

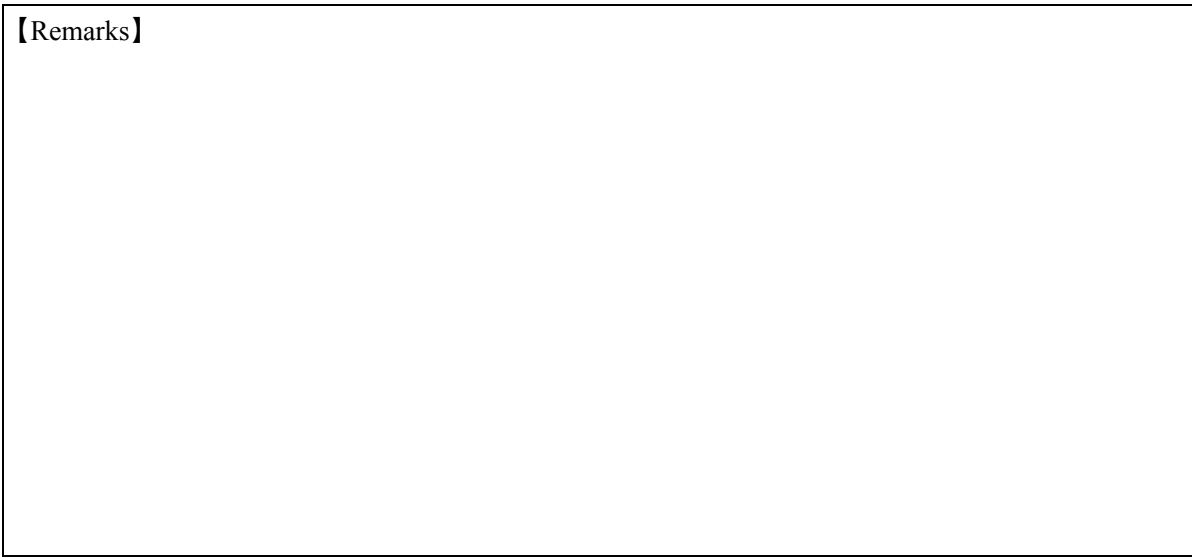
# ORiN2

## Programming guide

Version 1.0.12.0

September 7, 2012

【Remarks】



**【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

## Contents

1. Introduction.....	5
2. Implementation of CAO client.....	7
2.1. Outline.....	7
2.2. Basic knowledge .....	8
2.2.1. Early Binding and Rate Binding .....	8
2.2.2. Creation and management of object.....	10
2.2.3. Asynchronous processing.....	12
2.2.4. Variable type used with COM .....	12
2.2.5. Notation of data .....	18
2.2.6. Log output .....	19
2.2.7. Error code.....	19
2.3. An installation state of ORiN2 SDK .....	22
2.4. CAO client implementation.....	22
2.4.1. Open controller .....	22
2.4.2. Retrieve and set variable.....	24
2.4.3. Retrieve and set system variable.....	25
2.4.4. Event processing.....	25
2.5. Developing client with VB.NET .....	28
2.5.1. RCW registration to GAC .....	28
2.5.2. VB.NET project setting .....	29
2.5.3. Notes on object deletion .....	29
2.5.4. Method of handling event .....	30
2.6. Developing client with other languages.....	31
2.6.1. Client development with C++.....	31
2.6.2. Client development with C .....	35
2.6.3. Client development with C#.....	36
2.6.4. Client development with Delphi .....	40
2.6.5. Creating client in LabVIEW.....	42
2.7. Special functions of CAO .....	44
2.7.1. CRD switch function .....	44
2.7.2. Automatic object registration function .....	44
2.7.3. Dynamic method addition function.....	45
3. CRD programming guide.....	47
3.1. Outline.....	47
3.2. Basic knowledge .....	47
3.2.1. What is XML? .....	47

---

3.2.2. DOM.....	48
3.2.3. XML parser.....	48
3.2.4. XML schema.....	48
3.3. CRD file and CRD provider.....	49
3.4. Creating CRD file.....	50
3.4.1. Header and root element creation.....	50
3.4.2. Static data definition.....	50
3.4.3. System configuration definition.....	50
3.4.4. Device capability definition.....	51
3.5. Sample CRD file.....	53
3.5.1. Static data definition example.....	53
3.5.2. System configuration definition example.....	54
3.5.3. Device capability definition example.....	54
<b>4. CAP programming guide.....</b>	<b>57</b>
4.1. Outline.....	57
4.2. Basic knowledge.....	57
4.2.1. SOAP.....	57
4.3. CAP provider and CAP listener.....	58
4.4. Environmental construction.....	59
4.4.1. Setting of server side.....	59
4.4.2. Client side.....	67
4.5. Sample program.....	67
<b>5. Environmental setting.....</b>	<b>69</b>
5.1. Set-up steps DCOM.....	69
5.1.1. Preparation.....	70
5.1.2. Set item in WindowsNT/2000.....	70
5.1.3. Set item in Windows9x.....	75
5.1.4. Set item in Windows XP.....	76
5.1.5. Set item in Windows XP SP2.....	83
5.1.6. Set confirmation of DCOM.....	87
<b>6. CaoConfig.....</b>	<b>90</b>
6.1. Screen composition.....	90
6.1.1. Menu.....	90
6.1.2. Cao Engine tab.....	91

## 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)

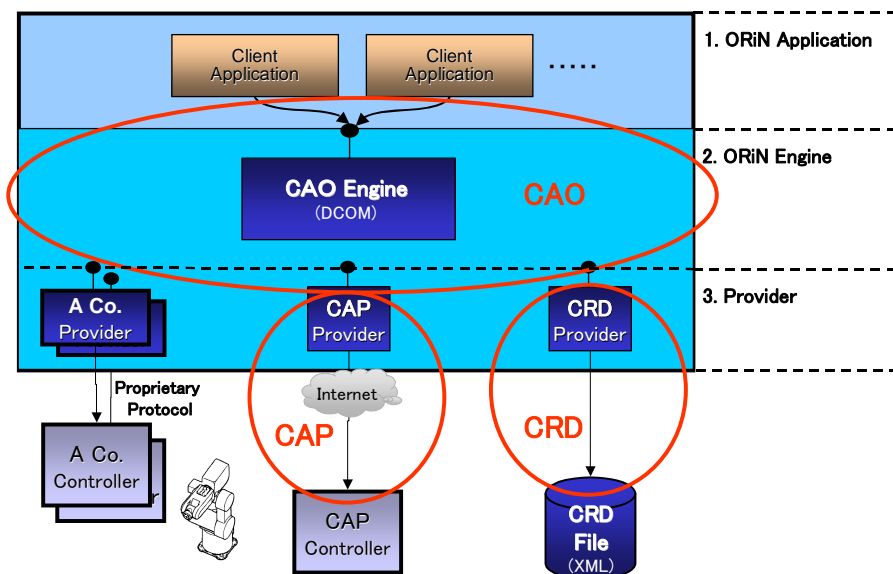


Figure 1-1 Three basic techniques of ORiN2

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.7)	Implementation of CAO client	ORiN application developer
Chapter 3 (p.47)	Introduction to CRD implementation	Robot vender ORiN application developer
Chapter 4 (p. 57)	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.

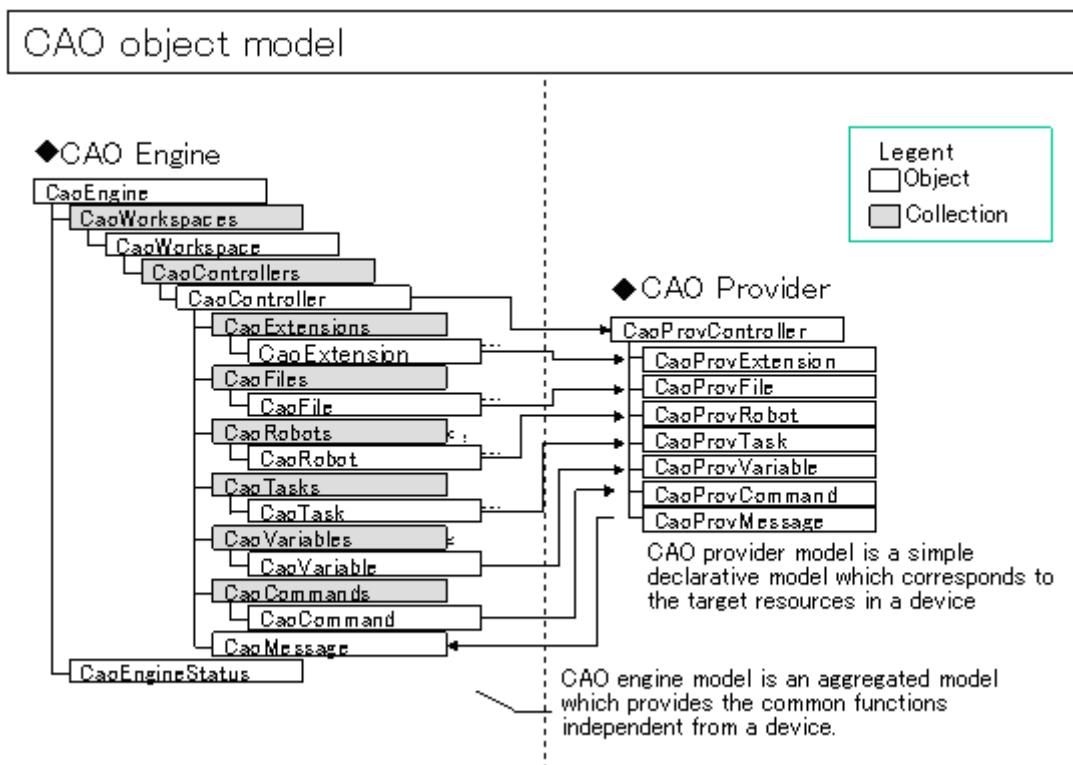


Figure2-1 Object model of CAO

## 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<sup>1</sup>. Followign 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 inquirey 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.

- (1) Select [Project]->[reference setting] in the menubar of VB6.
- (2) When the dialog is displayed as shown in Figure2-2, add the type library "CAO 1.0 type library".

