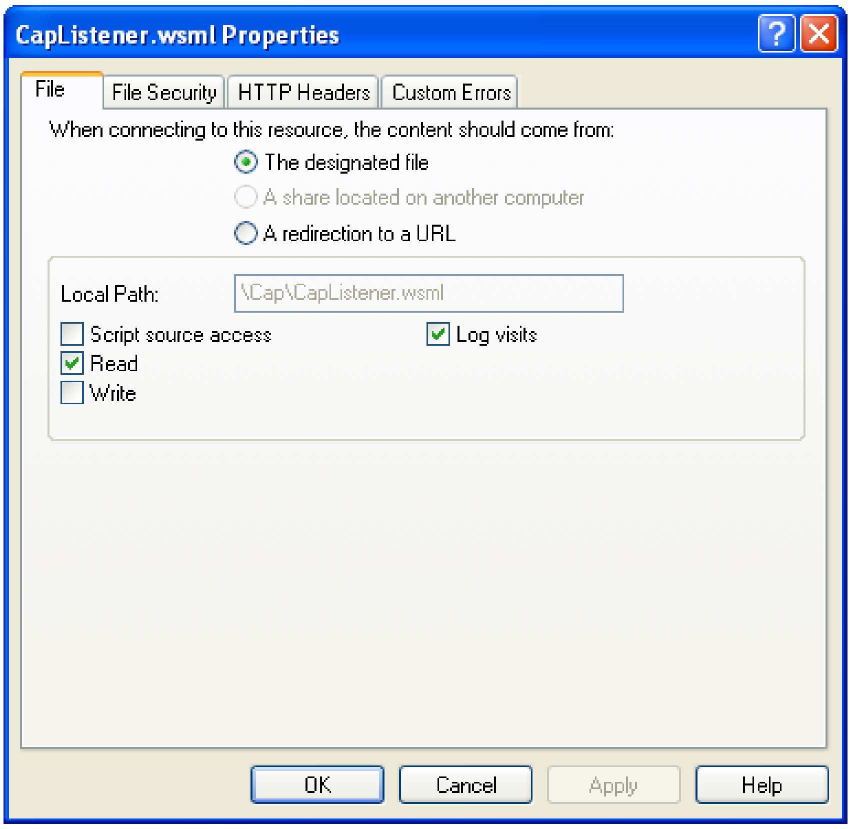
Name	Path	Status
CapListener.dll		
CapListener.map		
CapListener.WSDL		
CapListener.wsml		
CapListenerClient.wsml		

Figure4-5 Set screen of IIS

Right-click in the following two files in the Cap directory to display properties, and select the [Read] checkbox.

・ CapLister.WSML

・ CapListerClient.WSML


CapListener.wsml Properties

When connecting to this resource, the content should come from:

[✓ The designated file]

A share located on another computer

A redirection to a URL

Local Path: \Cap\CapListener.wsml

Script source access

[✓ Read]

Write

[✓ Log visits]

Figure4-6 Set screen of WSML file

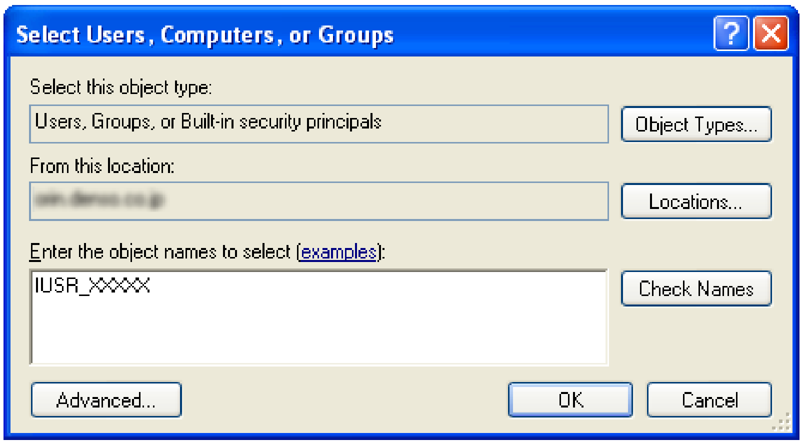
(5) Setting of the start authority of CAO engine

Set the start authority so that an user who accesses via the Internet can start CAO engine.

Right-click in < ORiN installation directory > ¥CAO¥Engine¥Bin¥CAO.EXE, and select [Property], select [Security], and then click [Add…].

Once Figure4-7 is displayed, add an Internet guest account, and then add the start authority.

To check the Internet guest account name, from the Start menu, click [Setting], click [Control panel], and then click [User Accounts].


Select Users, Computers, or Groups

Select this object type:
Users, Groups, or Built-in security principals
Object Types...

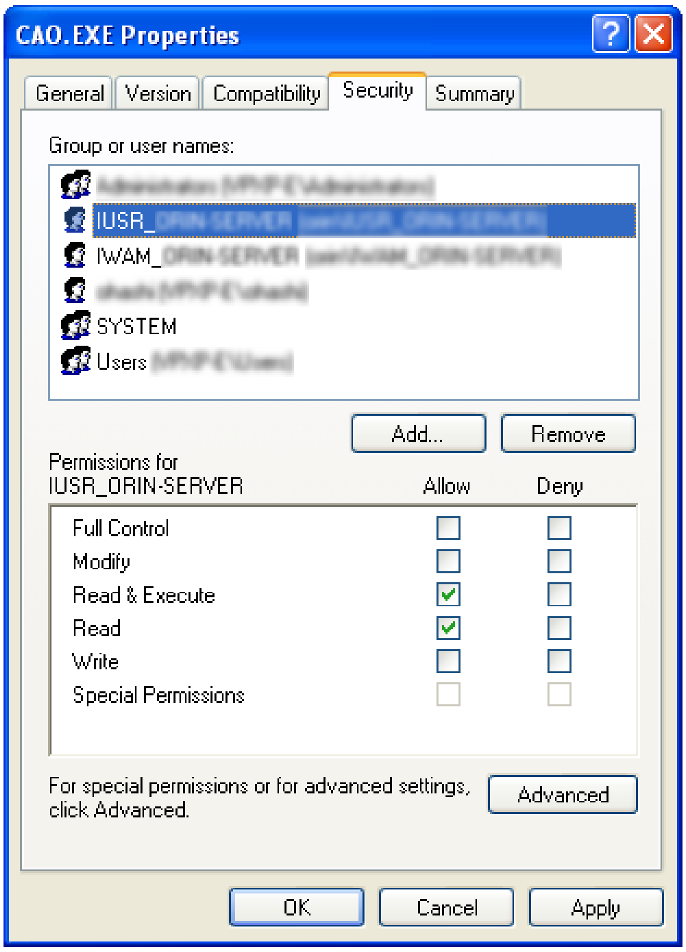
From this location:
[redacted]
Locations...

Enter the object names to select (examples):
IUSR_XXXXX
Check Names

Advanced... OK Cancel

Figure4-7 Addition of the Internet guest account

When the account is added successfully, the account name will be displayed as Figure4-8 shows.


CAO.EXE Properties

General | Version | Compatibility | Security | Summary

Group or user names:
- [redacted]
- IUSR_[redacted]
- IWAM_[redacted]
- [redacted]
- SYSTEM
- Users [redacted]

Add... Remove

Permissions for IUSR_ORIN-SERVER

Permissions for IUSR_ORIN-SERVER	Allow	Deny
Full Control	☐	☐
Modify	☐	☐
Read & Execute	✓	☐
Read	✓	☐
Write	☐	☐
Special Permissions	☐	☐

For special permissions or for advanced settings, click Advanced.
Advanced

OK Cancel Apply

Figure4-8 The Internet guest account addition result

(6) Setting of CapListner.WSDL file

Change the server name written in the line 3459 of the CapListner.WSDL file which is in the CapListener directory to the actual Web server name.

(Example) soap:address location='http://HOSTNAME/Cap/CapListener.WSDL' />

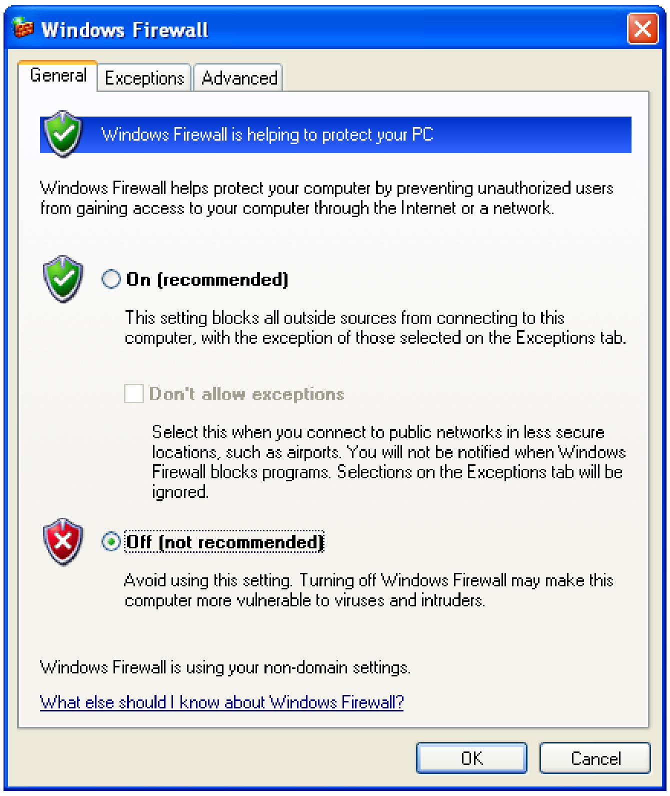
(7) For Windows XP SP2

Because IIS cannot be used by setting the firewall in default when Windows XP Service Pack 2(SP2) is applied, CAP cannot be used. This problem can be solved by the following two methods.

・ Disable the Windows firewall setting.

・ Perform the setting so that IIS connection is exceptionally permitted.

If the computer has been protected enough by the firewall or other security, you can disable the Windows firewall. If the firewall is disabled, you can use CAP provider without performing the following setting. In that case, as Figure4-9 shows, from the Start menu, click [Control panel], click [Windows Firewall], and then select [Off (not recommended)].



Windows Firewall is helping to protect your PC

Windows Firewall helps protect your computer by preventing unauthorized users from gaining access to your computer through the Internet or a network.

○ On (recommended)

This setting blocks all outside sources from connecting to this computer, with the exception of those selected on the Exceptions tab.

☐ Don't allow exceptions

Select this when you connect to public networks in less secure locations, such as airports. You will not be notified when Windows Firewall blocks programs. Selections on the Exceptions tab will be ignored.

● Off (not recommended)

Avoid using this setting. Turning off Windows Firewall may make this computer more vulnerable to viruses and intruders.

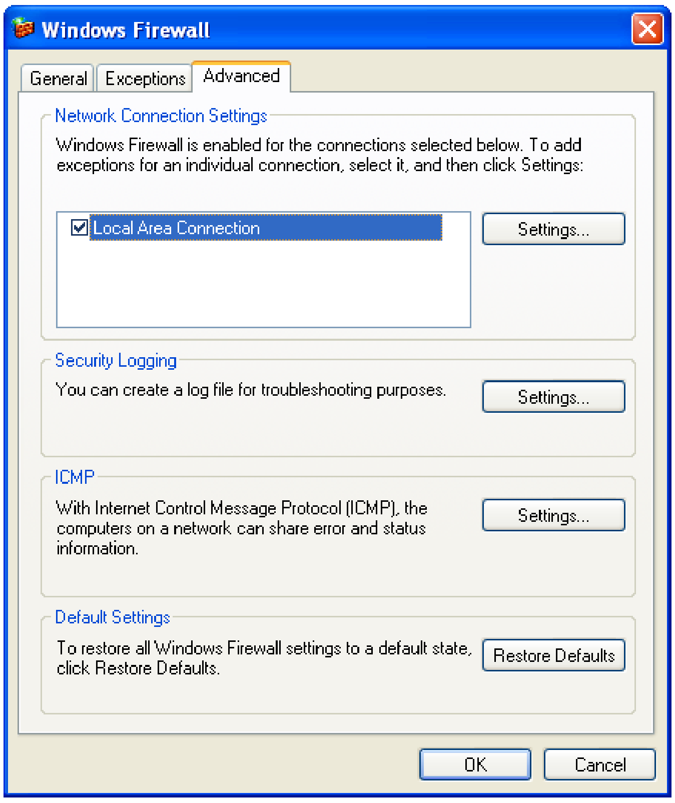
Windows Firewall is using your non-domain settings.

What else should I know about Windows Firewall?

Figure4-9 Nullification of Windows firewall

To permit the connection of IIS the firewall, set as follows.

From the Start menu, click [Control panel], click [Windows Firewall], and select [Advanced] tab. As Figure4-10 shows, from the [Network Connection Settings] pane, select the connection method corresponding to the current environment, and then click [Settings…]. (In this example, "Local Area Connection" is selected.)



Network Connection Settings
Windows Firewall is enabled for the connections selected below. To add exceptions for an individual connection, select it, and then click Settings:

[✓] Local Area Connection

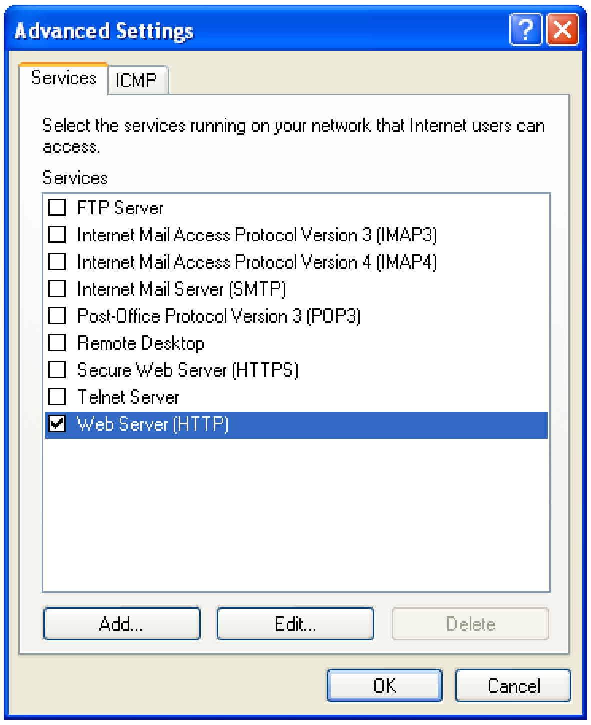
Security Logging
You can create a log file for troubleshooting purposes.