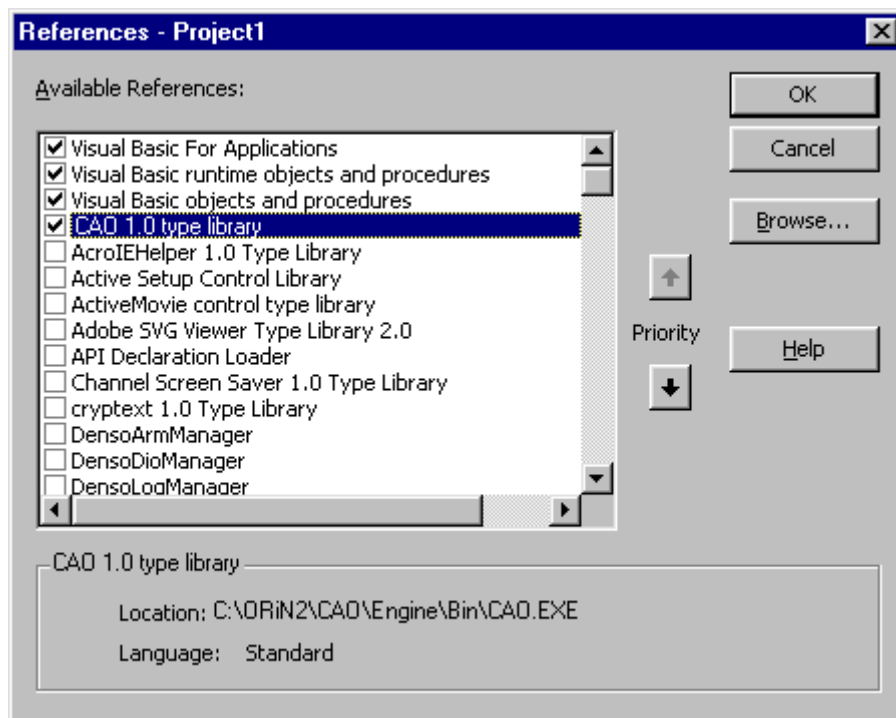


Figure2-2 Reference setting screen of VB

<sup>1</sup> It uses and it explains the early binding in this book.



- (3) 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<sup>2</sup>. The CaoEngine type can be called when the variable is declared, because the reference setting is finished in reference setting of 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 late 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 late 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.

---

<sup>2</sup> In VB.NET, the Set statement is not used.

### 2.2.2. Creation and management of object

For the creation of COM object, New key word 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<sup>3</sup>.

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

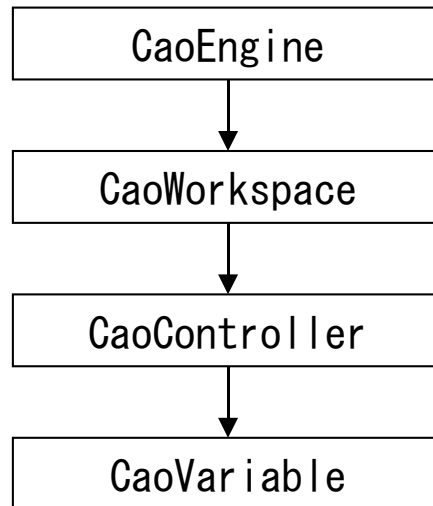
These object need to be created sequentially. Below is the order of creation.

- (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.

---

<sup>3</sup> 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 for 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<sup>4</sup> 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
  
```

<sup>4</sup> 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.

```

End If
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 an 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.<sup>5</sup>

- (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

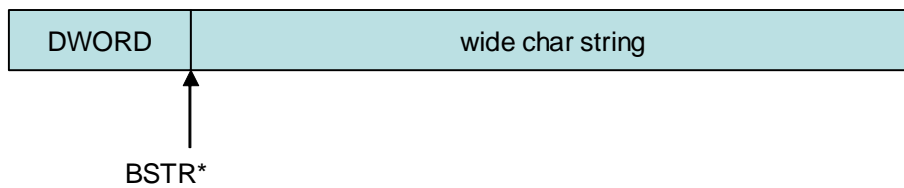
Four variable types, BSTR, VARIANT, SAFEARRAY, and HRESULT are characteristic variable types used in COM, and they are also often used in provider implementation.

#### 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.

<sup>5</sup> It is necessary to set the event procedure by the AddHandler method in VB.NET. Please refer to 2.5.4 for details.

Figure2-4 Structure of BSTR



To substitute value into BSTR, use SysAllocString() to secure 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 datatype, 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 <sup>6</sup>	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 VARIANT type, substitute identifier into vt at first, and then substitute value into union. A macro can be used for this substitution. Followin 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 befor using it. If VariantInit()

<sup>6</sup> To substitute to VT\_BOOL type, instead of TRUE, FALSE of BOOL type, VARIANT\_TRUE and VARIANT\_FALSE need to be used

is used for initialization, the identifier becomes VT\_EMPTY to show that the value is not put in VARIANT.

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 VARIANT 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 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 constructor, 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), an 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, an 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, an 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.

Followin 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<sup>7</sup>.

### 2.2.7. Error code

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

<sup>7</sup> 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.

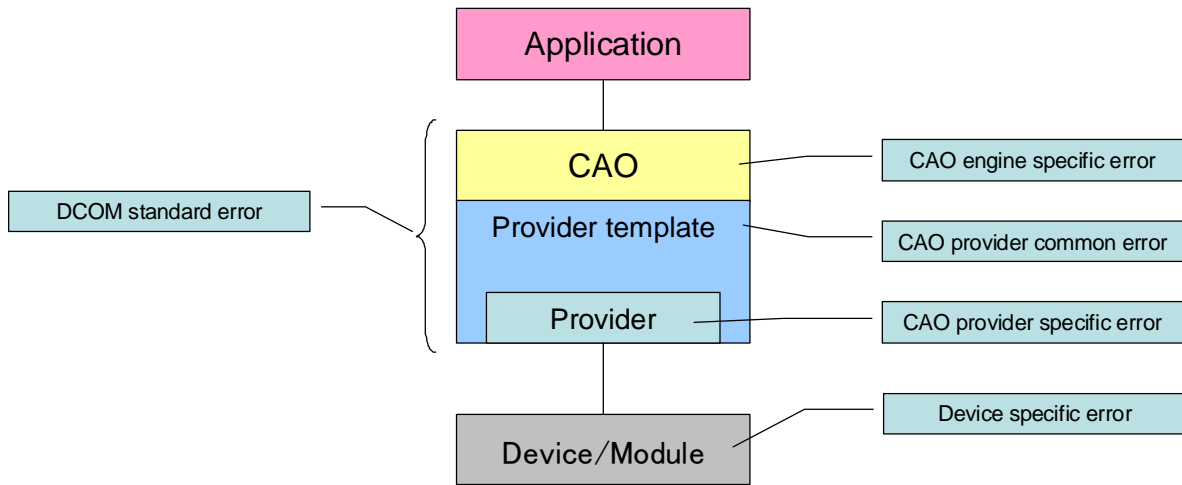


Figure 2-5 Errors generated in ORiN2

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 (0 x 0) with returning logical TRUE.
S_FALSE	0x00000001	Normal finish (0 x 1) 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 of 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.

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 name.
E_CANCEL	0x80000403	The command was canceled.
E_TIMEOUT	0x80000900	Timeout
E_NO_LICENSE	0x80000901	No license.
E_NOT_CONNECTED	0x80000902	Connection is not established.
<del>E_WINDOWS_MASK</del>	<del>0x8090xxxx</del>	<del>Windows standard error codes (16-bit) are stored into xxxx (the least significant two bytes)</del> <del>Ex)</del> <del>Error code: 2 (File not found)</del> <del>→ 0x80900002</del> (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 fourth 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 GetVariable 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 identifier, 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 identifier "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

Followign 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

'Retrieve CA0 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 .

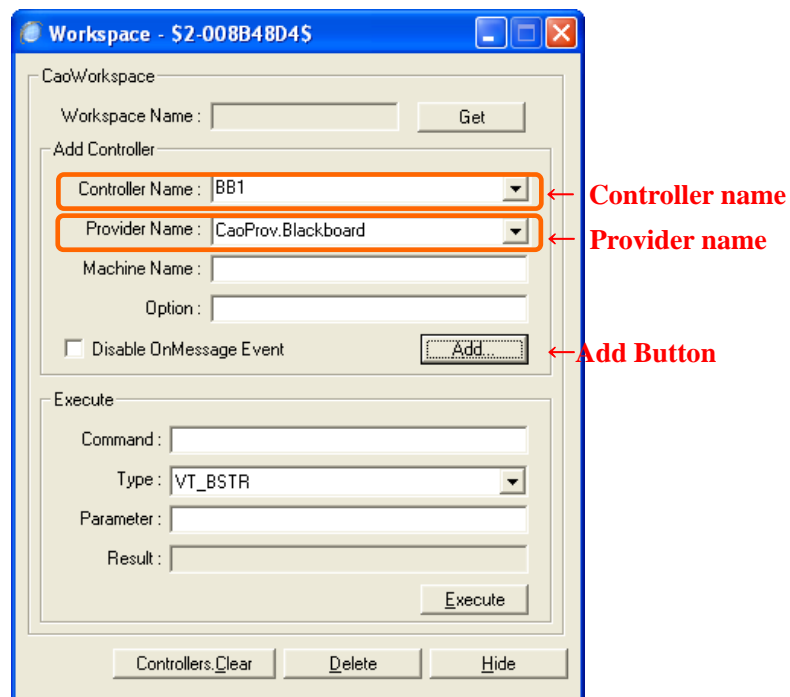


Figure2-7 Connection to Blackboard provider

When you press the [Add] button, and chile 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.

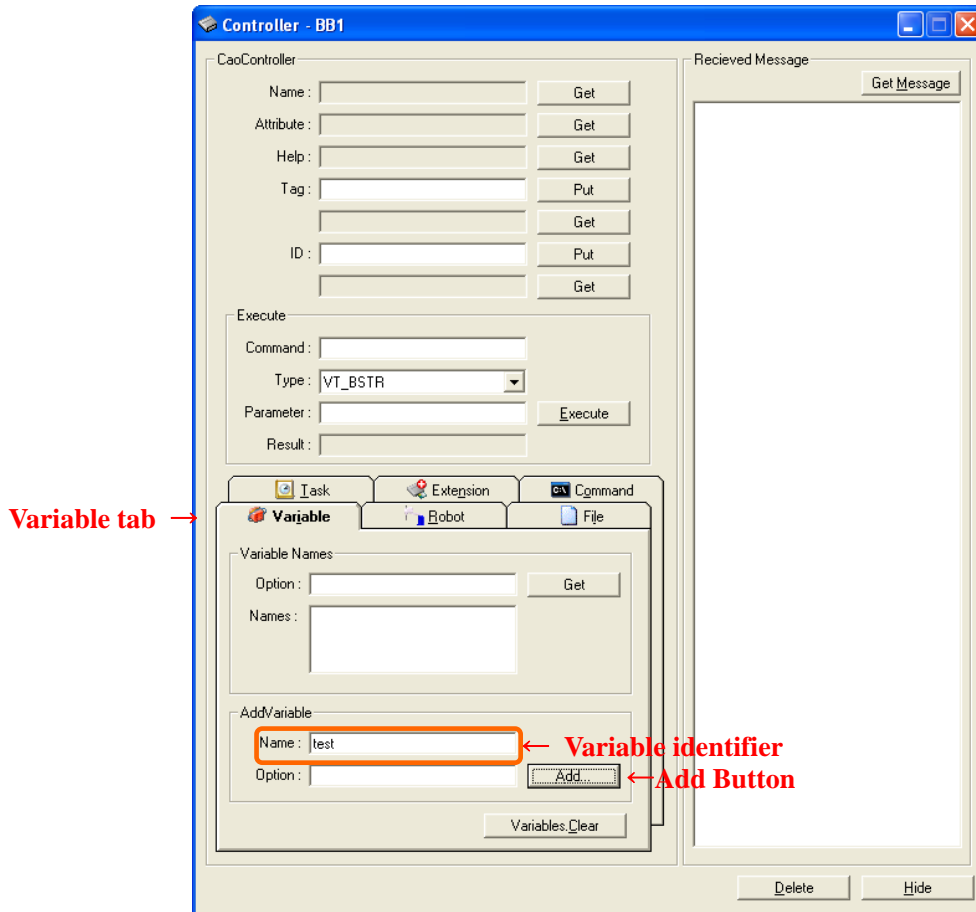


Figure2-8 Addition of variable object

When you press the [Add] button, a chile 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.