ICMP
With Internet Control Message Protocol (ICMP), the computers on a network can share error and status information.

Default Settings
To restore all Windows Firewall settings to a default state, click Restore Defaults.

Figure4-10 Exception setting of connection

Once a dialog box shown in Figure4-11 is displayed, select [Web Server (HTTP)].


Advanced Settings

Services | ICMP

Select the services running on your network that Internet users can access.

Services

- [ ] FTP Server
- [ ] Internet Mail Access Protocol Version 3 (IMAP3)
- [ ] Internet Mail Access Protocol Version 4 (IMAP4)
- [ ] Internet Mail Server (SMTP)
- [ ] Post-Office Protocol Version 3 (POP3)
- [ ] Remote Desktop
- [ ] Secure Web Server (HTTPS)
- [ ] Telnet Server
- [x] Web Server (HTTP)

Figure4-11 Selection of service added to exception

4.4.2. Client side

(1) Installation of SOAP Toolkit

Install Microsoft SOAP Toolkit 3.0. (see Table 4-1)

4.5. Sample program

Create a sample application in order to confirm the operation of CAP. With VB6, create a form that have one text box and two buttons, and then write the following codes.

List 4-1 Sample.frm

```
Private eng As CaoEngine
Private ctrl As CaoController
Private var As CaoVariable

Private Sub Form_Load()
       Dim ws As CaoWorkspace
       Set eng = New CaoEngine
       Set ws = eng.Workspaces(0)

        ' Server connect
       Set ctrl = ws.AddController( _                              (1)
                "RC1", "CaoProv.CAP", "", "Provider=CaoProv.DataStore,Server=XXXXX")

       ' Acquisition of variable
       Set var = ctrl.AddVariable("Var1")
End Sub
```

```
' Setting of variable
Private Sub Command1_Click()
       var = Text1.Text
End Sub

' Acquisition of variable
Private Sub Command2_Click()
       Text1.Text = var
End Sub
```

Look at (1) in the source code. AddController method establishes the connection to the provider via the Internet. The first to the third arguments of this method are the same as other providers. The content of the fourth argument is shown below.

Provider	:	Provider name accessed by way of the Internet
Server	:	Host name of Web server
Option	:	Option character string of provider that remotely accesses it

Thus, the CaoController object acquired in the AddController method can be used by using the CAP provider as a controller target of the specified provider name.
In the above example, “ctrl” object can be treated as a controller object of the DataStore provider.

5. Environmental setting

5.1. DCOM setting procedures

In ORiN2 SDK, to start CAO or CAO provider remotely, you need to perform settings by using DCOM configuration tool. If the security settings of DCOM client and/or server application is incorrect, an error occurs and the remote start or remote access will fail.

This chapter explains the DCOM setting procedure by using DCOMCNFG.exem which is a configuration tool. For details, refer to the following reference documents.

・ COM/COM + security
  http://www.microsoft.com/japan/com/compapers.asp - comseq
・ Method of setting DCOM to VB5 application program with VB5 DCOMCNFG.EXE
  http://support.microsoft.com/default.aspx?scid=kb;ja;JP183607
・ "DCOM Deployment guide" (published by Syoei-sha) Written by Frank E. Redmond III, Translated by Top studio, and Supervised by Ryoji Kaneko

The content introduces here is a setting to start CAO or the CAO provider with DCOM remotely, and the security setting level is low (security is not set) in this example. Please note that the system can be attacked by malicious external program. If possible, please review the setting according to an individual environment.

5.1.1. Preparation

(1) Prepare a computer that will be a DCOM server and a computer that will be a DCOM client. The following table shows the applicable OS.

Table 5-1 OS and correspondence of DCOM

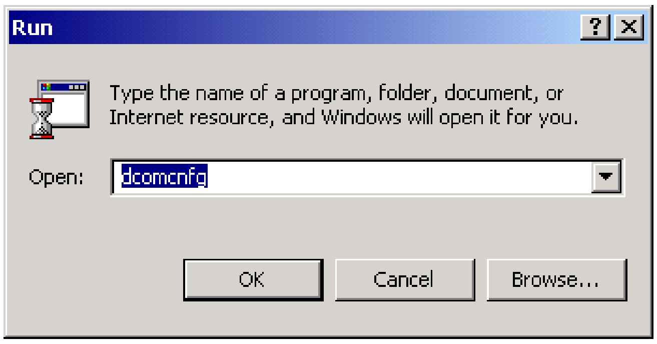
OS	DCOM server	DCOM client
Windows95	✓	✓
Windows98	✓	✓
WindowsNT 4.0 WorkStation	✓	✓
WindowsNT 4.0 Server	✓	✓
Windows2000 Professional	✓	✓
Windows2000 Server	✓	✓
Windows XP Professional	✓	✓

(2) Please register the DCOM client and the DCOM server in the same domain.

5.1.2. Set item in Windows NT/2000

(1) DCOMCNFG.EXE is started.

From the Start menu, click [Run], and then enter “dcomcnfg" to start DCOMCNFG.EXE.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: dcomcnfg

Figure5-1 Start of DCOMCNFG

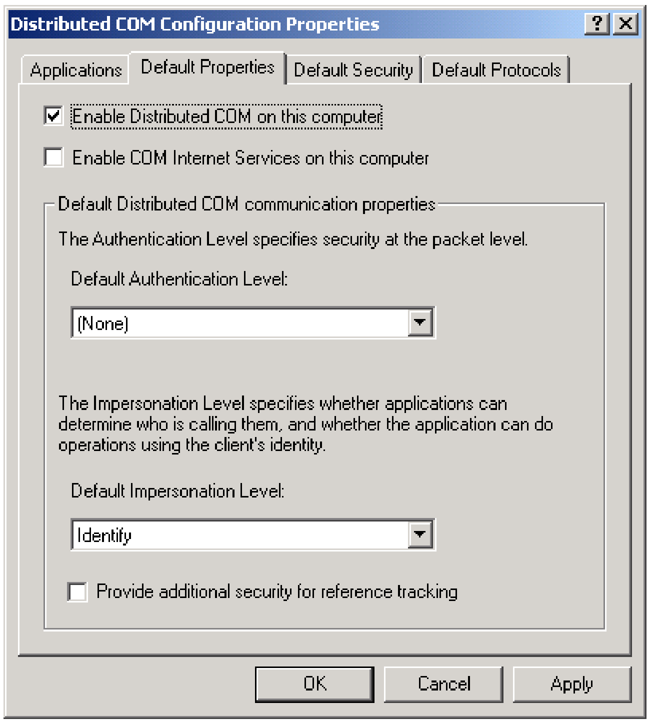
(2) Setting of default value

You can set the default values for DCOM property. Once you set the default values, it affect all DCOM applications installed in the local computer. Therefore, you have to be very careful to enter such values. In addition, do not change this setting when there is a problem in security. (However, in this case, CAO cannot be used via DCOM.)

Click [Default properties] tab.

As Figure5-2 shows, select the [Enable Distributed COM on this computer] checkbox. On the

[Default Distributed COM communication properties] pane, in the [Default Authentication Level], select [None], and in the [Default Impersonation Level], select [Identify].


Distributed COM Configuration Properties

Applications | Default Properties | Default Security | Default Protocols

[✓] Enable Distributed COM on this computer

[ ] Enable COM Internet Services on this computer

Default Distributed COM communication properties

The Authentication Level specifies security at the packet level.

Default Authentication Level: (None)

The Impersonation Level specifies whether applications can determine who is calling them, and whether the application can do operations using the client's identity.

Default Impersonation Level: Identify

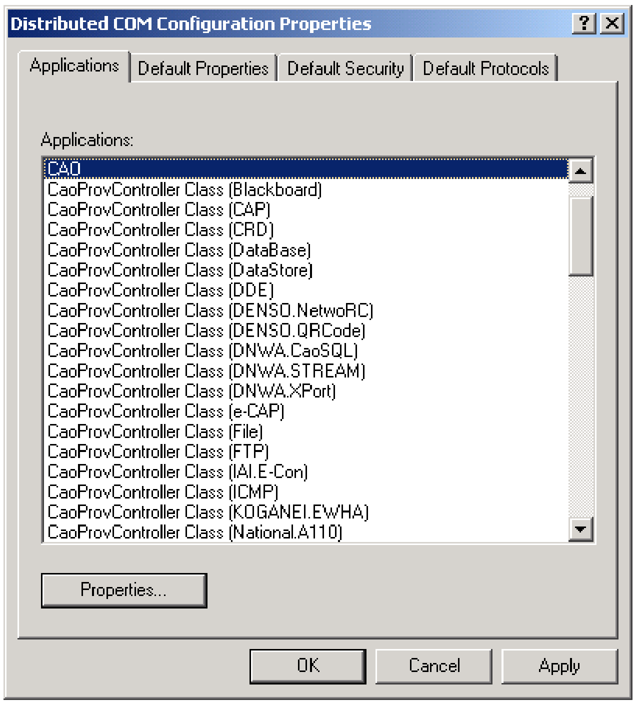
[ ] Provide additional security for reference tracking

Figure5-2 Default property

(3) Setting of each application

From the tab of open DCOMCNFG.EXE, select [Applications] tab, select an application that you want to start by DCOM, and then click [Property].

This operation must be done for all applications that are remotely started. It means, if providers are started by DCOM, this processing must be done for all the providers that are started by DCOM. Moreover, if remote CAO is used, this process must be done to CAO.


Distributed COM Configuration Properties

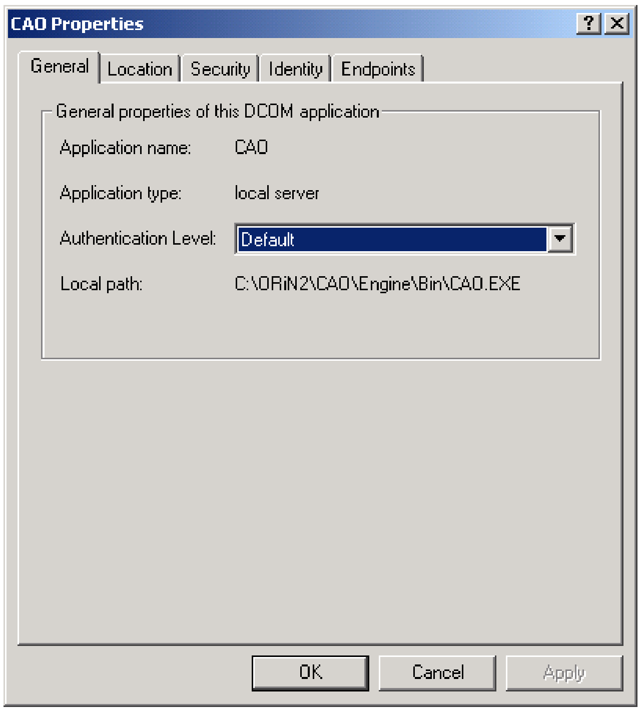
Applications | Default Properties | Default Security | Default Protocols

Applications:

CAO
CaoProvController Class (Blackboard)
CaoProvController Class (CAP)
CaoProvController Class (CRD)
CaoProvController Class (DataBase)
CaoProvController Class (DataStore)
CaoProvController Class (DDE)
CaoProvController Class (DENSO.NetwoRC)
CaoProvController Class (DENSO.QRCode)
CaoProvController Class (DNWA.CaoSQL)
CaoProvController Class (DNWA.STREAM)
CaoProvController Class (DNWA.XPort)
CaoProvController Class (e-CAP)
CaoProvController Class (File)
CaoProvController Class (FTP)
CaoProvController Class (IAI.E-Con)
CaoProvController Class (ICMP)
CaoProvController Class (KOGANEI.EWHA)
CaoProvController Class (National.A110)

Figure5-3 Setting of each application

Select [General] tab. Set the Authentication Level to [Default] since this sample uses the default authentication level.


CAO Properties

General | Location | Security | Identity | Endpoints

General properties of this DCOM application

Application name: CAO

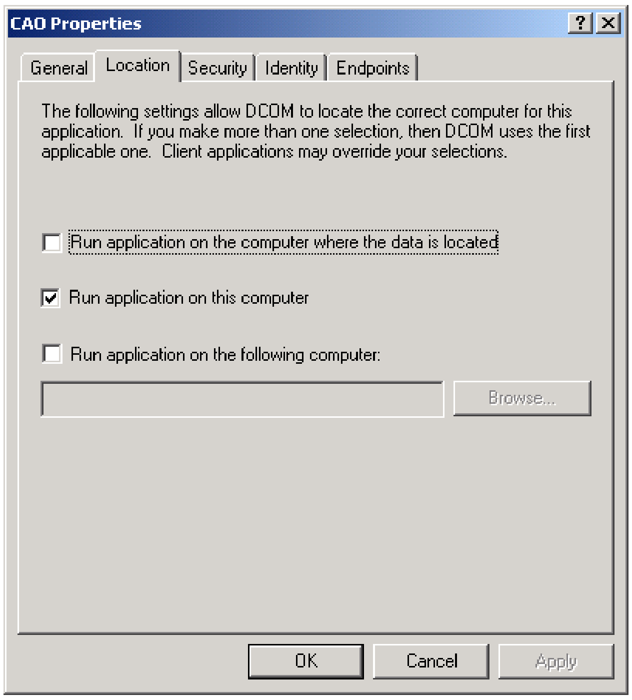
Application type: local server

Authentication Level: Default

Local path: C:\ORiN2\CAO\Engine\Bin\CAO.EXE

Figure5-4 General setting

Select [Location] tab. In this setting, the location where the application is executed can be specified. Select [Run application on this computer].Remove other checks.