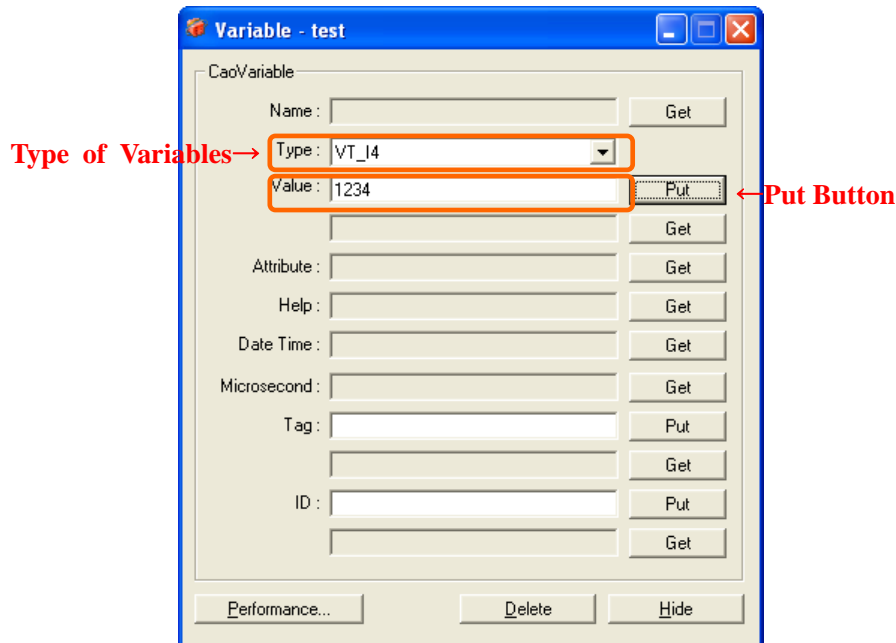


Figure2-9 Writing of variable

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.



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)<sup>8</sup>

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 “Nothing” 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 decrease 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
```

<sup>8</sup> 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) An 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++<sup>9</sup>

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 ohandle 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.

<sup>9</sup> 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](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 CoUninitialize() before the program ends.

Following is an 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 CaoController
    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"<sup>10</sup>.

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

<sup>10</sup> 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 actual 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 CoUninitialize() 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_GaoSQLEngine,
                     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->IptTbl->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 CoUninitialize() 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#

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

is the same as 2.5.

### 2.6.3.1. 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. 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;
```

### 2.6.3.3. 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.

### 2.6.3.4. 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);
```

### 2.6.3.5. 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 OnMessage event handler.

```

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 2-5****ImportSample.cpp**

```

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
            {
                // create a CAO engine
                caoEng = new CaoEngine();
                caoWss = caoEng.Workspaces;
                caoWs = caoWss.Item(0);

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

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

                // register an OnMessage event handler
                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);
    }
}

// OnMessage event handler
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 click "Unit creation" button.  
(Figure2-11) The unit file is created at the specified directory with file name "CAOLib\_TLB.pas".

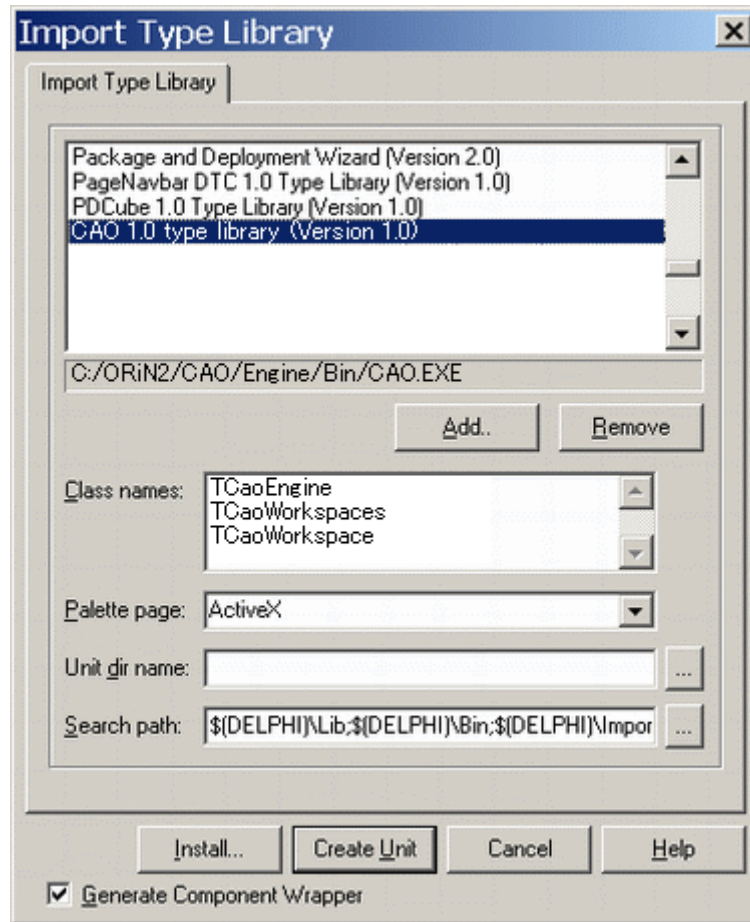


Figure2-11 Read type library screen

- (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 procedure.

### 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 2-6****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('', 'GaoProv.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.

```

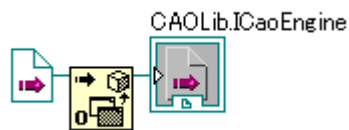
### 2.6.5. Creating client in LabVIEW

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].

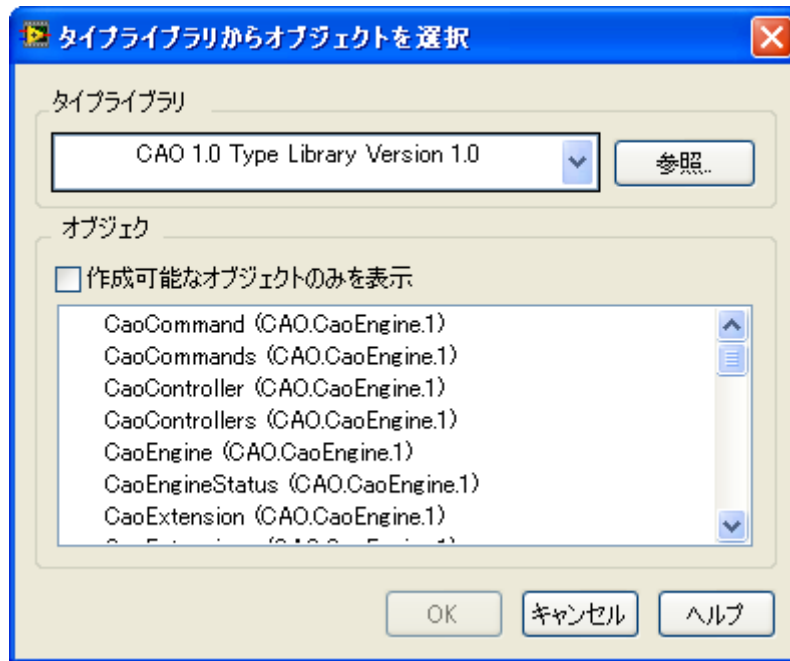
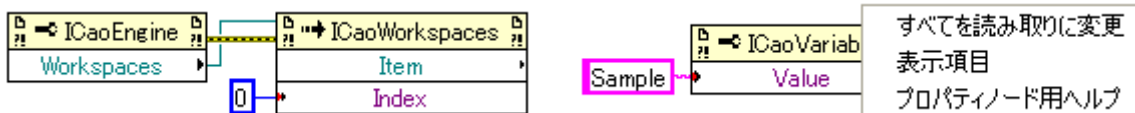


Figure 2-12 Window to select object from type library

- (2) Setup properties, and use methods.

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



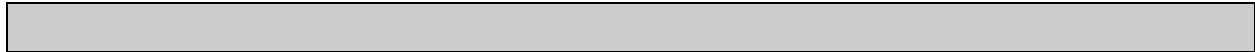
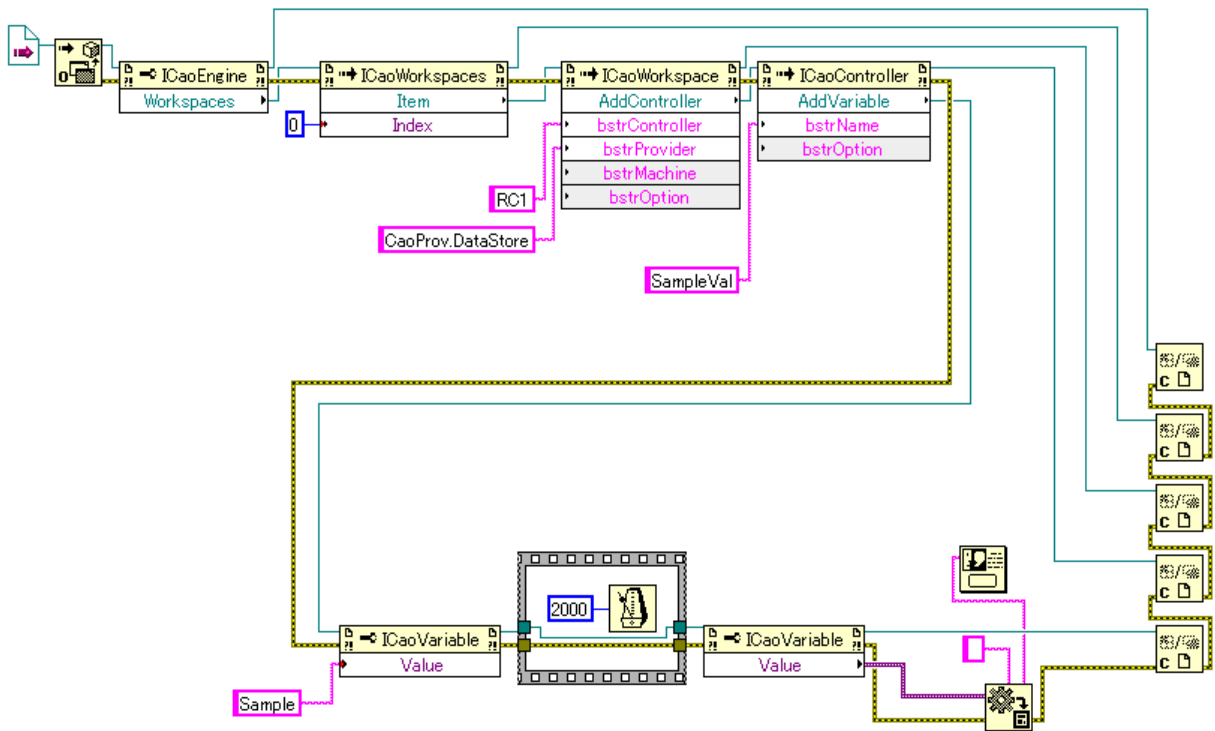
- (3) Close reference

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 referd 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, retron the value "E\_CRDIMPL"<sup>11</sup>.

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 registerd 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.

<sup>11</sup> 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 2-7**

**CRDSample.xml**

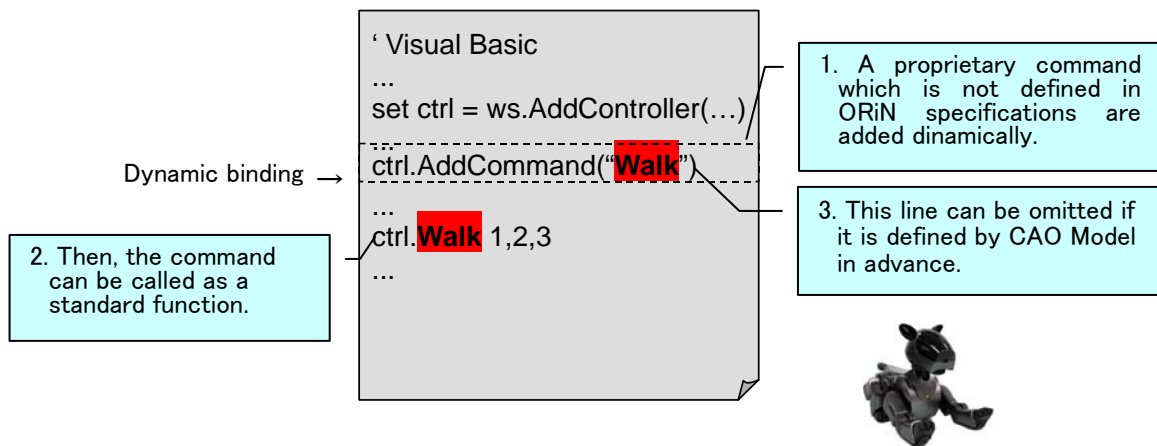
```
<?xml version="1.0" encoding="Shift_JIS"?>
<CRD xsi:schemaLocation="http://www.orin.jp/CRD/CRDSchema CRDSchema.xsd"
      xmlns="http://www.orin.jp/CRD/CRDSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <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.



**Figure2-13 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 proceder 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 auther. 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>RA0</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 element described in XML as objects. If each application differently handles XML object, it is very burdensome for developers. Therefore, W3C established DOM as a unified standard. Another XML API other than DOM is SAX (Simple API for XML), and both of them are widely used.

DOM has several levels, and the higher the level, the newer and the 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 as 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 parser 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 have 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 a description of possible XML document structure. The schema describes the 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.AddVariable("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 schime 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="I0100"/>
</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 a 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 an 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 following.

- 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="I0100"/>
  </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 chiefly 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 by the firewall to communicate by using an original port number. Because the worm etc. using the security hole of DCOM exist in recent years, 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. In SOAP, only it is to provide for the structure of the message exchanged between objects, existing HTTP and SMTP are used for the protocol for the communication. As a result, if HTTP and SMTP can be communicated, using the distributed object technology through the firewall becomes possible by using SOAP.

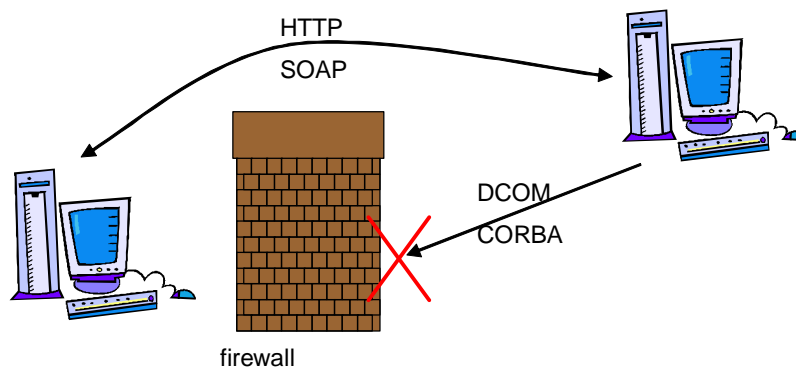


Figure4-1 Image of SOAP

### 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.

CAP provider and CAP outline of operation of the listener Figure4-2 Is shown. The CAP provider is a provider as shown in figure to generate and to transmit the message to access a remote provider. This message is transmitted to the CAP listener to have opened Web server to the public. The CAP listener is an interface to receive, to analyze the semi-message, and to call the COM component (CAO provider) in CAP.

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

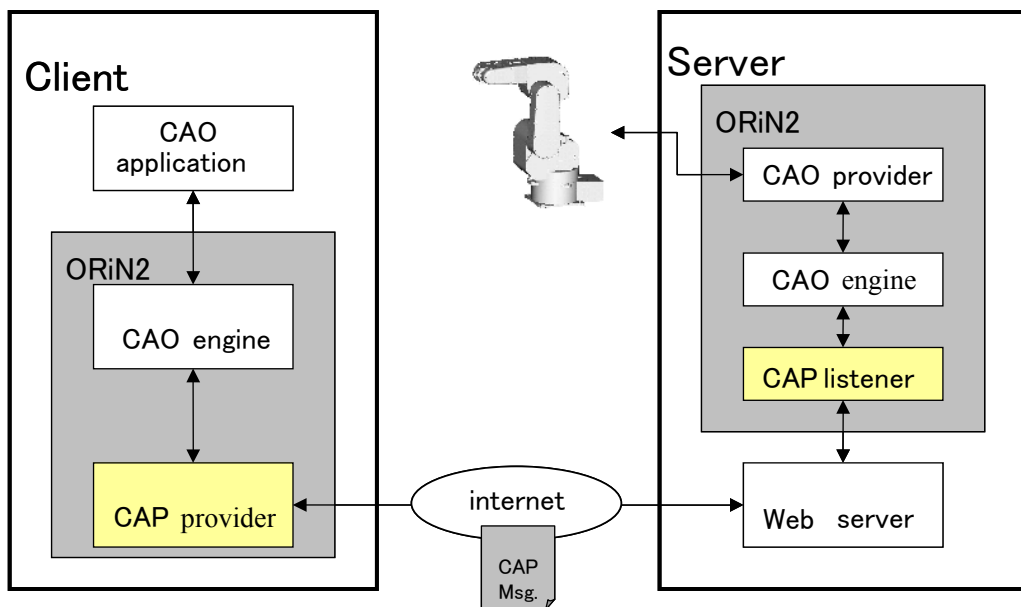


Figure4-2 Outline of CAP

## 4.4. Environmental construction

This chapter explains the environmental construction to use CAP method.

The side where a remote provider is called is expressed by using the CAP provider and "Client" and the called side are expressed at the following, "Server".

It is assumed the one that both server PC and client PC ORiN2 SDK is installed.

### 4.4.1. Setting of server side

#### (1) Installation of Web server

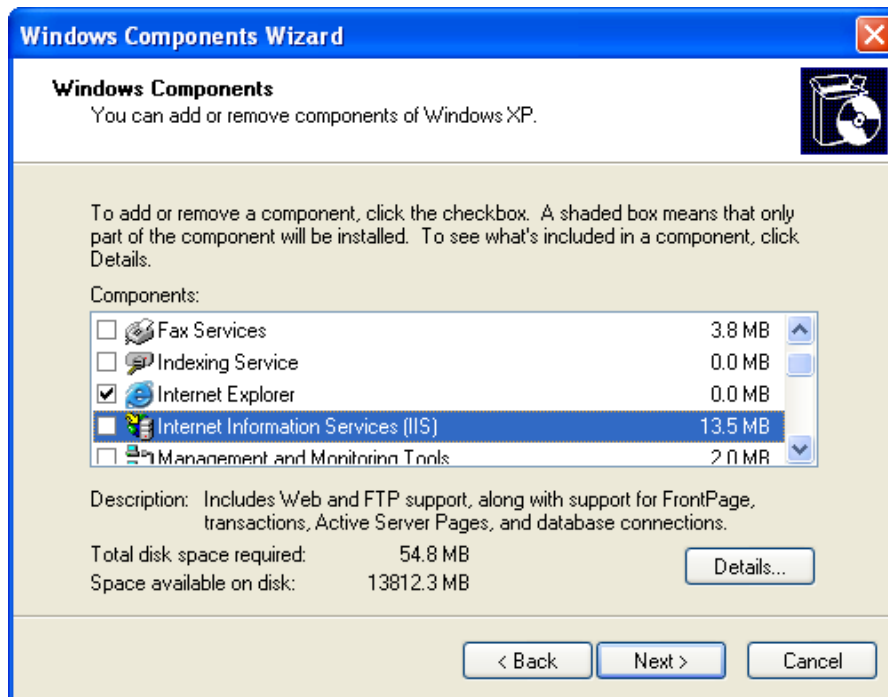
The message transmitted with SOAP as mentioned above starts the CAP listener with the Web server, and analyzes the message. Therefore, it is necessary to construct the Web server first of all.

Web server IIS (Internet Information Service) is appended to Windows 2000 Server/Windows 2000 Professional/Windows XP Professional by the standard<sup>12</sup>. However, it is necessary to do an additional installation because it is not installed in the default installation. (It is installed by the standard for Windows 2000 Server.)

In this paragraph, it explains enumerating it as an example of Windows XP Professional.

"Control panel" → of "Setting" → "Add or Remove Programs" is started from the start menu, and "Add/Remove Windows Components" is selected.

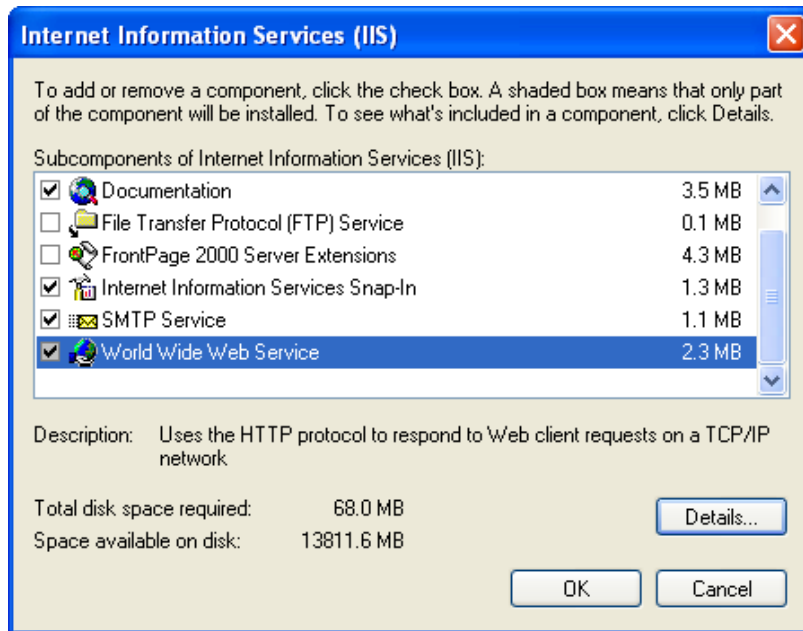
Figure4-3 Because the shown screen is displayed, the check is put in the check box of Internet Information Services (IIS) from among that.



<sup>12</sup> 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**

Figure4-4 Please the shown screen must be displayed, and double-click "WWW (World Wide Web) Service", and put the check in the check box of all components. The check enters automatically for "Internet Information Service Snap-In" and "Common Component".