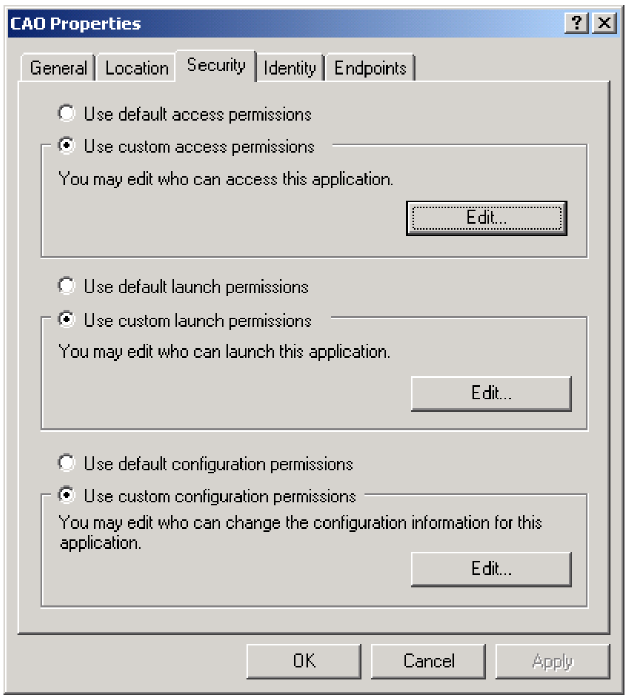


The following settings allow DCOM to locate the correct computer for this application. If you make more than one selection, then DCOM uses the first applicable one. Client applications may override your selections.

- [ ] Run application on the computer where the data is located
- [✓] Run application on this computer
- [ ] Run application on the following computer:

Figure5-5 Location setting

Select [Security] tab.



CAO Properties

General | Location | Security | Identity | Endpoints

Use default access permissions [✓ Use custom access permissions]

You may edit who can access this application.

Edit...

Use default launch permissions [✓ Use custom launch permissions]

You may edit who can launch this application.

Edit...

Use default configuration permissions [✓ Use custom configuration permissions]

You may edit who can change the configuration information for this application.

Edit...

OK | Cancel | Apply

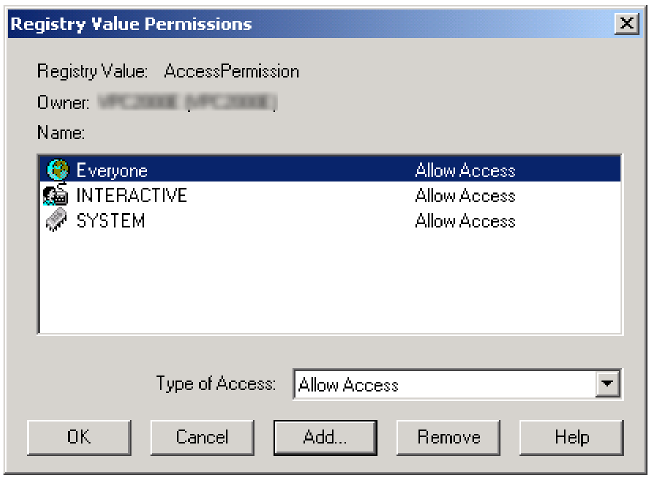
Figure5-6 Security setting

Select "Use custom access permissions", click [Edit...] button. Add the following access right.

・ Everyone – Allow Access

・ INTERACTIVE - Allow Access

・ SYSEM - Allow Access



Registry Value Permissions

Registry Value: AccessPermission

Owner: [redacted]

Name:

Everyone	Allow Access
INTERACTIVE	Allow Access
SYSTEM	Allow Access

Type of Access: Allow Access

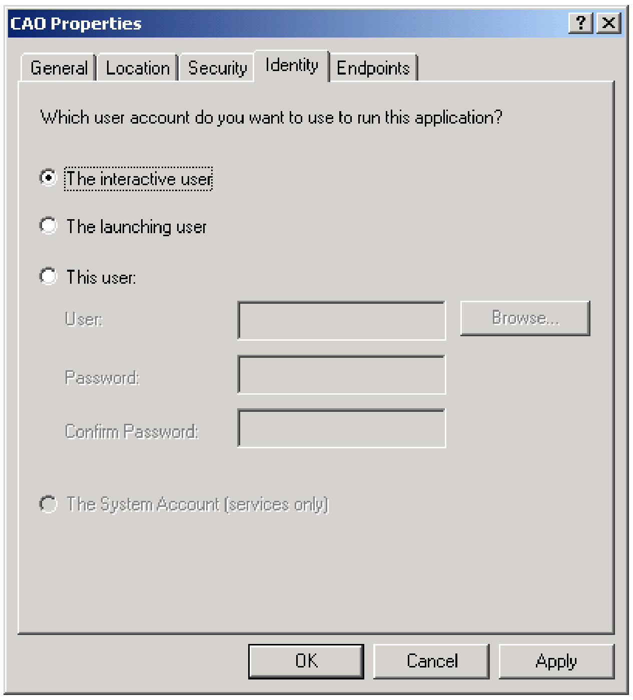
OK | Cancel | Add... | Remove | Help

Figure5-7 Addition of access right

Select [Use custom launch Permissions], click [Edit...] button. Add the following launch permissions.

- Everyone – Allow Access
- INTERACTIVE - Allow Access
- SYSEM - Allow Access

Select [Identify] tab, and then select [The interactive user]. If you start DCOM server with Windows NT, [The launching user] is acceptable. In addition, you may select [This user] and enter a user. However, at the service registration, [The launching user] and [The Interactive user] cannot be selected. (You can use the system account instead.)



CAO Properties

Which user account do you want to use to run this application?

[✓ The interactive user]

The launching user

This user:

User:

Password:

Confirm Password:

The System Account (services only)

Figure5-8 Identification setting

5.1.3. Set item in Windows9x

To use Windows 98 or Windows 95, please download DCOM98 or DCOM95 and DCOMCNFG98&95 from following URL according to OS, and install it in your computer.

< Microsoft COM technology-related file download>

http://www.microsoft.com/com/default.mspx

(1) Start DCOMCNFG.EXE.

From the Start menu, on the [Run…] textbox, enter “DCOMCNFG" to start DCOMCNFG.EXE.

(2) Set the default properties

Set as follows.

・ Authentication Level:	None
・ Impersonation Level:	Identify.

(3) Set the default access right

Add the following access right.

・ World-Permit.

Because the DCOM98 of Windows 9x does not support remote start function, to set 9x-series computer as a server, you need to start the program beforehand. To start not the CAO engine but provider DLL beforehand, with dllhost.exe, start the provider DLL by using command prompt as follows.

```
> dllhost.exe {<AppID>}
```

For instance, to start TCmini provider, write as follows.

```
> dllhost.exe {2c140adc-c109-43df-a059-c3b116257658}
```

< AppID > has been described to the CaoProvController.RGS file of each provider source project. Also, GUI of CaoProvExec tool which is supplied with ORiN2 SDK allows you to perform the same setting as above. For information of how to use CaoProvExec, see "CAO provider development guide".

5.1.4. Set item in Windows XP

(1) Start of DCOMCNFG

From the Start menu, from [Run…], enter "dcomcnfg" to start DCOMCNFG.EXE.

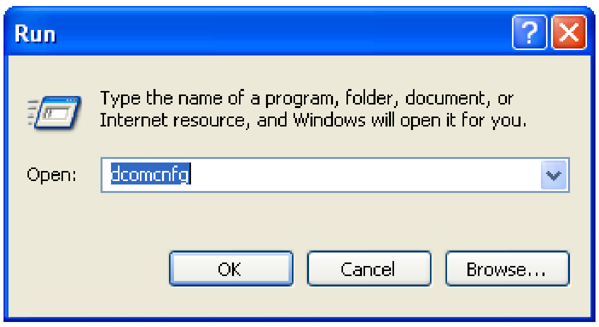

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
dcomcnfg
OK
Cancel
Browse...

Figure5-9 Starting DCOMCNFG

(2) Setting of My Computer

As the Figure5-10 shows, click [Component Services], click [Computer], click [My Computer], and then right-click [Properties].

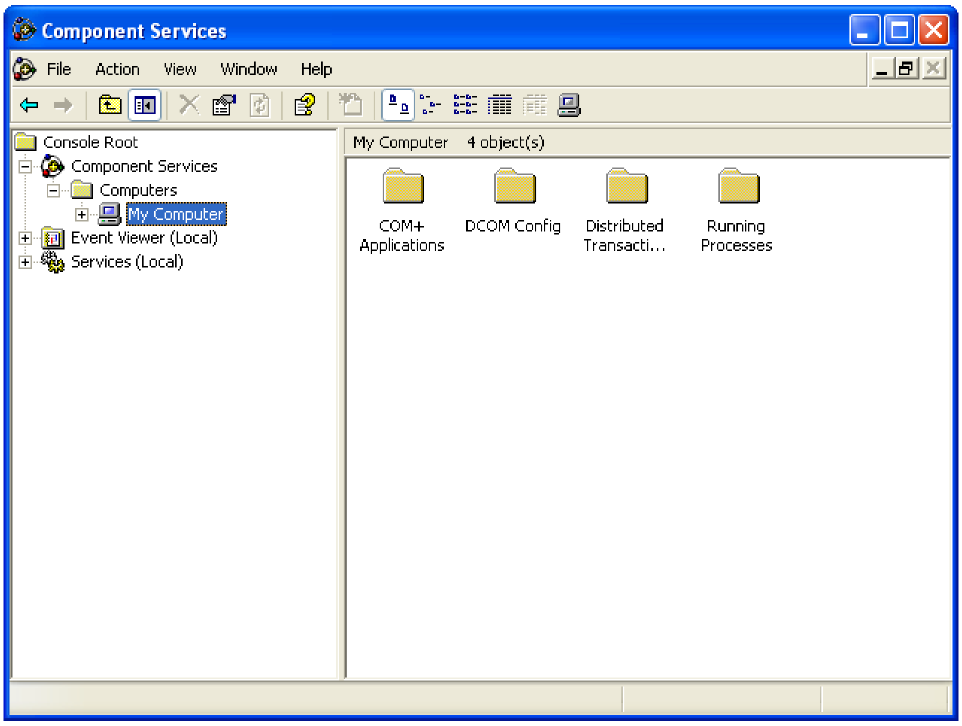
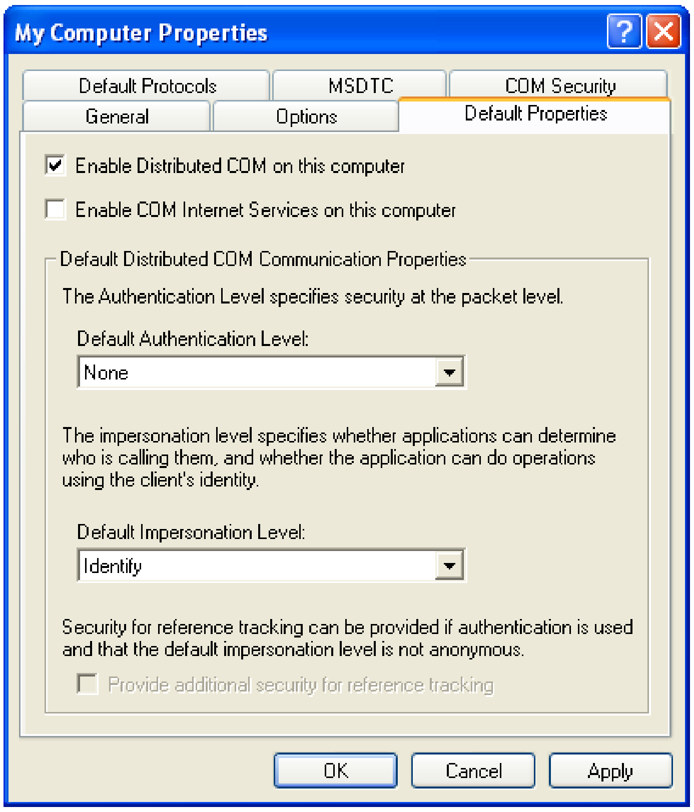


Figure5-10 Set screen of DCOMCNFG

As Figure5-11 shows, select [Default Properties] tab.

Select [Enable Distributed COM on this computer].

Set [Default Authentication Level] to [None], and [Default Impersonation Level] to [Identify].



[✓] Enable Distributed COM on this computer

[ ] Enable COM Internet Services on this computer

Default Distributed COM Communication Properties

The Authentication Level specifies security at the packet level.

Default Authentication Level: None

The impersonation level specifies whether applications can determine who is calling them, and whether the application can do operations using the client's identity.

Default Impersonation Level: Identify

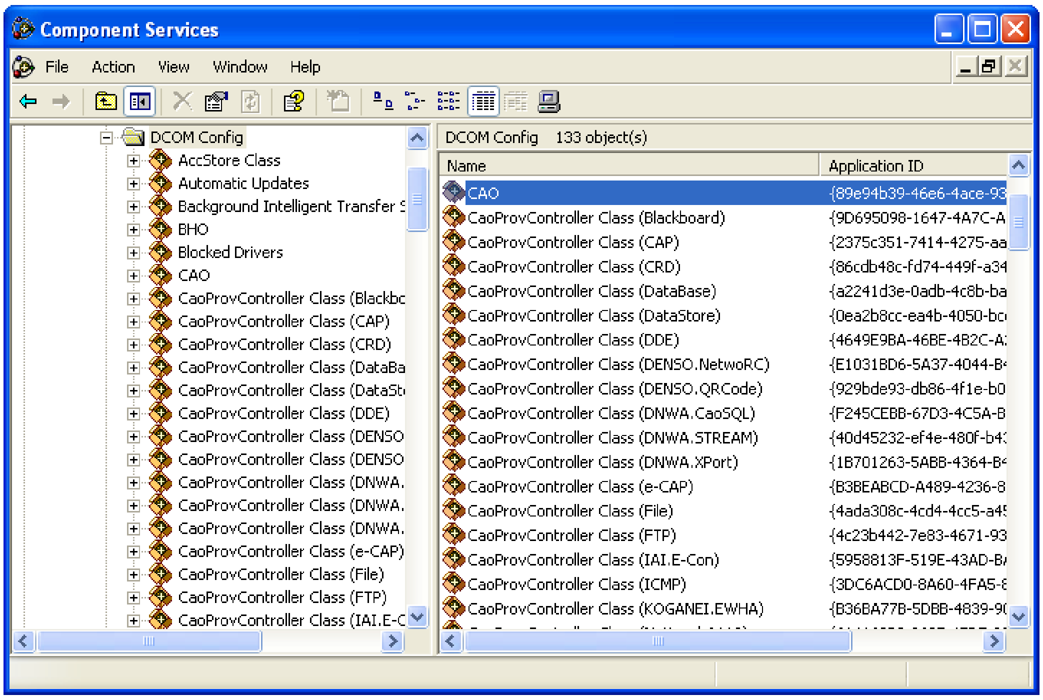
Security for reference tracking can be provided if authentication is used and that the default impersonation level is not anonymous.