**Figure4-4 Addition of the Internet information service**

When the installed component is selected, and "Following" button is pressed, the installation of the Windows component is begun.

"Windows XP Professional installation disk" is demanded while installing it.

(2) Installation of SOAP Toolkit

Next, to analyze the message that has been transmitted with SOAP, Microsoft SOAP Toolkit 3.0 is installed. When downloaded soapsdk.exe is double-clicked, and the button is pressed according to the instruction, the installation is completed. The setting is not cared about like default.

**Table 4-1 Soap Toolkit 3.0 download site**

File name	URL
soapsdk.exe	<a href="http://www.microsoft.com/downloads/details.aspx?">http://www.microsoft.com/downloads/details.aspx?</a>

	<a href="#">FamilyID=c943c0dd-ceec-4088-9753-86f052ec8450&amp;DisplayLang=en</a>
--	--

(3) Making of virtual directory

A virtual directory is made by using SOAPVDIR.CMD of Microsoft SOAP Toolkit 3.0. A virtual directory here is the one that the address of the access with HTTP was allocated in a physical passing. SOAPVDIR.CMD is in the following directories.

< SOAP installation directory > \Binaries

The following commands are input to making by the command prompt.

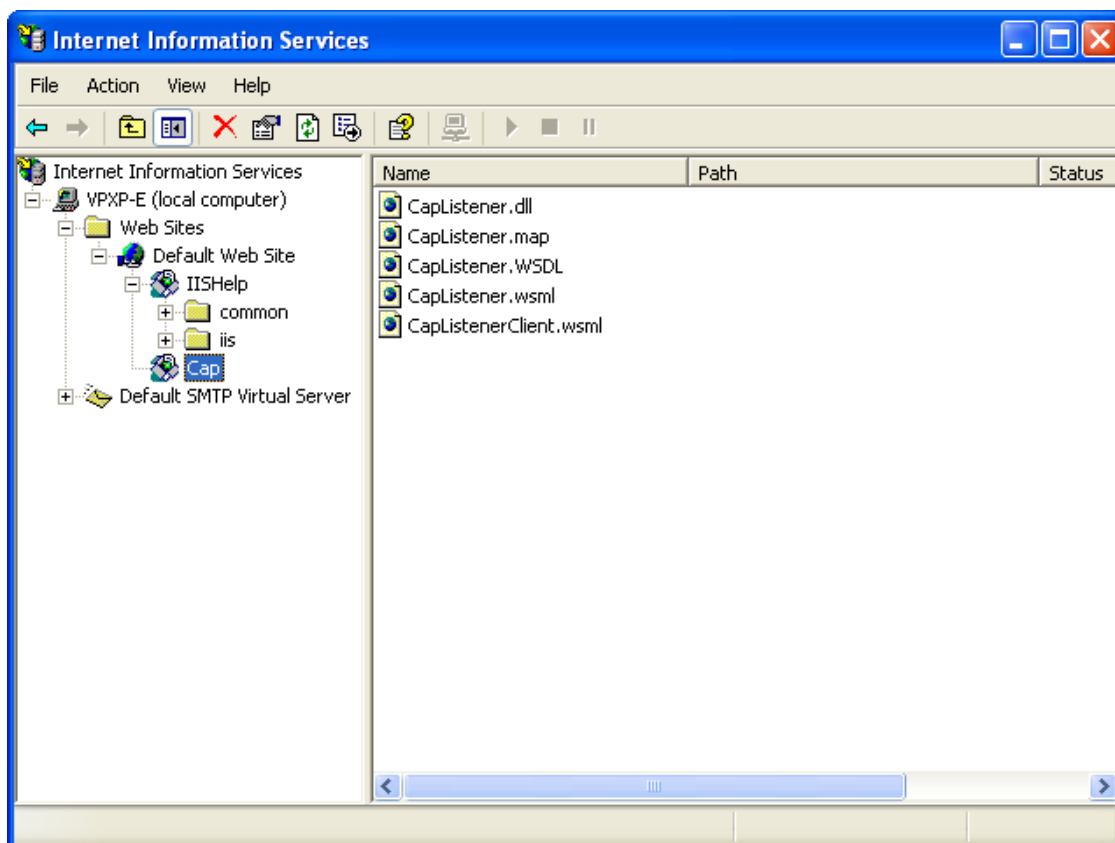
SOAPVDIR.CMD CREATE Cap < Bin directory of CapListener >

(4) Setting of IIS

Next, it is set to start the CAP listener from IIS.

"Control panel" → of "Setting" → "Administrative Tools" is started from the start menu, and "Internet Information Services" is started.

It can be confirmed that virtual directory "Cap" made ahead exists by selecting "Web sites" → of "Local computer" → "Default Web Site" from the Internet Information Services.



**Figure4-5 Set screen of IIS**

Here, please right-click in the following two files in the Cap directory, display properties, and put the check in the check box of "Read".

- CapLister.WSML
- CapListerClient.WSML

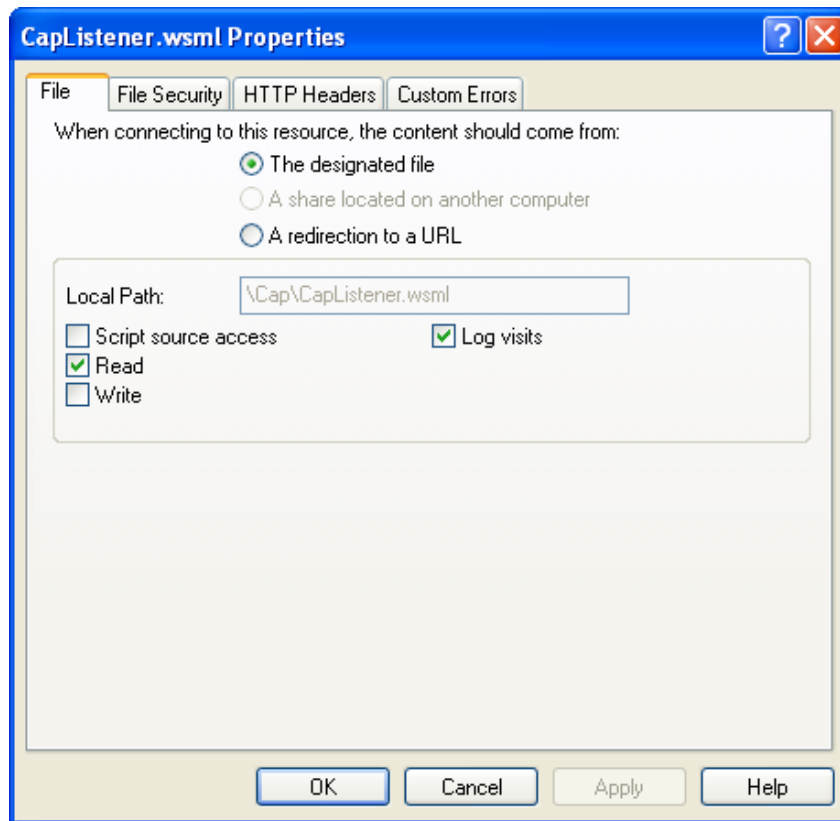


Figure4-6 Set screen of WSML file

(5) Setting of start authority of CAO engine

The following set the authority so that the user who has accessed it by way of the Internet may start the CAO engine.

Please right-click in < ORiN installation directory > ¥CAO¥Engine¥Bin¥CAO.EXE, and select "Security" → of "Properties" → "Add...".

Figure4-7 Because the dialog is displayed, the Internet guest account is added, and the start authority is added.

The Internet guest account name can be confirmed from the start menu by "Control panel" → of "Setting" → "User Accounts".

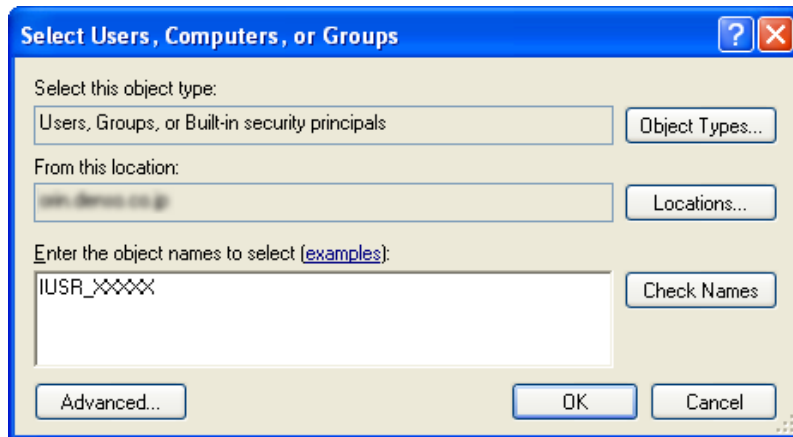


Figure4-7 Addition of the Internet guest account

When succeeding in the addition of the account Figure4-8 The account name is displayed in.

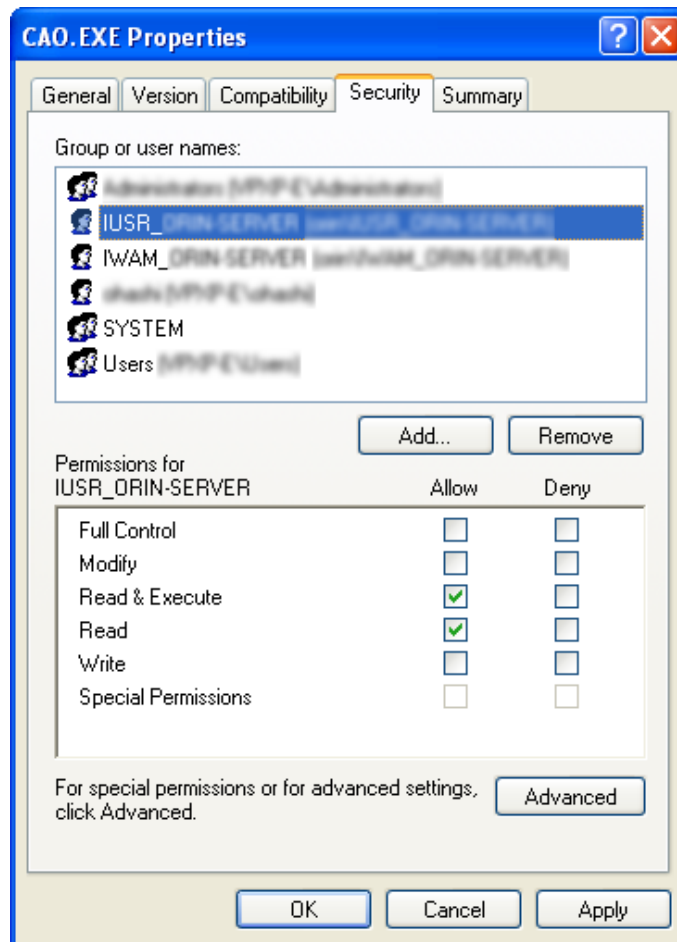


Figure4-8 The Internet guest account addition result



## (6) Setting of CapListner.WSDL file

Only the server name of the 3459th line of the CapListner.WSDL file in the CapListener directory is changed to the name of an actual Web server.

(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.

- It is "Invalidate it. " as for the setting of the Windows firewall
- The setting that exceptionally permits the connection of IIS is done.

When PC has been protected enough by the firewall etc. , the setting of the Windows firewall can be invalidated. When the setting of the firewall is invalidated, the CAP provider can be used even if the setting described as follows is not done. In that case, Figure4-9 Please start "Control panel" →"Windows Firewall" from the start menu as .... shown, and put the check in "Off".

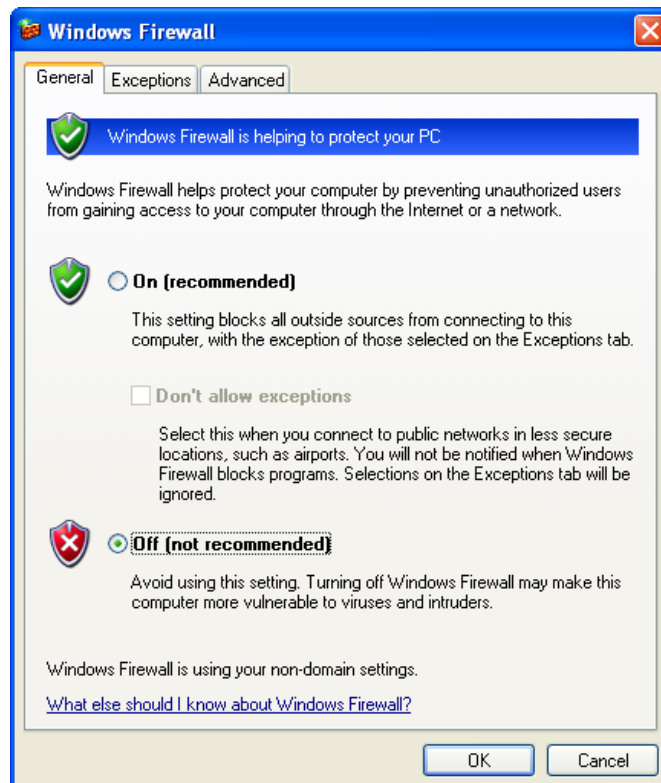
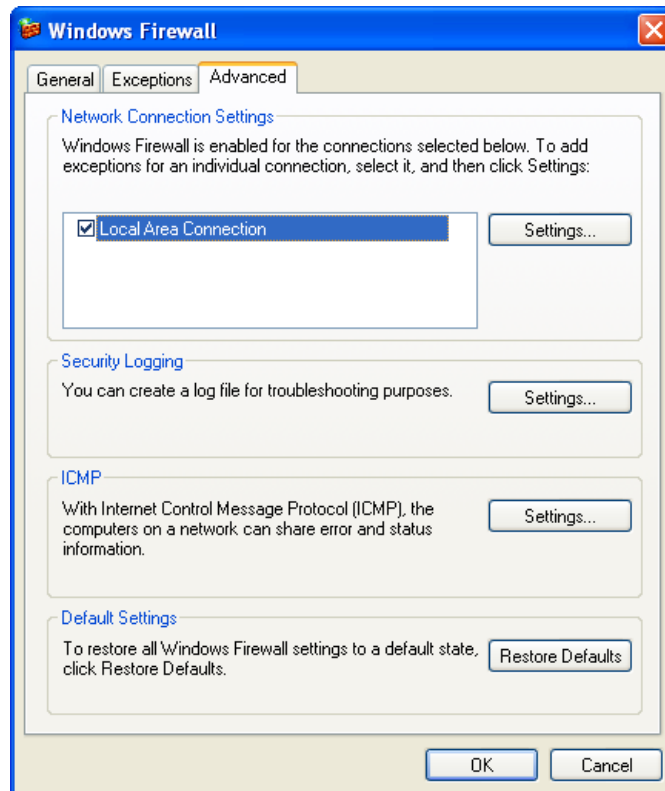


Figure4-9 Nullification of Windows firewall

When the connection of IIS is permitted by the firewall, it sets it as follows.

Please start "Control panel" → "Windows Firewall" from the start menu, and select "Advanced" tab.  
Figure4-10 Please select the connection method corresponding to a present environment, and press "Settings..." button from among "Network Connection Settings" in .... showing. (In this example, "Local Area Connection" was selected. )



**Figure4–10 Exception setting of connection**

Figure4-11 The dialog is displayed, and put the check in "Web Server (HTTP)", please.

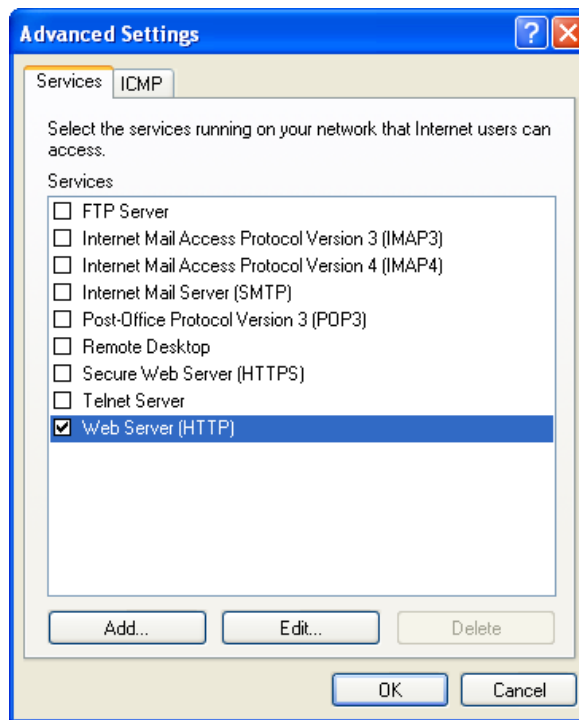


Figure4-11 Selection of service added to exception

#### 4.4.2. Client side

- (1) Installation of SOAP Toolkit

Microsoft SOAP Toolkit 3.0 is installed. (Table 4-1 Reference)

#### 4.5. Sample program

To confirm the operation like CAP, the sample application is made. Please make the form with 2 buttons and one text boxes by starting VB6, and describe 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(
        "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
```