[ ] Provide additional security for reference tracking

Figure5-11 My Computer Properties

(3) Setting of each application

From the [Component Services], click [Computer], click [My Computer]. From [DCOM Config], right-click [CAO], and then select [CAO Properties].



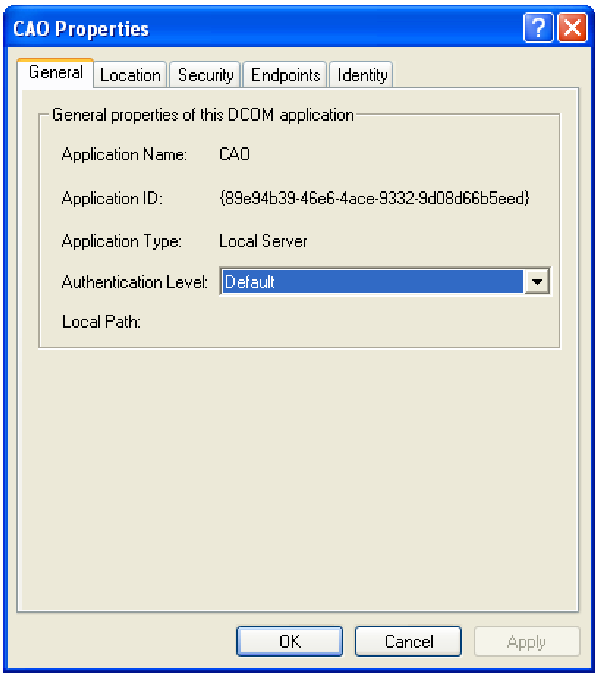
- DCOM Config
  - AccStore Class
  - Automatic Updates
  - Background Intelligent Transfer S
  - BHO
  - Blocked Drivers
  - CAO
  - CaoProvController Class (Blackbc
  - CaoProvController Class (CAP)
  - CaoProvController Class (CRD)
  - CaoProvController Class (DataBa
  - CaoProvController Class (DataSt
  - CaoProvController Class (DDE)
  - CaoProvController Class (DENSO
  - CaoProvController Class (DENSO
  - CaoProvController Class (DNWA.
  - CaoProvController Class (DNWA.
  - CaoProvController Class (DNWA.
  - CaoProvController Class (e-CAP)
  - CaoProvController Class (File)
  - CaoProvController Class (FTP)
  - CaoProvController Class (IAI.E-C

DCOM Config 133 object(s)

Name	Application ID
CAO	{89e94b39-46e6-4ace-93
CaoProvController Class (Blackboard)	{9D695098-1647-4A7C-A
CaoProvController Class (CAP)	{2375c351-7414-4275-aa
CaoProvController Class (CRD)	{86cdb48c-fd74-449f-a34
CaoProvController Class (DataBase)	{a2241d3e-0adb-4c8b-ba
CaoProvController Class (DataStore)	{0ea2b8cc-ea4b-4050-bc
CaoProvController Class (DDE)	{4649E9BA-46BE-4B2C-A
CaoProvController Class (DENSO.NetwoRC)	{E1031BD6-5A37-4044-B
CaoProvController Class (DENSO.QRCode)	{929bde93-db86-4f1e-b0
CaoProvController Class (DNWA.CaoSQL)	{F245CEBB-67D3-4C5A-B
CaoProvController Class (DNWA.STREAM)	{40d45232-ef4e-480f-b4
CaoProvController Class (DNWA.XPort)	{1B701263-5ABB-4364-B
CaoProvController Class (e-CAP)	{B3BEABCD-A489-4236-8
CaoProvController Class (File)	{4ada308c-4cd4-4cc5-a4
CaoProvController Class (FTP)	{4c23b442-7e83-4671-93
CaoProvController Class (IAI.E-Con)	{5958813F-519E-43AD-B
CaoProvController Class (ICMP)	{3DC6ACD0-8A60-4FA5-8
CaoProvController Class (KOGANEI.EWHA)	{B36BA77B-5DBB-4839-9

Figure5-12 DCOMCNFG DCOM structure setting window

Select [General] tab. Set [Authentication Level] to [Default].



General properties of this DCOM application

Application Name: CAO

Application ID: {89e94b39-46e6-4ace-9332-9d08d66b5eed}

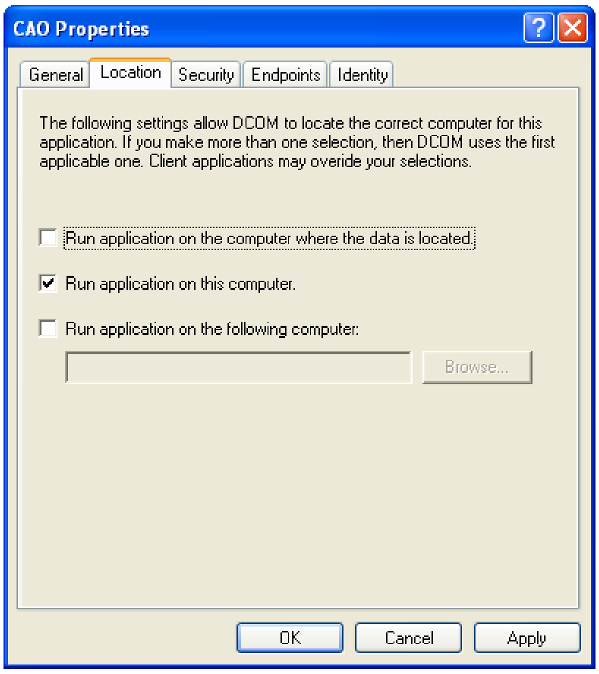
Application Type: Local Server

Authentication Level: Default

Local Path:

Figure5-13 Setting of each application (general)

Select [Location] tab. Select [Run application on this computer].



The following settings allow DCOM to locate the correct computer for this application. If you make more than one selection, then DCOM uses the first applicable one. Client applications may overide your selections.

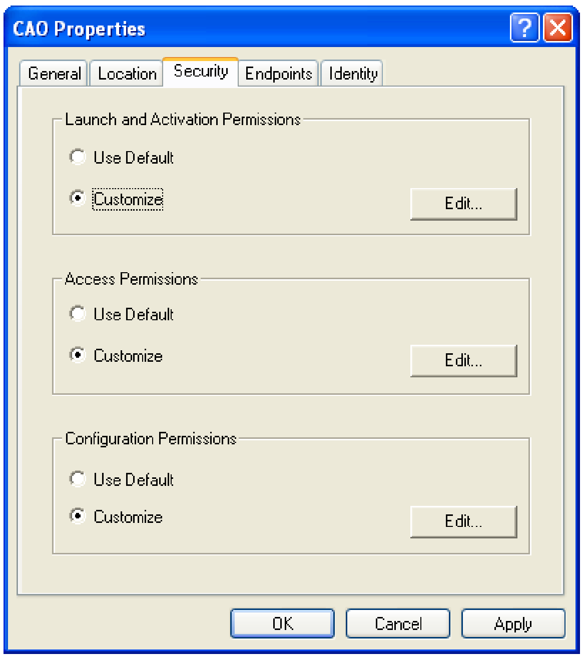
☐ Run application on the computer where the data is located.

☑ Run application on this computer.

☐ Run application on the following computer:

Figure5-14 Location setting of each application

Select [Security] tab. On the [Launch and Activation Permissions] pane, select [Customize]. On the [Access Permissions] pane, select [Customize]. Click [Edit]



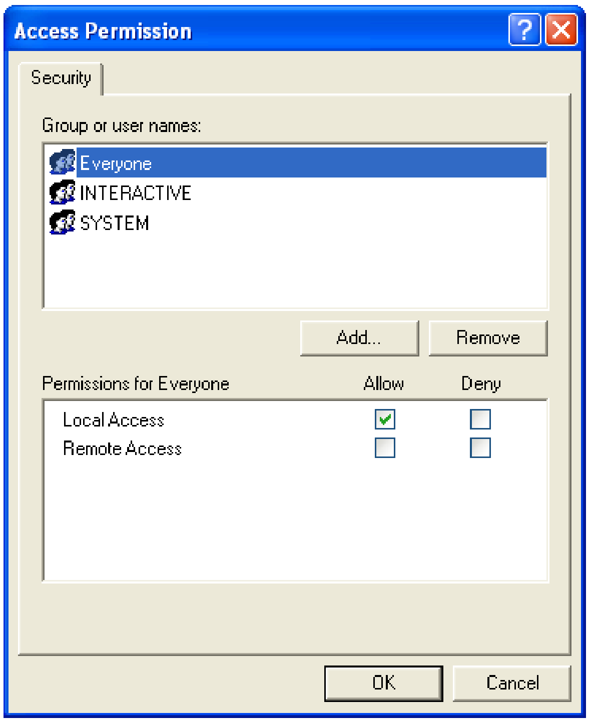
Launch and Activation Permissions
Use Default [✓ Customize]

Access Permissions
Use Default [✓ Customize]

Configuration Permissions
Use Default [✓ Customize]

Figure5-15 Set security of each application

Click [Add…] button. Add [Everyone], [INTERACTIVE], and [SYSTEM], and then select [Local access].



Access Permission

Security

Group or user names:

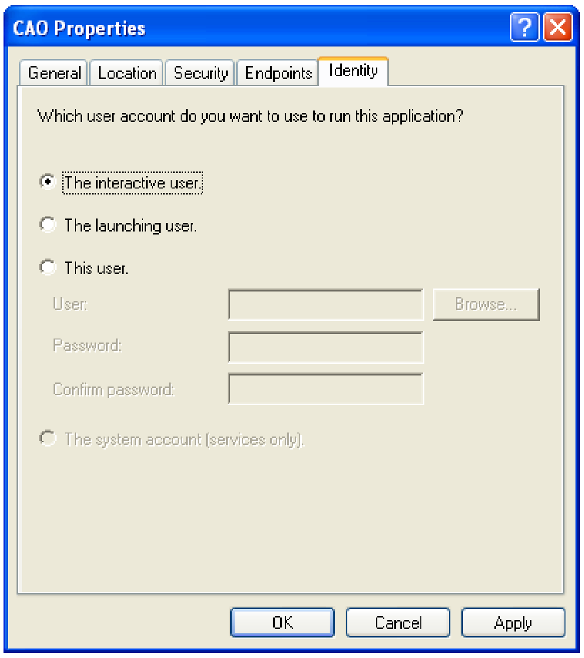
- Everyone
- INTERACTIVE
- SYSTEM

Permissions for Everyone

	Allow	Deny
Local Access	✓	
Remote Access		

Figure5-16 Security access permission for each application

Select [Identify] tab. Select [The interactive user].



CAO Properties

General | Location | Security | Endpoints | Identity

Which user account do you want to use to run this application?

- [✓ The interactive user.]
- The launching user.
- This user.
  - User:
  - Password:
  - Confirm password:
- The system account (services only).

Figure5-17 ID setting for each application

Perform this settings for CAO providers that you will use as well.

5.1.5. Set item in Windows XP SP2

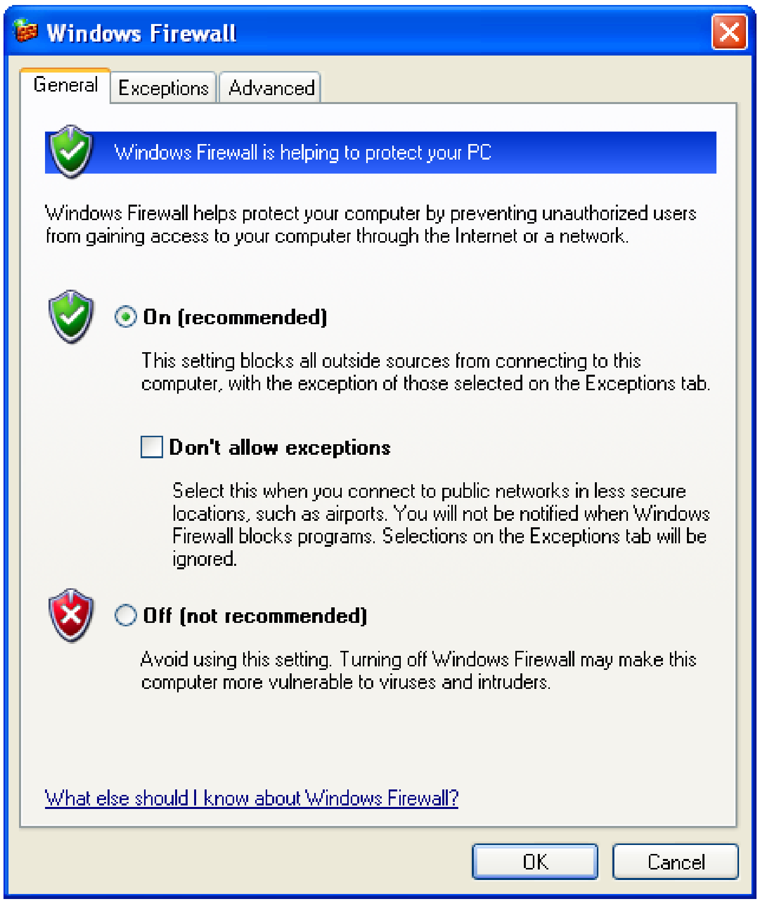
To establish strong security, security items (such as DCOM) of Windows XP Service Pack 2 (SP2) have been reinforced and the firewall function is active in the default setting.