Please look at (1) in the source code. The provider by way of the Internet is connected by this AddController method. The first-third argument of this method is the one similar to 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.

The ctrl object can be treated as a controller target of the DataStore provider at the example of the above-mentioned.

## 5. Environmental setting

### 5.1. Set-up steps DCOM

In ORiN2 SDK, it is necessary to set it with a configuration tool of DCOM to start CAO or the CAO provider remotely. It fails in a remote start or the access by the error's occurring if security is not correctly set to the DCOM client and the server application.

It explains set-up steps of DCOM that uses the configuration tool DCOMCNFG.exe here. Please refer to the following reference for details.

- 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 Deproimentogaid, Oyosha, written by Frank E. Redmond III, top studio translation, and Ryoji Kaneko supervision

The content introduces here is a setting to start CAO or the CAO provider with DCOM remotely, and the security setting is a Lorre bell (state that security doesn't hang) in the example of this book. Please note that there is a possibility that a malicious program is attacked from the outside in this setting enough. Please review the setting according to an individual environment if it is possible.

### 5.1.1. Preparation

- (1) Please prepare the machine that becomes DCOM server and the machine that becomes DCOM client.  
The following OS is targeted here.

**Table 5-1 OS and correspondence of DCOM**

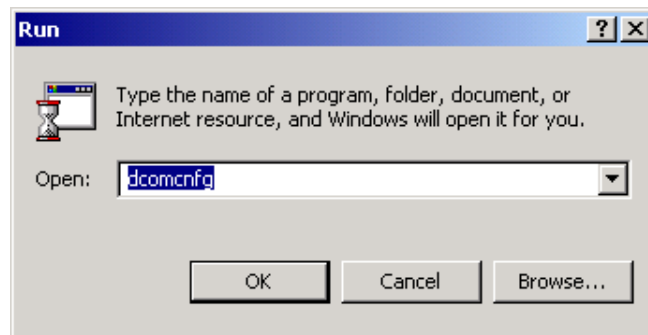
OS	DCOM server	DCOM client
Windows95	√	√
Windows98	√	√
WindowsNT 4.0 WorkStation	√	√
WindowsNT 4.0 Server	√	√
Windows2000 Professional	√	√
Windows2000 Server	√	√
Windows XP Professional	√	√

- (2) Please make the DCOM client and the server participate in the same domain.

### 5.1.2. Set item in WindowsNT/2000

- (1) DCOMCNFG.EXE is started.

It is input from "Run" of the start menu as "dcomcnfg", and starts DCOMCNFG.EXE.



**Figure5-1 Start of DCOMCNFG**

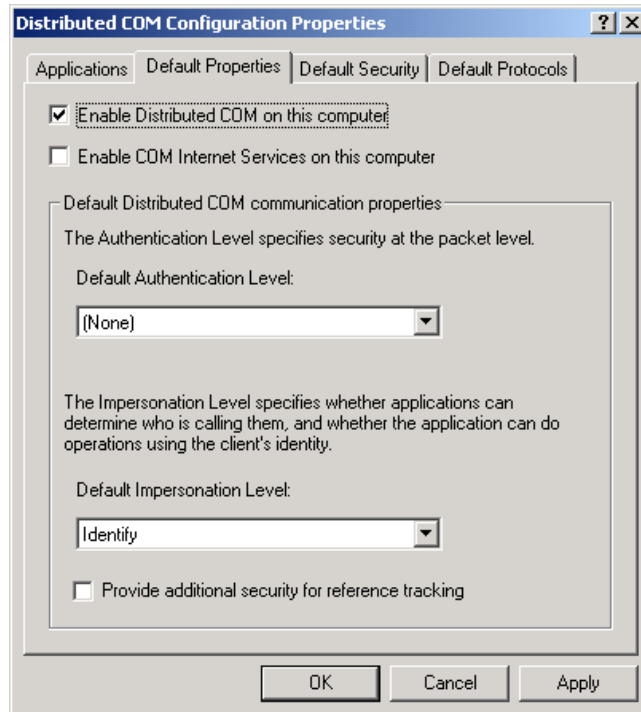
- (2) Setting of default

The value that becomes predetermined of all the DCOM applications installed in local PC can be edited by setting a predetermined item. Therefore, it is necessary to set it carefully. Moreover, please do not change this setting when there is a problem in security. (However, the use by way of DCOM for CAO cannot be likely to be done in that case. )

Please select "Default properties" tab.

Figure5-2 Please put the check in "Enable Distributed COM on this computer" as.... shown. Please

set "Default Authentication Level" of predetermined decentralization COM communication properties in "(None)" and set "Default Impersonation Level" to "Identify".