Therefore, it is impossible to access remote CAO provider.

5.1.5.1. Setting of Windows firewall

If the computer has been protected enough by the firewall or other securities, you can disable the Windows firewall. When the firewall is disabled, CAO provider can be remotely used even if the following setting is not done.

From the Start menu, click [Setting], click [Control Panel], and then click [Windows Firewall].



Windows Firewall is helping to protect your PC

Windows Firewall helps protect your computer by preventing unauthorized users from gaining access to your computer through the Internet or a network.

[✓ On (recommended)]

This setting blocks all outside sources from connecting to this computer, with the exception of those selected on the Exceptions tab.

☐ Don't allow exceptions

Select this when you connect to public networks in less secure locations, such as airports. You will not be notified when Windows Firewall blocks programs. Selections on the Exceptions tab will be ignored.

○ Off (not recommended)

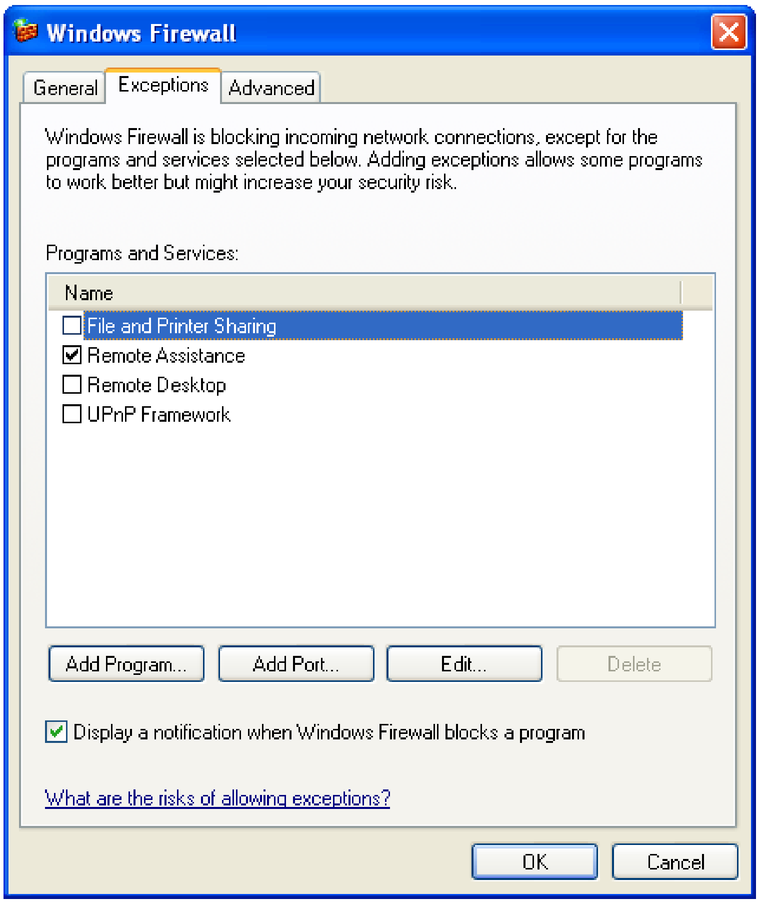
Avoid using this setting. Turning off Windows Firewall may make this computer more vulnerable to viruses and intruders.

What else should I know about Windows Firewall?

Figure5-18 Windows firewall setting

5.1.5.2. Exception setting of firewall

From the Start menu, click [Setting], click [Control Panel], click [Windows Firewall]. Select [Exception] tab.



Windows Firewall is blocking incoming network connections, except for the programs and services selected below. Adding exceptions allows some programs to work better but might increase your security risk.

Programs and Services:

Name
☐ File and Printer Sharing
☑ Remote Assistance
☐ Remote Desktop
☐ UPnP Framework

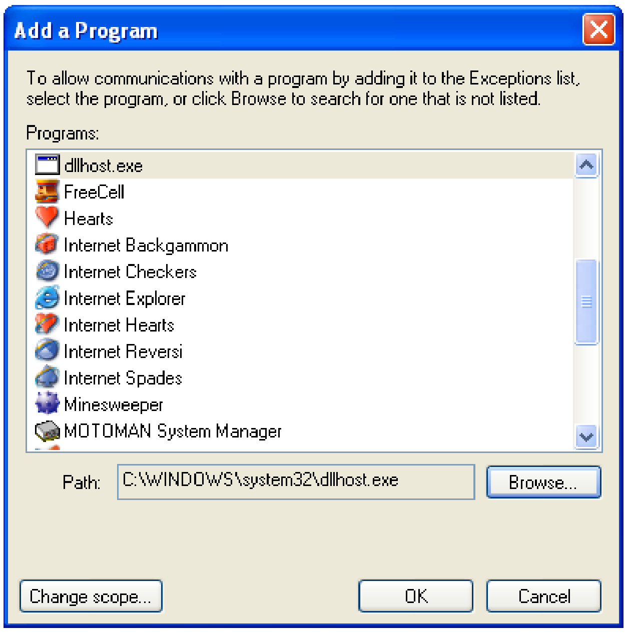
☑ Display a notification when Windows Firewall blocks a program

What are the risks of allowing exceptions?

Figure5-19 Addition of exceptions

On the [Exceptions] tab, once [Add Program…] button is clicked, the following window appears (see Figure5-20). Add the following programs. If there is no list exist on the Program pane, click [Browse..].

< Windows installation directory > ¥WINDOWS¥system32¥dllhost.exe


Add a Program

To allow communications with a program by adding it to the Exceptions list, select the program, or click Browse to search for one that is not listed.

Programs:

dllhost.exe
FreeCell
Hearts
Internet Backgammon
Internet Checkers
Internet Explorer
Internet Hearts
Internet Reversi
Internet Spades
Minesweeper
MOTOMAN System Manager

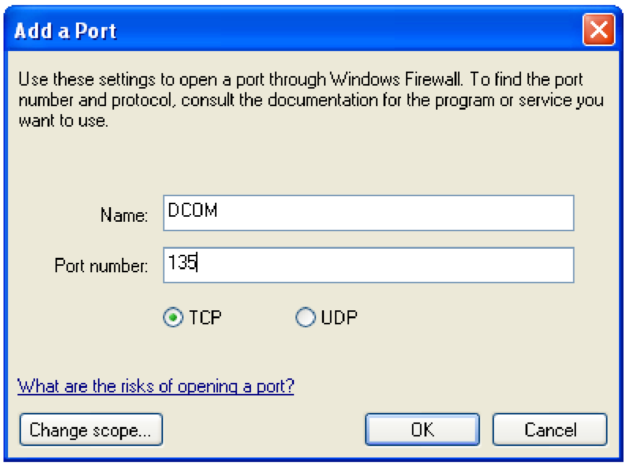
Path: C:\WINDOWS\system32\dllhost.exe

Figure5-20 Addition of exception program

On the [Windows Firewall] window, on the [Exceptions] tab, click [Add Port..] button.

Add a following port on the [Add a Port] window.

- Name: DCOM
- Port number: 135


Add a Port

Use these settings to open a port through Windows Firewall. To find the port number and protocol, consult the documentation for the program or service you want to use.

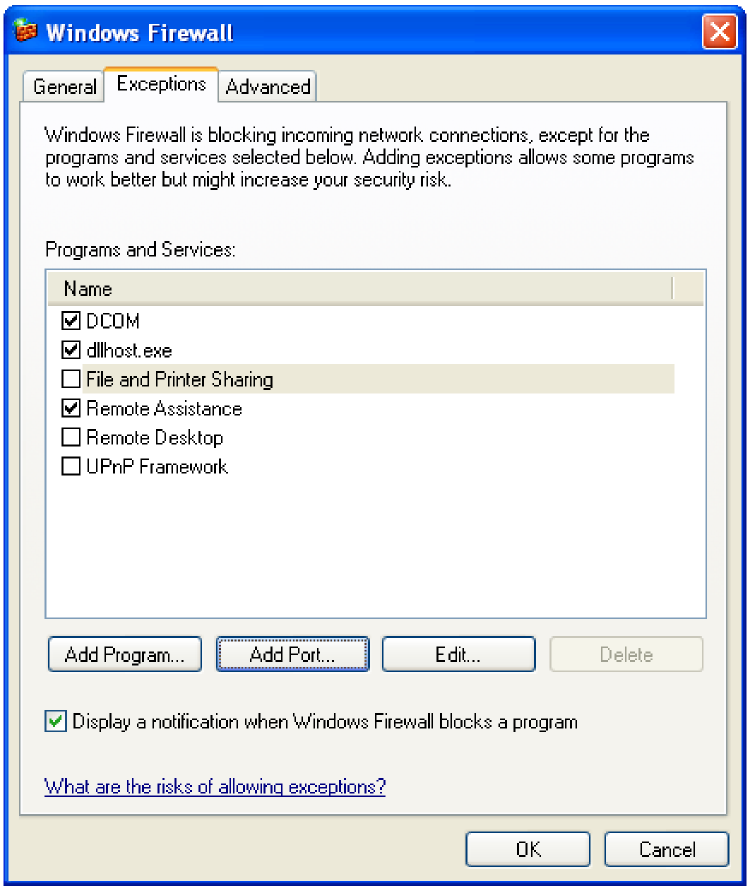
Name: DCOM

Port number: 135

[✓ TCP] UDP

Figure5-21 Addition of exception port number

As shown in Figure5-22 , check that the [DCOM] and [dllhost.exe] checkbox are properly selected, and then click [OK] button. The security setting of the firewall is completed.


Windows Firewall

General | Exceptions | Advanced

Windows Firewall is blocking incoming network connections, except for the programs and services selected below. Adding exceptions allows some programs to work better but might increase your security risk.

Programs and Services:

Name
✓ DCOM
✓ dllhost.exe
☐ File and Printer Sharing
✓ Remote Assistance
☐ Remote Desktop
☐ UPnP Framework

✓ Display a notification when Windows Firewall blocks a program

What are the risks of allowing exceptions?

Figure5-22 Additional result of exception

5.1.5.3. Setting of DCOM

Perform the same settings as 5.1.4Set item in Windows XP.

5.1.6. Set confirmation of DCOM

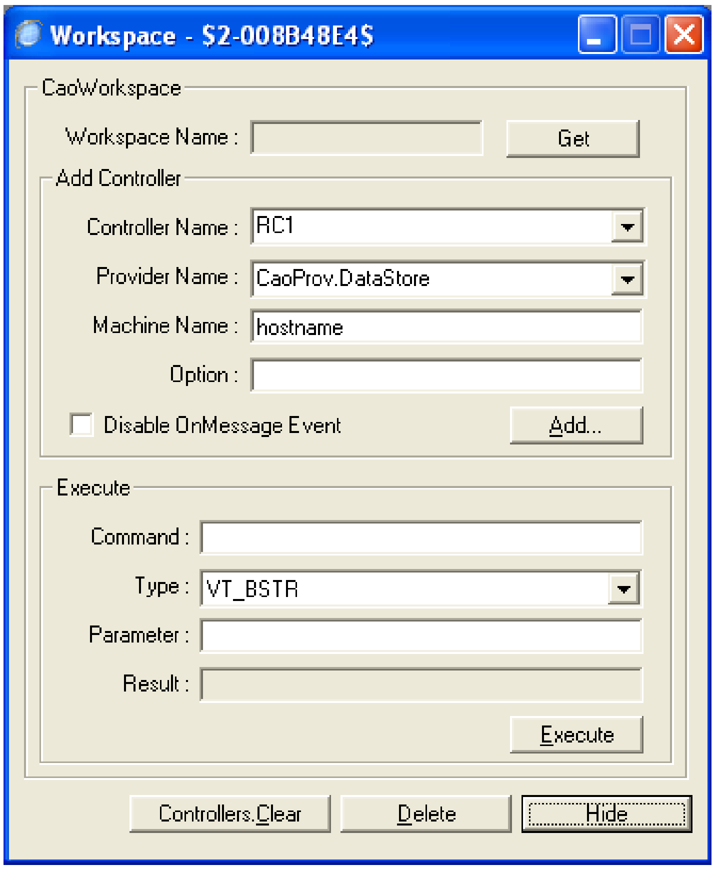
To confirm the DCOM settings, perform the provider connection test remotely by using CaoTester, which is a test tool provided with ORiN2 SDK.

Please refer to the CaoTester user's guide for details.

In this time, DataStore provider is used as a remote connection destination provider. Perform "(3) Setting of each application" to the DataStore provider.

Start CaoTester with a computer on the client side. Add following items to the workspace child window as shown Figure5-23.

Controller Name	:	Enter any arbitrary character strings.
Provider Name	:	CaoProv.DataStore
Machine Name	:	Enter the host name of the DCOM server.