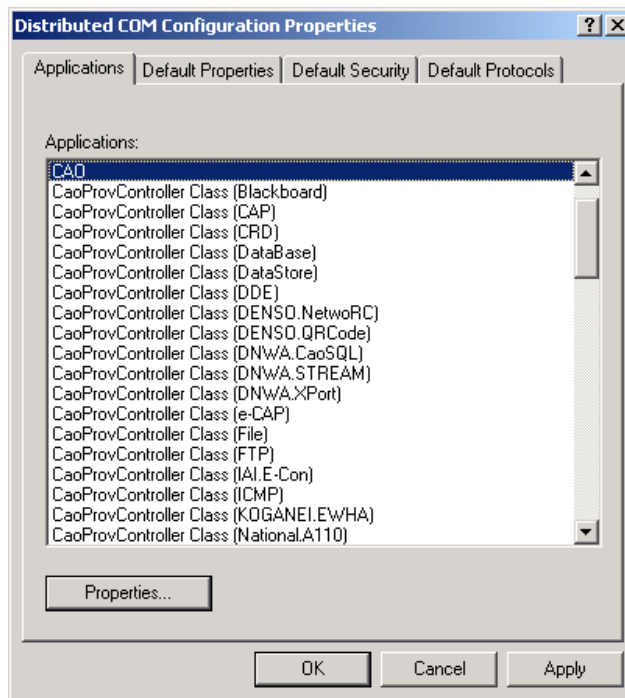


**Figure5-2 Predetermined property**

(3) Setting of each application

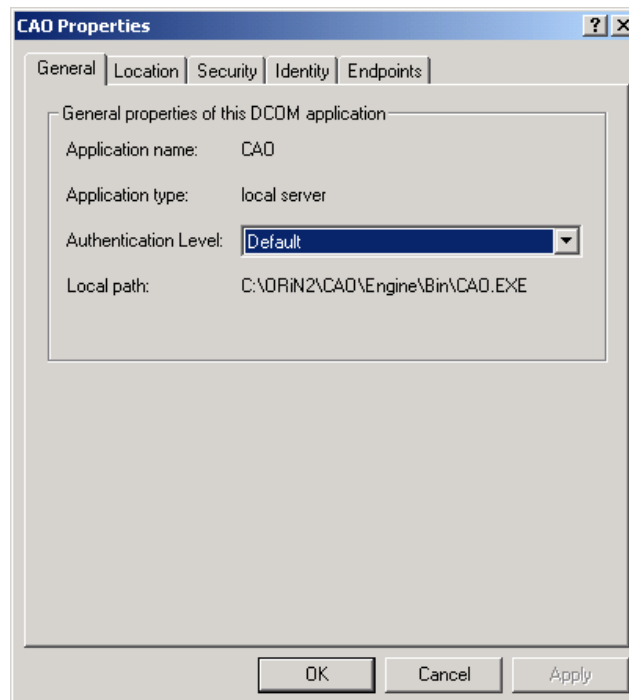
Please select "Applications" tab from the tab of open DCOMCNFG.EXE, activate the application that DCOM wants to start, and press the property button.

It is necessary to do this operation to all the applications that start remotely. In a word, the provider is done and when DCOM starts, this processing is done to all providers that start. Moreover, if remote CAO is used, it is necessary to do similar processing also to CAO.



**Figure5-3 Setting of each application**

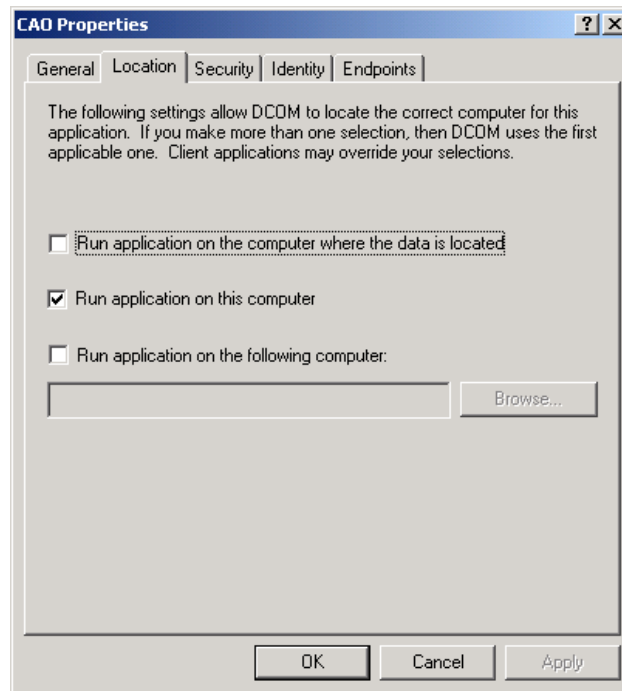
First of all, please select "General" tab. The value set at a predetermined attestation level is used here, and make "Autentication Level" "Default", please.



**Figure5-4 Whole setting**



Next, please select "Location" tab. Application..execute..place..specify. Please select "that executes the application on this computer. Please remove other checks.



**Figure5-5 Place setting**

Please select "Security" tab about the following.

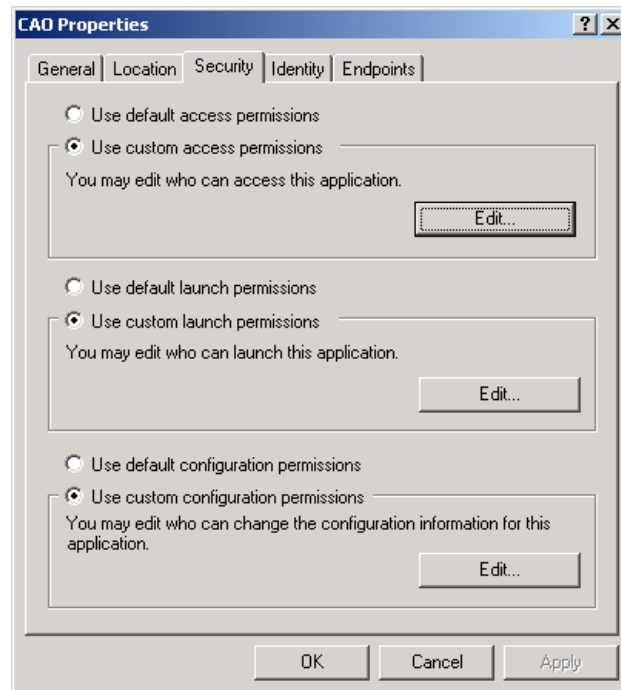


Figure5-6 Security setting

Please select "Use custom access Permissions", press "Edit..." button, and add the following access rights.

- Permission of Everyone-access
- Permission of Interactive-access
- Permission of System-access

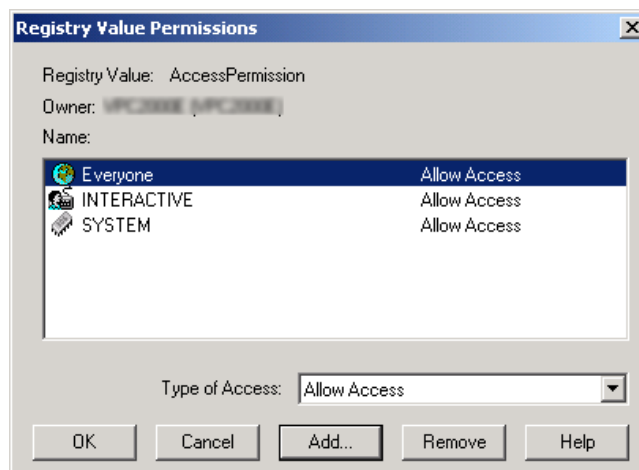


Figure5-7 Addition of access right

Please similarly select " Use custom launch Permissions ", press "Edit..." button, and add the

following launch access rights.

- Permission of Everyone-access
- Permission of Interactive-access
- Permission of System-access

Please select "Identify" tab. And, "The interactive user" is selected. When the DCOM server is started with WindowsNT, it is possible even by "The launching user". Moreover, you may properly input the user by "This user". However, when service is registered, "The launching user" and "The Interactive user" cannot be selected. (Instead, the system account can be used. )

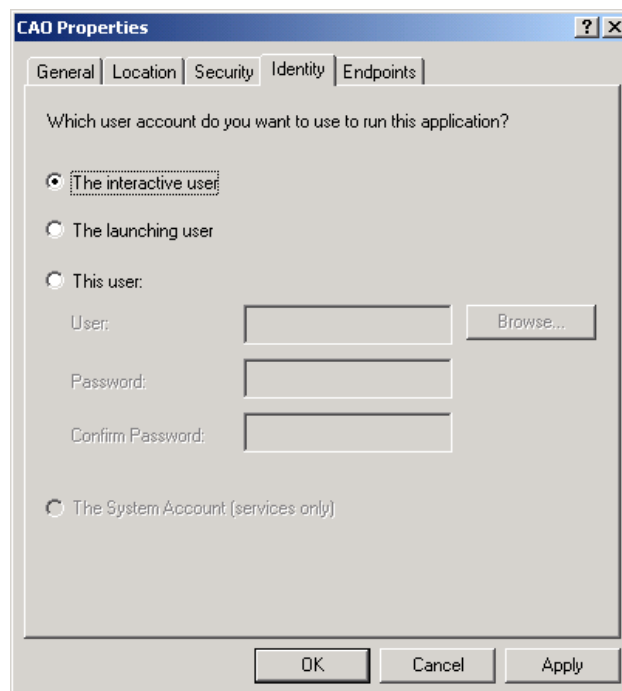


Figure5-8 Identification setting

### 5.1.3. Set item in Windows9x

Please download DCOM98 or 95 and DCOMCNFG98&95 from following URL according to OS, and install it in PC when you use Windows 98 and Windows 95.

< download of file related to Microsoft COM technology >

<http://www.microsoft.com/com/default.msp>

- (1) DCOMCNFG.EXE is started.

It is input from "Run..." of the start menu as "DCOMCNFG", and starts DCOMCNFG.EXE.

- (2) Setting of predetermined properties

Please set it as follows.

- The Authentication Level:None
- The Impersonation Level:Identify.

(3) Setting of predetermined access right

Please add the following access rights.

- It world-permits.

Because the function of a remote start is not supported in DCOM98 of Windows 9x, the program is not started beforehand to make PC of 9x system a server and it doesn't become it. Please use dllhost.exe to start not the CAO engine but provider DLL beforehand, and start as follows by the command prompt.

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

For instance, it executes it as follows for the TCmini provider.

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

< AppID > has been described to the CaoProvController.RGS file of each provider source project. Moreover, the same setting as the above-mentioned can be done by using GUI when the CaoProvExec tool of the ORiN2 SDK attachment is used. Use. Please refer to "[CAO provider development guide](#)".

#### 5.1.4. Set item in Windows XP

(1) Start of DCOMCNFG

It is input from "Run..." of the start menu as "dcomcnfg", and starts DCOMCNFG.EXE.

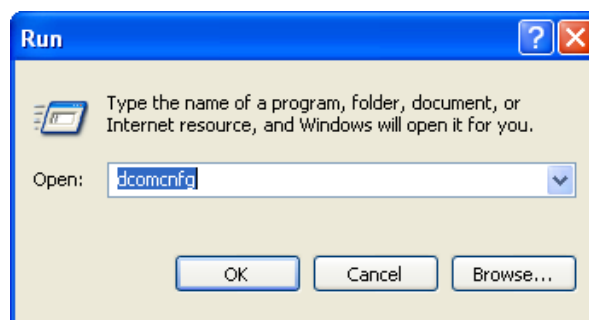


Figure5-9 Method of starting DCOMCNFG

(2) Setting of microcomputer pewter

Figure5-10 Selects "My Computer" → of "Component Services" →"Microcomputer pewter" as

shown, and starts "Properties" from right-clicking.