Workspace - $2-008B48E4$

CaoWorkspace

Workspace Name :

Add Controller

Controller Name : RC1

Provider Name : CaoProv.DataStore

Machine Name : hostname

Option :

☐ Disable OnMessage Event

Execute

Command :

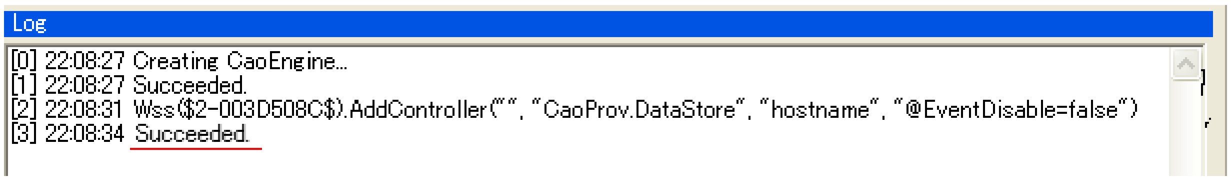
Type : VT_BSTR

Parameter :

Result :

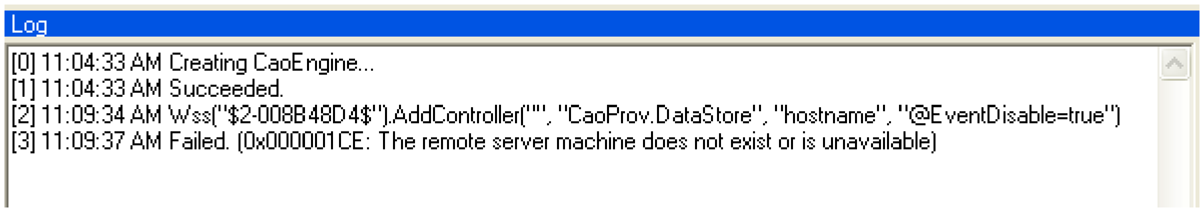
Figure5-23 Remote provider connection by DCOM

Click “Add" button of the workspace child window. Once the remote connection is successfully completed, [Succeeded] will be displayed in the log window (see Figure 5-24) that is on the left bottom side of CaoTester. When the remote connection fails, [Failed...] will be displayed (see Figure5-25).



```
[0] 22:08:27 Creating CaoEngine...
[1] 22:08:27 Succeeded.
[2] 22:08:31 Wss($2-003D508C$).AddController("", "CaoProv.DataStore", "hostname", "@EventDisable=false")
[3] 22:08:34 Succeeded.
```

Figure 5-24 DCOM connection result (Succeed)



```
[0] 11:04:33 AM Creating CaoEngine...
[1] 11:04:33 AM Succeeded.
[2] 11:09:34 AM Wss("$2-008B48D4$").AddController("", "CaoProv.DataStore", "hostname", "@EventDisable=true")
[3] 11:09:37 AM Failed. (0x000001CE: The remote server machine does not exist or is unavailable)
```

Figure5-25 DCOM connection result (Failed)

5.2. Integration into Visual Studio

You can integrate ORiN2 SDK into Visual Studio for facilitating ORiN application development.

To integrate ORiN2 SDK into Visual Studio, double-click <ORiN Install Directory>¥Tools¥VSAddins¥Bin¥ORiN2AddIns.exe.

From the drop-down list box, select a version of Visual Studio that you want to integrate, and then click [Install] button (see Figure 5-26) to complete integration. To release the integration, click [Uninstall] button (see Figure 5-27).

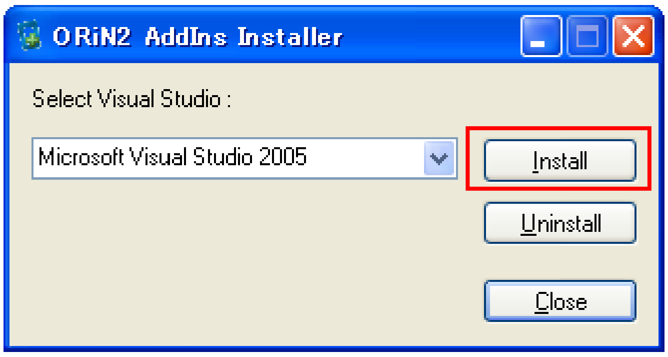


Figure 5-26 Integrate into Visual Studio

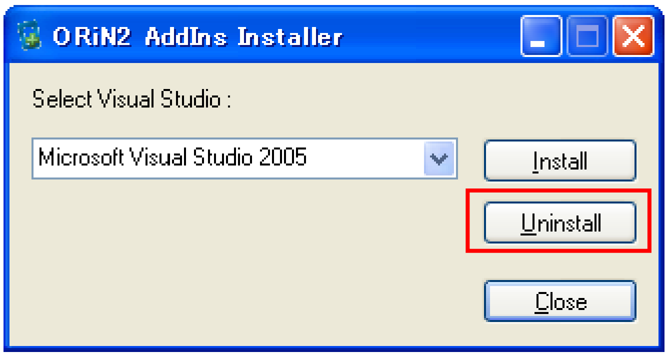


Figure 5-27 Release the integration with Visual Studio

5.2.1. Visual Studio menu bar

Once the ORiN2 SDK is integrated into Visual Studio, ORiN2 menu will be added into the menu bar of Visual Studio (see Figure 5-28). ORiN2 menu contains tools and documents provided by ORiN2 SDK. You can easily call necessary tool or document by simply clicking these submenus.

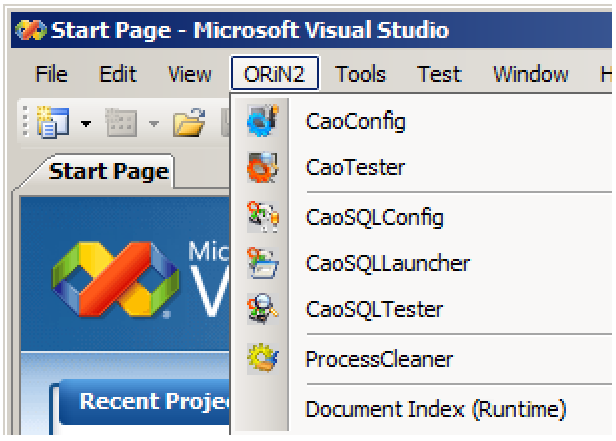
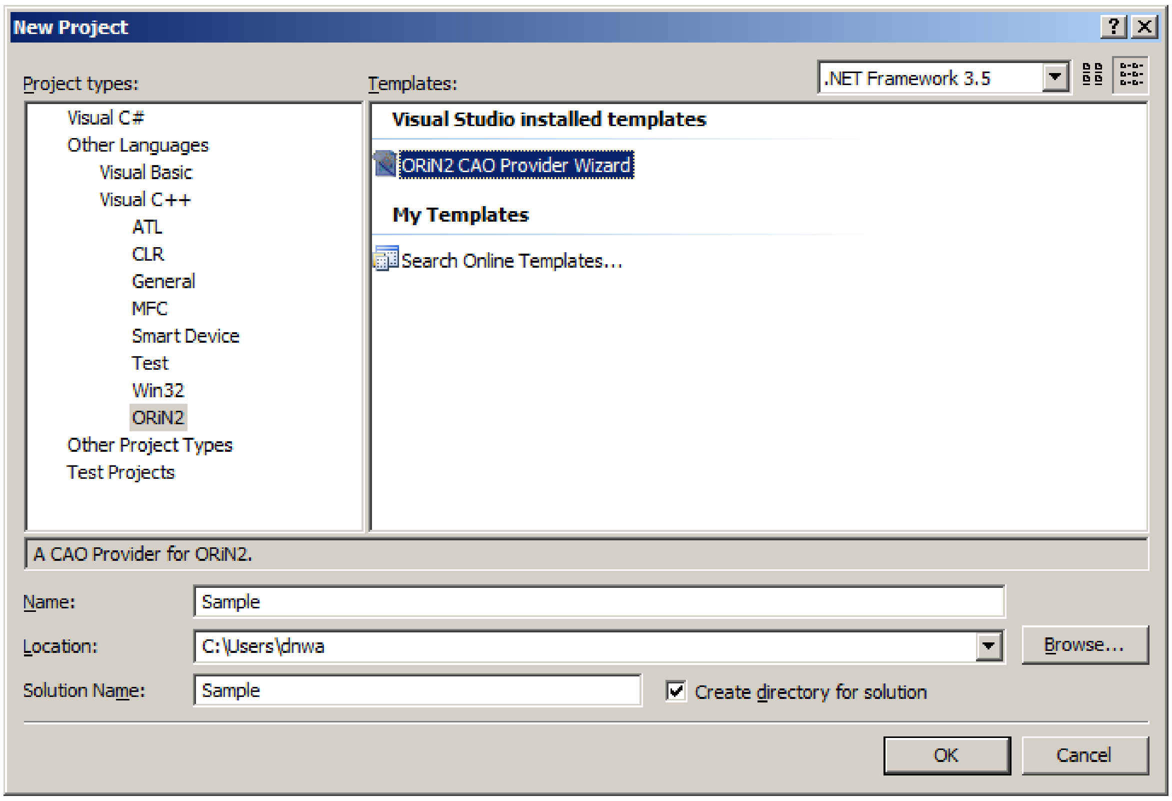


Figure 5-28 ORiN2 menu

5.2.2. CAO Provider Wizard

If your ORiN2 package is "ORiN2 SDK (Provider Development)", then ORiN2 items will be added into a new project of Visual Studio (see Figure 5-29). Select "ORiN2 CAO Provider Wizard", and then click [OK] button to create a project for provider development. For detailed information about the provider development, see <ORiN install directory> ¥CAO¥Provider¥Doc¥ProviderDevGuide_en.pdf



A CAO Provider for ORiN2.

Name: Sample
Location: C:\Users\dnwa
Solution Name: Sample
☑ Create directory for solution

Figure 5-29 CAO Provider Wizard

6. CaoConfig

CaoConfig is a tool to change the operation setting of the CAO engine and the CAO provider. This tool can set items like the priority of CAO engine, log output level, enable/disable each CAO provider, and CRD redirection function.

Items set by CaoConfig are recorded in the registry.

6.1. Window layout

6.1.1. Menu

6.1.1.1. File Menu

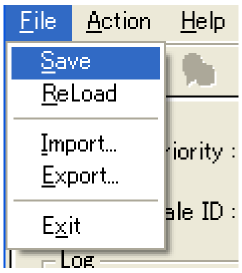


Figure 6-1 CaoConfig File Menu

[Save]

Save the CaoConfig information in the registry.

[Reload]

Information on CaoConfig is acquired from the registry.

[Import]

Import CaoConfig settings from an XML file.

[Export]

Export CaoConfig settings to an XML file.

[Exit]

Close CaoConfig.

6.1.1.2. Action Menu

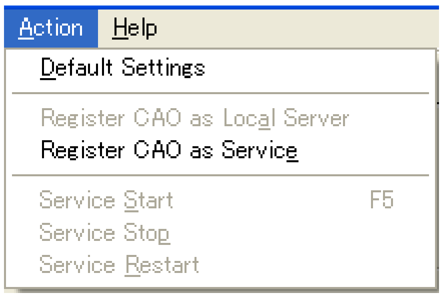


Figure 6-2 CaoConfig Action Menu

[Default Settings]

Set the selected item to the default setting.

When CaoEngine tab is selected, all items in CaoEngine tab will be set to the default settings.

When CaoProvider tab is selected, an item selected in the Provider List will be set to the default setting.

[Service Start]

Start the CAO service. This selection is available only when CAO is registered as a service.

[Service Stop]

Stop the CAO service. This selection is available only when CAO is registered as a service.

[Service Restart]

Restart the CAO service. This selection is available only when CAO is registered as a service.

6.1.1.3. Help Menu

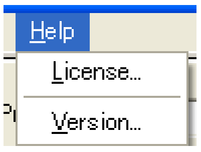


Figure 6-3 CaoConfig Help Menu

[License]

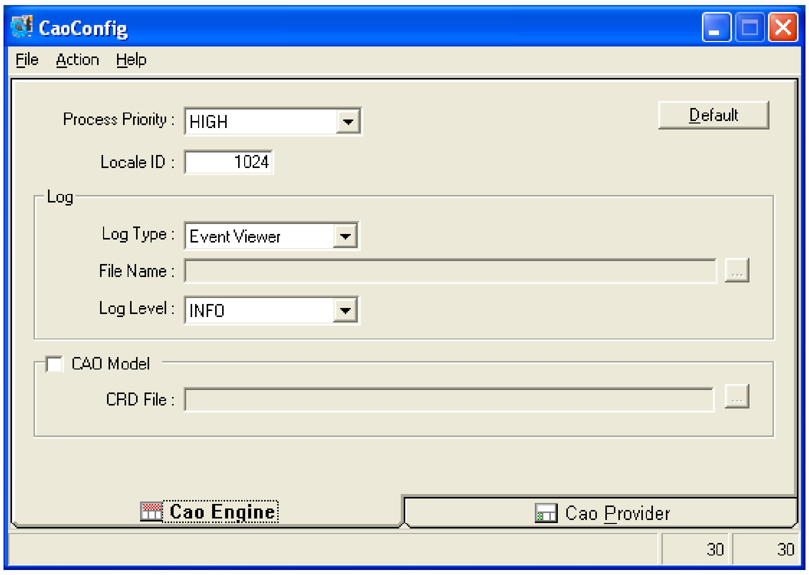
Display the CAO license management window.

[Version]

Display the CaoConfig version information.

6.1.2. Cao Engine tab

You can set CAO Engine operation settings from CaoEngine tab.



Process Priority : HIGH
Locale ID : 1024
Log Type : Event Viewer
File Name :
Log Level : INFO
CRD File :

Figure6-4 Cao Engine tab

[Process priority]

Set CAO engine process priority. The order of priority is as follows.

REAL TIME > HIGH > **NORMAL** > IDEL

[Local ID]

Set language ID to be used.

[Log type]

Select CAO.exe log output type. Following are possible log output type.

Table 6-1 Log type

Output destination	Remarks
Console	Output to console
Message Box	Output to a message box. (When service starts.)
Event Viewer	Output to the event viewer. (When service starts.)
Debug Viewer	Debug output.
Text File	Output to the specified text file.

SysLog	Output to the SysLog server.

[Parameter]

Set the log output parameter.

Parameter differs depending on the log type.

Table 6-1 CaoEngine parameter

Log type	Parameter
Console	(not used)
Message Box	(not used)
Event Viewer	(not used)
Debug Viewer	(not used)
Text File	Text file path of the output destination Example) C:¥CaoLog.txt
SysLog	SysLog server of the output destination Conn=UDP:<IP address of SysLog server> Example) Conn=UDP:192.168.0.1

[Log level]

Set a log output level. (Logs being output will be higher level than the log level specified here.) Choose desired log level from the five shown below. “FATAL” is the most critical, whereas “DEBUG” is the least critical. The default setting is “INFO".

FATAL > ERROR > WARN > **INFO** > DEBUG

[**CAO Model] - Automatic registration of object**

Set whether the Automatic object registration function of the Engine is enabled or disabled.

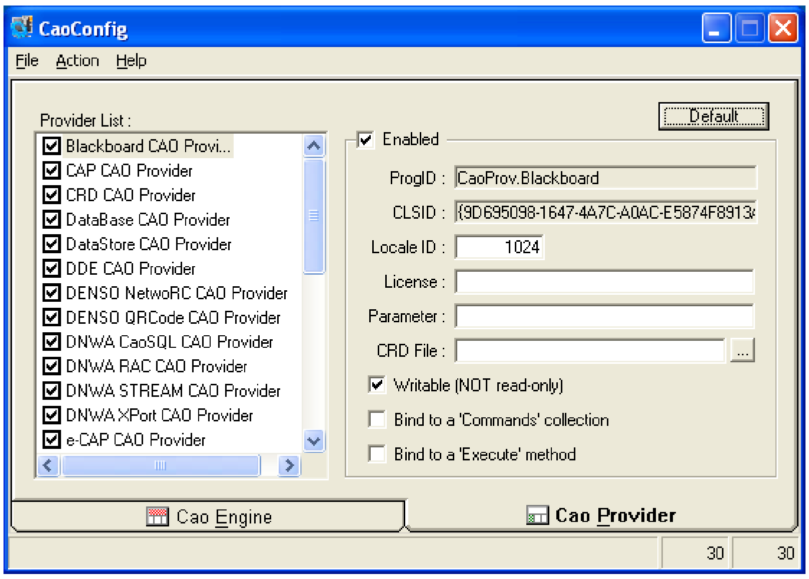
For detailed information about the Automatic object registration function, refer to 2.7.2.

[CRD file]

Specify a CRD file path used in the Automatic object registration function.

6.1.2.1. Cao Provider tab

You can set the operation setting for each provider on the CaoProvider tab.



Provider List :
- ✓ Blackboard CAO Provi...
- ✓ CAP CAO Provider
- ✓ CRD CAO Provider
- ✓ DataBase CAO Provider
- ✓ DataStore CAO Provider
- ✓ DDE CAO Provider
- ✓ DENSO NetwoRC CAO Provider
- ✓ DENSO QRCode CAO Provider
- ✓ DNWA CaoSQL CAO Provider
- ✓ DNWA RAC CAO Provider
- ✓ DNWA STREAM CAO Provider
- ✓ DNWA XPort CAO Provider
- ✓ e-CAP CAO Provider

✓ Enabled

ProgID :	CaoProv.Blackboard
CLSID :	{9D695098-1647-4A7C-A0AC-E5874F8913
Locale ID :	1024
License :	
Parameter :	
CRD File :	

✓ Writable (NOT read-only)

☐ Bind to a 'Commands' collection

☐ Bind to a 'Execute' method

Figure6-5 Cao Provider tab

Cao Provider tab contains following items.

[Provider list]

Display the list of the installed provider. To enable a provider, select a check box.

[Enable] – Enable/Disable setting

Set availability of the provider. Selecting this checkbox enables the provider.

[Prog ID] – Program ID

Display the Program ID of the provider.

[CLSID] – Class ID

Display the Class ID of the provider.

[Locale ID]

Set the Locale ID of the provider.

[License]

Set the license of the provider.

[Parameter]

Set the parameter of the provider.

[CRD file]

Set the path to the CRD file used at the CRD file switch function of each provider. For detailed information, see 2.7.1.

[Writable (NOT read-only)] – **Read-only setting**

Selecting this checkbox will set the provider to Read-only.

[**Bind to a 'Commands' collection]** – **Commands collection method dynamical addition function**

Set whether the commands collection method dynamical addition function of CaoCommands is enabled or disabled. For detailed information, see 2.7.3.

[Bind to a 'Execute' method] – **Execute method dynamical addition function**

Set whether the execute method dynamical addition function of CaoController::Execute is enabled or disabled. For detailed information, see 2.7.3.

[Run as a local server (Dllhost)]

Run a provider on the different process (dllhost.exe) from CAO.

Appendix A. Appendix

Appendix A.1. CAO engine function list

◆ CaoEngine Object-engine

Class	Property, method, and event		Description	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoEngine	EngineStatus	P	Acquisition of engine status object	R		Object: IcaoEngineStatus	
(class number: 0)	Workspaces	P	Acquisition of workspace collection	R		Collection: ICaoWorkspaces	
	AddWorkspace	M	Addition of workspace object	S	Workspace name: BSTR, [Option:BSTR]	Object: ICaoWorkspace	The same function as CaoWorkspaces::Add().
	Execute	M	Execution of enhancing command	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function expansion (reserved)
Meaning of a symbol	M:Method P:Property E:Event			(note 1)	･Arguments enclosed by square brackets [] can be omitted. ･The default value of BSTR type data's omittable argument is NULL. ･The default value of Numeric type data's omittable argument is 0. ･The default value of VARIANT type data's omittable argument is VT_ERROR.		

(note 1)The value of a symbol is as follows. Only the method and the property of W attribute are influenced from ON/OFF of the Access restriction function of CAO engine (writing restriction function).
R-Read: Read the status and the configuration of the controller or the provider or the engine.
W-Write: Change the controller's status and configuration. However, Execute method is classified as S attribute because Execute method depends on the content of the command. The writing restriction is implemented in the provider, if necessary.
S-Setup: Change the provider or the status of the engine and the configuration.

◆ CaoWorkspace(s) Object-workspace

Class	Property, method, and event		Description	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoWorkspaces	Count	P	Acquisition of number of collections	R		Number of collections: Long	
(class number: 1)	Add	M	Addition of workspace object	S	Workspace name:BSTR, [Option:BSTR]	Object: ICaoWorkspace	The workspace name is any character string. When the workspace name is omitted (NULL specification), a unique name is automatically given.
	Clear	M	Release of all workspace objects	S			
	IsMember	M	Registration confirmation of workspace object	R	Workspace name/Number:VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of workspace object	R	Workspace name/Number:VARIANT	Object: ICaoWorkspace	Default member
	Remove	M	Opening of workspace object	S	Workspace name/Number:VARIANT		
CaoWorkspace	Controllers	P	Acquisition of controller collection	R		Collection: ICaoControllers	
(class number: 2)	Index	P	Acquisition of workspace number	R		Workspace number: Long	
	Name	P	Acquisition of workspace name	R		Workspace name: BSTR	
	ProviderNames	P	Acquisition of provider name list	R	[Option:BSTR]	Provider name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of options : Filter condition..
	AddController	M	Addition of controller object	S	Controller-name:BSTR, Provider name:BSTR, [Machine name:BSTR, [Option:BSTR]]	Object: ICaoController	The same function as CaoControllers::Add().
	Execute	M	Execution of enhancing command	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function expansion (reserved)

Class	Property, method, and event	Description	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
Value of a symbol	M:Method P:Property E:Event			･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data’s omittable argument is NULL. ･The default value of Numeric type data’s omittable argument is 0. ･The default value of VARIANT type data’s omittable argument is VT_ERROR.		

◆ CaoController(s) Object-controller

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoControllers	Count	P	Acquisition of number of collections	R		Number of collections: Long	
(class number: 3)	Add	M	Addition of controller object	S	Controller-name:BSTR, Provider name:BSTR, [Machine name:BSTR], [Option:BSTR]	Object: ICaoController	Controller-name is any character string. Example of option: Time-out value, Event permission Event permission: @EventEnable[=True/False] Default value is "True"
	Clear	M	Release of all controller objects	S			
	IsMember	M	Registration confirmation intended for controller	R	Controller-name/Number:VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of controller object	R	Controller-name/Number:VARIANT	Object: ICaoController	Default member
	Remove	M	Opening of controller object	S	Controller-name/Number:VARIANT		
CaoController	Attribute	P	Acquisition of attribute	R		Attribute: Long	
(class number: 4)	CommandNames	P	Acquisition of expansion board name list	R	[Option：BSTR]	Command name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Commands	P	Acquisition of command collection	R		Collection: ICaoCommands	
	ExtensionNames	P	Acquisition of command name list	R	[Option:BSTR]	Expansion board name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Extensions	P	Acquisition of expansion board collection	R		Collection: ICaoExtensions	
	FileNames	P	Acquisition of file name list	R	[Option:BSTR]	File name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Files	P	Acquisition of file collecti	R		Collection: ICaoFiles	File collection of root directory.

Class	Property, method, and event		Explanation	R/W	Argument of function IN	Argument of function OUT RETVAL	Remarks
			on				
	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of controller number	R		Controller number: Long	
	Name	P	Acquisition of controller name	R		Controller name: BSTR	
	RobotNames	P	Acquisition of robot name list	R	[Option:BSTR]	Robot name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Robots	P	Acquisition of robot collection	R		Collection: ICaoRobots	
	Tag	P	Tag	R/S	Tag data:VARIANT	Tag data: VARIANT	
	TaskNames	P	Acquisition of task name list	R	[Option:BSTR]	Task name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Tasks	P	Acquisition of task collection	R		Collection: ICaoTasks	
	VariableNames	P	Acquisition of Variable name list	R	[Option:BSTR]	Variable name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Variables	P	Acquisition of variable collection	R		Collection: ICaoVariables	
	AddCommand	M	Addition of command object	S	Command name:BSTR, [Option:BSTR]	Object: ICaoCommand	The same function as CaoCommands::Add().
	AddExtension	M	Addition of expansion board object	S	Expansion board name:BSTR,[Option:BSTR]	Object: ICaoExtension	The same function as CaoExtensions::Add().
	AddFile	M	Addition of file object	S	File name:BSTR, [Option:BSTR]	Object: ICaoFile	The same function as CaoFiles::Add().
	AddRobot	M	Addition of robot object	S	Robot name:BSTR, [Option:BSTR]	Object: ICaoRobot	The same function as CaoRobots::Add().

Class	Property, method, and event		Explanation	R/W	Argument of function IN	OUT RETVAL	Remarks
	AddTask	M	Addition of task object	S	Task name:BSTR, [Option:BSTR]	Object: ICaoTask	The same function as CaoTasks::Add().
	AddVariable	M	Addition of variable object	S	Variable name:BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Execute	M	Execution of enhancing co mmand	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function expansion
	GetMessage	M	Acquisition of message	R		Message: ICaoMessage	The message is acquired from the message qu eue in the engine. Message can be acquired o nly when the OnMessage event function is in valid.
	OnMessage	E	Message reception event	R	Message:ICaoMessage		･This realizes the following call. [Provider(controller)]→[Engine]→[Client] - A negative range of the message number ha s been reserved with ORiN. - The nullification of the event is specified by the option of AddController().
Value of a symb ol	M:Method P:Property E:Event				･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable ar gument is NULL. ･The default value of Numeric type data's omittable argument is 0. ･The default value of VARIANT type data's omittabl e argument is VT_ERROR.		

◆ CaoExtension(s) Object-expansion board

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoExtensions (class number: 5)	Count	P	Acquisition of number of collections	R		Number of collections: Lo ng	
	Add	M	Addition of expansion boa rd object	S	Expansion board name:BS TR, [Option:BSTR]	Object: ICaoExtension	
	Clear	M	Release of all expansion b oard objects	S			
	IsMember	M	Registration confirmation o f expansion board object	R	Expansion board name/Nu mber:VARIANT	Check result: VARIANT_B OOL	
	Item	M	Acquisition of expansion b oard object	R	Expansion board name/Nu mber:VARIANT	Object: ICaoExtension	Default member
	Remove	M	Opening of expansion boar d object	S	Expansion board name/Nu mber:VARIANT		
CaoExtension (class number: 6)	Attribute	P	Acquisition of attribute	R		Attribute: Long	
	Help	P	Help	R		Help character string: BST R	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of expansion b oard number	R		Expansion board number: Long	
	Name	P	Acquisition of expansion b oard name	R		Expansion board name: B STR	
	Tag	P	Tag	R/S	Tag data:VARIANT	Tag data: VARIANT	
	VariableNames	P	Acquisition of Variable na me list	R	[Option:BSTR]	Variable name list: VARIA NT(VT_VARIANT|VT_AR RAY)	Example of option : Filter condition
	Variables	P	Acquisition of variable col lection	R		Collection: ICaoVariables	
	AddVariable	M	Addition of variable object	S	Variable name:BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Execute	M	Execution of enhancing co mmand	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function expansion
Value of a symb	M:Method				･Arguments enclosed with square brackets [] can be		

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
ol	P:Property E:Event			omitted. ･The default value of BSTR type data’s omittable argument is NULL. ･The default value of Numeric type data’s omittable argument is 0. ･The default value of VARIANT type data’s omittable argument is VT_ERROR.		

◆ CaoFile(s) Object-file

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoFiles	Count	P	Acquisition of number of collections	R		Number of collections: Long	
(class number: 7)	Add	M	Addition of file object	S	File name:BSTR, [Option:BSTR]	Object: ICaoFile	File creation: @Create = numerical value default value is 0 0 : create a file Other than 0 : not create a file When the provider is read-only, if @Create is other than 0, a writing error occurs.
	Clear	M	Release of all file objects	S			
	IsMember	M	Registration confirmation of file object	R	File name/Number:VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of file object	R	File name/Number:VARIANT	Object: ICaoFile	Default member
	Remove	M	Opening of file object	S	File name/Number:VARIANT		
CaoFile	Attribute	P	Acquisition of attribute	R		Attribute: Long	
(class number: 8)	DateCreated	P	At the date	R		At the date: VARIANT.	
	DateLastAccessed	P	The final access date	R		The final access date: VARIANT	
	DateLastModified	P	The final change date	R		The final change date: VARIANT	
	FileNames	P	Acquisition of file name list	R	[Option:BSTR]	File name list: VARIANT (VT_VARIANT|VT_ARRAY)	Example of option : Filter condition When the attribute is a directory, the child file name list is returned.
	Files	P	Acquisition of file collection	R		Collection: ICaoFiles	
	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of file number	R		File number: Long	
	Name	P	Acquisition of file name	R		File name: BSTR	

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
	Path	P	Acquisition of path	R		Path name: BSTR	Absolute path. The file name is not included. The last delimiter is included.
	Size	P	Acquisition of size of file	R		Size of file: Long	
	Tag	P	Tag	R/S	Tag data:VARIANT	Tag data: VARIANT	
	Type	P	Acquisition of type of file	R		File type: BSTR	
	Value	P	Content of file	R/W	Data:VARIANT	Data: VARIANT	Default member
	VariableNames	P	Acquisition of Variable na me list	R	[Option:BSTR]	Variable name list: VARIA NT(VT_VARIANT|VT_AR RAY)	Example of option : Filter condition
	Variables	P	Acquisition of variable col lection	R		Collection: ICaoVariables	
	AddFile	M	Addition of file object	S	File name:BSTR, [Option:BSTR]	Object: ICaoFile	The same function as CaoFiles::Add().
	AddVariable	M	Addition of variable object	S	Variable name:BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Copy	M	Copy	W	Copy destination file nam e:BSTR [Option:BSTR]		
	Delete	M	Deletion	W	[Option:BSTR]		
	Execute	M	Execution of enhancing co mmand	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function expansion
	Move	M	Movement	W	Moving destination file na me:BSTR [Option:BSTR]		
	Run	M	Task creation	W	[Option:BSTR]	Task name: BSTR	
Value of a symb ol	M:Method P:Property E:Event				･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable ar gument is NULL. ･The default value of Numeric type data's omittable argument is 0. ･The default value of VARIANT type data's omittabl e argument is VT_ERROR.		

◆ CaoRobot(s) Object-robot

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoRobots	Count	P	Acquisition of number of collections	R		Number of collections: Long	
(class number: 9)	Add	M	Addition of robot object	S	Robot name:BSTR, [Option:BSTR]	Object: ICaoRobot	
	Clear	M	Release of all robot objects	S			
	IsMember	M	Registration confirmation of robot object	R	Robot name/Number:VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of robot object	R	Robot name/Number:VARIANT	Object: ICaoRobot	Default member
	Remove	M	Opening of robot object	S	Robot name/Number:VARIANT		
CaoRobot	Attribute	P	Acquisition of attribute	R		Attribute: Long	
(class number: 10)	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of robot number	R		Robot number: Long	
	Name	P	Acquisition of CaoRobot name	R		Robot name: BSTR	
	Tag	P	Tag	R/S	Tag data:VARIANT	Tag data: VARIANT	
	VariableNames	P	Acquisition of Variable name list	R	[Option:BSTR]	Variable name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Variables	P	Acquisition of variable collection	R		Collection: ICaoVariables	
	Accelerate	M	Refer to the specification of the ACCEL sentence of SLIM.	W	Axis Number:long, Acceleration:float, [Deceleration:float]		Axis number 1: Acceleration/Deceleration of tool end. 0: Acceleration/Deceleration of all axis Other number: Acceleration/Deceleration of specified axis.

Class	Property, method, and event		Explanation	R/W	Argument of function IN	Argument of function OUT RETVAL	Remarks
	AddVariable	M	Addition of variable object	S	Variable name: BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Change	M	Refer to the specification of the CHANGE sentence of SLIM.	W	Hand name: BSTR		
	Chuck	M	Refer to the specification of the GRASP sentence of SLIM.	W	[Option:BSTR]		
	Drive	M	Refer to the specification of the DRIVE sentence of SLIM.	W	Axis Number: long, Motion distance: float, [Motion option: BSTR]		It is impossible to move multiple axis at the same time, just like SLIM does. In that case, MOVE is recommended to be used.
	Execute	M	Execution of enhancing co mmand	S	Command: BSTR [Parameter:VARIANT]	Result: VARIANT	For function expansion
	GoHome	M	Refer to the specification of the GOHOME sentence of SLIM.	W	[Option:BSTR]		
	Hold	M	Refer to the specification of the HOLD sentence of SLIM.	W	[Option:BSTR]		In SLIM, this sentence means “suspend of rob ot program”. In CAO, this command means “suspend of robot motion”.
	Halt	M	Refer to the specification of the HALT sentence of SLIM.	W	[Option:BSTR]		In SLIM, this sentence means “force stop of r obot program”. In CAO, this command means “force stop of robot motion”.
	Move	M	Refer to the specification of the MOVE sentence of SLIM.	W	Interpolation specification: long, Posed row:VARIANT, [Motion option:BSTR]		Interpolation specification 1: PTP 2: Linear 3: Circular - Pose specification depends on the manufactu rer’s specification. Array is used for the circul ar interpolation specification. Default value of motion option is "".

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
	Rotate M	Refer to the specification of the ROTATE sentence of SLIM.	W	Rotation surface: VARIAN T, Angle:float, Rotation center:VARIANT, [Motion option: BSTR]		Rotation surface specification depends on the manufacturer's specification.
	Speed M	Refer to the specification of the SPEED/JSPEED se ntence of SLIM.	W	Axis Number: long, Speed: float		Axis number 1: Speed of tool end 0: All axis speeds Other number: Specified axis speed.
	Unchuck M	Refer to the specification of the RELEASE sentence of SLIM.	W	[Option: BSTR]		"Release" of SLIM cannot be used because it is a reserved word. Therefore, use Chuck/Un chuck instead.
	Unhold M	Refer to the specification of the HOLD sentence of SLIM.	W	[Option: BSTR]		Since the Hold sentence of SLIM means "sus pend of program", there is no command for r esume. In CAO, HOLD command means "sus pend of robot motion". "Unhold" is used to r esume the robot motion suspended by HOLD command.
Value of a symb ol	M:Method P:Property E:Event			･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable ar gument is NULL. ･The default value of Numeric type data's omittable argument is 0. ･The default value of VARIANT type data's omittabl e argument is VT_ERROR.		

◆ CaoTask(s) Object-task

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoTasks (class number: 11)	Count	P	Acquisition of number of collections	R		Number of collections: Long	
	Add	M	Addition of task object	S	Task name: BSTR, [Option:BSTR]	Object: ICaoTask	
	Clear	M	Release of all task objects	S			
	IsMember	M	Registration confirmation of task object	R	Task name/Number: VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of task object	R	Task name/Number: VARIANT	Object: ICaoTask	Default member
	Remove	M	Opening of task object	S	Task name/Number: VARIANT		
CaoTask (class number: 12)	Attribute	P	Acquisition of attribute	R		Attribute: Long	
	FileName	P	Acquisition of correspondence file name	R		File name: BSTR	
	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of task number	R		Task number: Long	
	Name	P	Acquisition of task name	R		Task name: BSTR	
	Tag	P	Tag	R/S	Tag data: VARIANT	Tag data: VARIANT	
	VariableNames	P	Acquisition of Variable name list	R	[Option:BSTR]	Variable name list: VARIANT(VT_VARIANT|VT_ARRAY)	Example of option : Filter condition
	Variables	P	Acquisition of variable collection	R		Collection: ICaoVariables	
	AddVariable	M	Addition of variable object	S	Variable name: BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Delete	M	Deletion of task	W	[Option:BSTR]		
	Execute	M	Execution of enhancing command	S	Command: BSTR [Parameter: VARIANT]	Result: VARIANT	For function expansion

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
	Start M	Beginning of task	W	Mode: long, [Option:BSTR]		Mode 1: One-cycle execution 2: Continuous execution 3: One-step forward 4: One-step back Example of option: Start position
	Stop M	Stop of task	W	Mode: long, [Option:BSTR]		Mode 0: Default stop 1: Instantaneous stop 2: Step-stop 3: Cycle-stop 4: Initialization stop (note) Default stop is any stop of above 1 to 4 stop.
Value of a symbol	M:Method P:Property E:Event			･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable argument is NULL. ･The default value of Numeric type data's omittable argument is 0. ･The default value of VARIANT type data's omittable argument is VT_ERROR.		

◆ CaoVariable(s) Object-variable

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoVariables (class number: 13)	Count	P	Acquisition of number of collections	R		Number of collections: Long	
	Add	M	Addition of variable object	S	Variable name: BSTR, [Option:BSTR]	Object: ICaoVariable	
	Clear	M	Release of all variable objects	S			
	IsMember	M	Registration confirmation of variable object	R	Variable name/Number:VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of variable object	R	Variable name/Number:VARIANT	Object: ICaoVariable	Default member
	Remove	M	Opening of variable object	S	Variable name/Number:VARIANT		
CaoVariable (class number: 14)	Attribute	P	Acquisition of attribute	R		Attribute: Long	
	DateTime	P	Acquisition of time and date stamp (date)	R		Time and date stamp: VARIANT	
	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of variable number	R		Variable number: Long	
	Microsecond	P	Acquisition of time and date stamp (microsecond)	R		Time and date stamp: Long	
	Name	P	Acquisition of Variable name	R		Variable name: BSTR	The system Variable name starts by @.
	Tag	P	Tag	R/S	Tag data:VARIANT	Tag data: VARIANT	
	Value	P	Acquisition of value	R/W	Value:VARIANT	Value:VARIANT	Default member
Value of a symbol	M:Method P:Property E:Event				･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable argument is NULL. ･The default value of Numeric type data's omittable argument is 0.		

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
				･The default value of VARIANT type data’s omittable argument is VT_ERROR.		

◆ CaoCommand(s) Object-command

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoCommands	Count	P	Acquisition of number of collections	R		Number of collections: Long	
(class number: 15)	Add	M	Addition of command object	S	Command name: BSTR, [Option:BSTR]	Object: ICaoCommand	
	Clear	M	Release of all command objects	S			
	IsMember	M	Registration confirmation of command object	R	Command name/Number: VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of command object	R	Command name/Number: VARIANT	Object: ICaoCommand	Default member
	Remove	M	Opening of command object	S	Command name/Number: VARIANT		
	ExecuteComplete	E	Command execution completion event			Command number: Long	The result is acquired in the Result property.
CaoCommand	Attribute	P	Acquisition of attribute	R		Attribute: Long	
(class number: 16)	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID:VARIANT	ID:VARIANT	
	Index	P	Acquisition of command number	R		Command number: Long	
	Name	P	Acquisition of command name	R		Command name: BSTR	
	Parameters	P	Command parameter	R/W	Command parameter: VARIANT	Command parameter: VARIANT	
	Result	P	Execution result of Execute() immediately before	R		Execution result: VARIANT	
	State	P	Acquisition of state	R		State: Long	The first bit in state 0: Waiting 1: Processing Other bits: depend on the provider used.
	Tag	P	Tag	R/S	Tag data: VARIANT	Tag data: VARIANT	
	Timeout	P	Time-out	R/W	Time-out: long	Time-out: Long	
	Cancel	M	Cancellation of processing				

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
		command				
	Execute M	Execution of command		Mode: long		Mode 0: Synchronous execution 1: Asynchronous execution. In asynchronous e xecution, S_FALSE is returned.
Value of a symb ol	M:Method P:Property E:Event			･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable ar gument is NULL. ･The default value of Numeric type data's omittable argument is 0.		

◆ CaoMessage Object-message

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoMessage (class number: 17)	DateTime	P	Creation date	R		Creation date: VARIANT.	
	Description	P	Explanation	R		Explanation: BSTR	
	Destination	P	Destination address	R		Destination address: BSTR	
	Number	P	Message number	R		Message number: Long	
	SerialNumber	P	Message sequential number	R		Message sequential number: Long	Sequential number which is automatically added to Engine ranging from 0 to "LONG_MAX". When it reaches LONG_MAX, it is cleared to 0.
	Source	P	Sending origin	R		Message text: VARIANT	
	Value	P	Message text	R		Message text: VARIANT	
	Clear	M	Clearness of message	W			
	Reply	M	Reply of message	W			
Value of a symbol	M:Method P:Property E:Event			(note 1)	･Arguments enclosed with square brackets [] can be omitted. ･The default value of BSTR type data's omittable argument is NULL. ･The default value of Numeric type data's omittable argument is 0.		

◆ CaoEngineStatus Object-engine status

Class	Property, method, and event		Explanation	R/W	Argument of function		Remarks
					IN	OUT RETVAL	
CaoEngineStatus (class number: 18)	CurrentDateTime	P	Current date	R		Current date: VARIANT	
	ComputerName	P	Computer name	R		Computer name: BSTR	
	ObjectCounts	P	Number of objects of each class	R	Class name/number: VARIANT	Number of objects: Long	
	StartDateTime	P	Beginning date	R		Beginning date: VARIANT	
	Values	P	(reserved for the future)	R	Status name/number: VARIANT	Data: VARIANT	
	Version	P	Acquisition of version of Engine	R		Version: BSTR	<Major Ver.>.<Minor Ver.>.<Revision>
Value of a symbol	M:Method P:Property E:Event			(note 1)	·Arguments enclosed with square brackets [] can be omitted. ·The default value of BSTR type data's omittable argument is NULL. ·The default value of Numeric type data's omittable argument is 0.		

Appendix A.2. Glossary

- CAO (Controller Access Object)

  CAO is "Standard program interface" that offers a common interface and the function to the client application and the robot controller.

- CAO interface (CAO Interface)(API)

  An interface of the client application side. This enables client application to handle a robot controller provided by CAO. “CAO API” indicates this interface.

- CAO engine(CAO Engine)

  A common function to the CAO provider and the CAO interface are offered to the client with the CAO engine by middleware (EXE) that mounts the interface of CAO. A common function to the CAO provider includes the collection function, the message output, and the CRD switch function, etc.

- CAO provider(CAO Provider)

  CAO provider is DLL where the part that depended on the robot makers is mounted by the subordinate position module of CAO.

- CAO provider template(CAO Provider template)

  C++ template that supports mounting CAO Provider. This template is generated with ProviderWizard automatically.

- CAP(Controller Access Protocol)

  It is "Communication protocol for the Internet" to access CAO provider by way of the Internet. In CAP, the message to access the CAO provider that uses SOAP is defined.

- CAP provider(CAP Provider)

  It is CAO provider to generate, and to send and receive the CAP message.

- CAP listener(CAP Listener)

  The CAP message is received, and the server to operate CAO on a remote machine program.

- CAP message(CAP Message)

  SOAP message defined by CAP.

- CRD　(Controller Resource Definition)

  Data schema that defines the data general-purpose to express resource information on a robot different depending on the maker and the model format.

- CRD resource(CRD Resource)

  A robot resource expressed by CRD.

- CRD data(CRD Data)

  CRD-based robot resource data that is written in XML.

- CRD Dataskema(CRD Data Schema)

  A schema that defines the format of the CRD data.

- CRD file(CRD File)

  An XML file where the CRD data is described.

- CRD provider(CRD Provider)

  A provider to access the CRD data.

- RAC (Robot Action Command)

  An integrated robot language that send and receive commands between the client application and the robot.