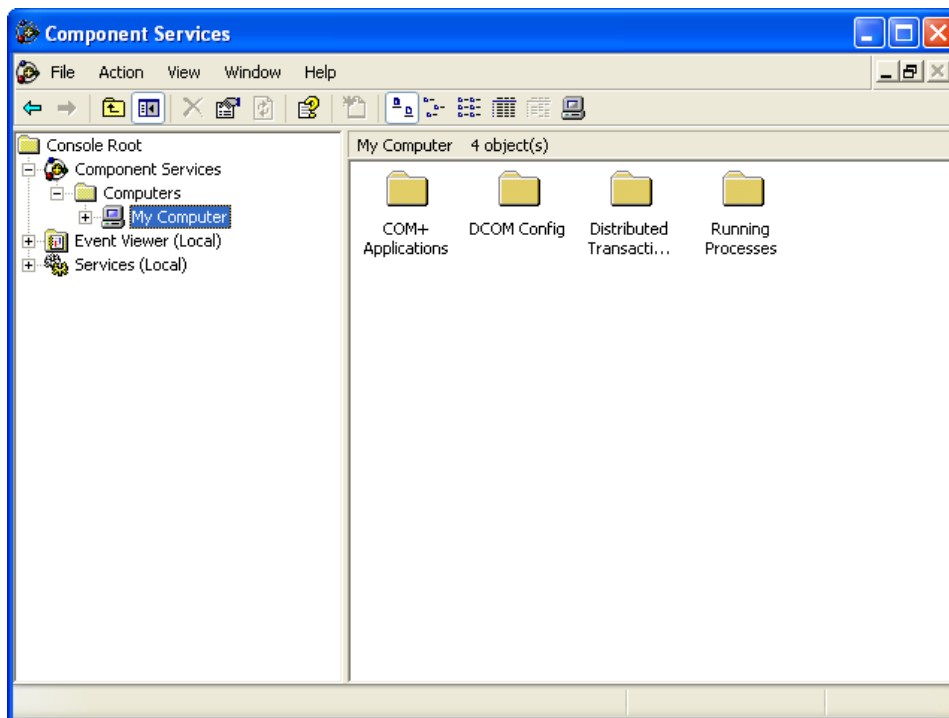


Figure5-10 Set screen of DCOMCNFG

Figure5-11 Selects "Default Properties" tab as shown.

The check is put in "Enable Distributed COM on this computer".

"Default Authentication Level" is set and "None" and "Default Impersonation Level" are set to "Identify".

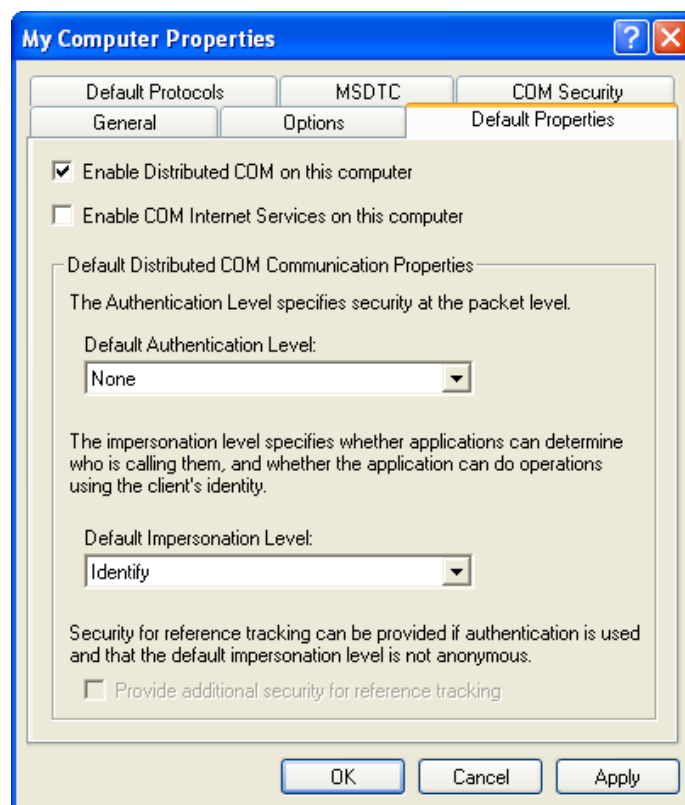


Figure5-11 Property of microcomputer pewter

(3) Setting of each application

"CAO" is selected from among "Microcomputer pewter" → of "My Computer" → of "Component Services" → "DCOM Config", and the property of right-clicking is started.

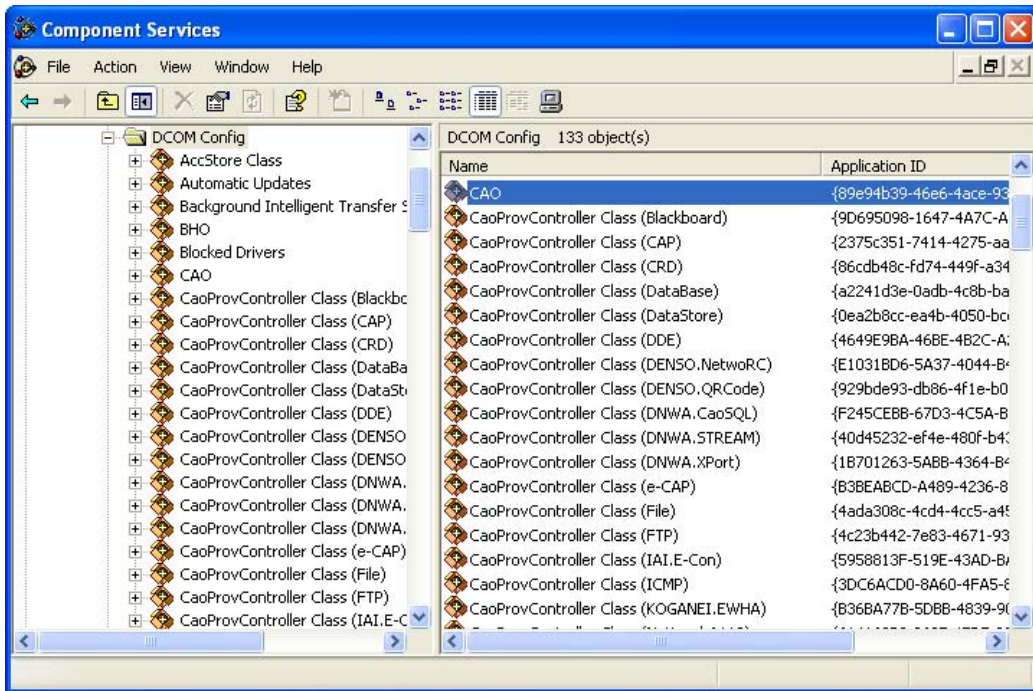
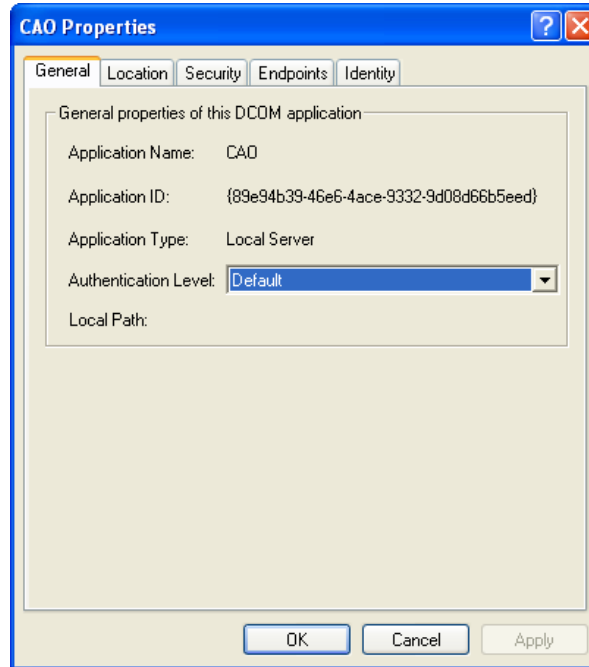


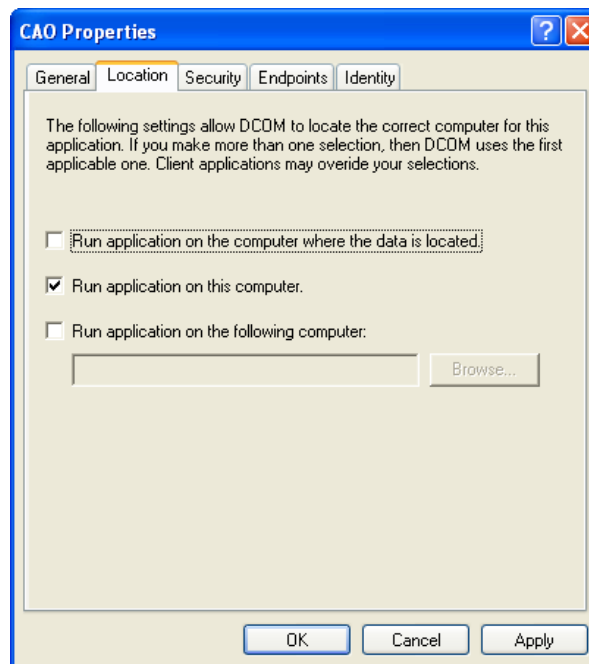
Figure5-12 Composition setting screen of DCOMCNFG

"General" tab is selected, and "Authencation Level" is set to "Default".



**Figure5-13 Set whole of each application**

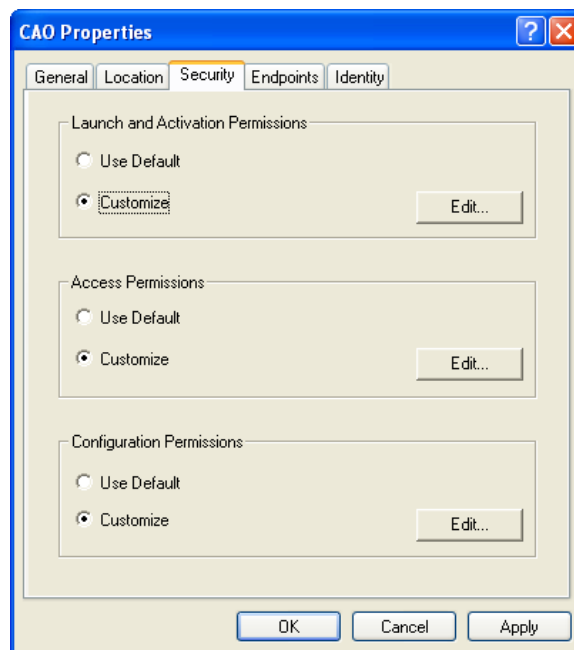
"Location" tab is selected, and the check is put in "Run application on this computer".



**Figure5-14 Set place of each application**

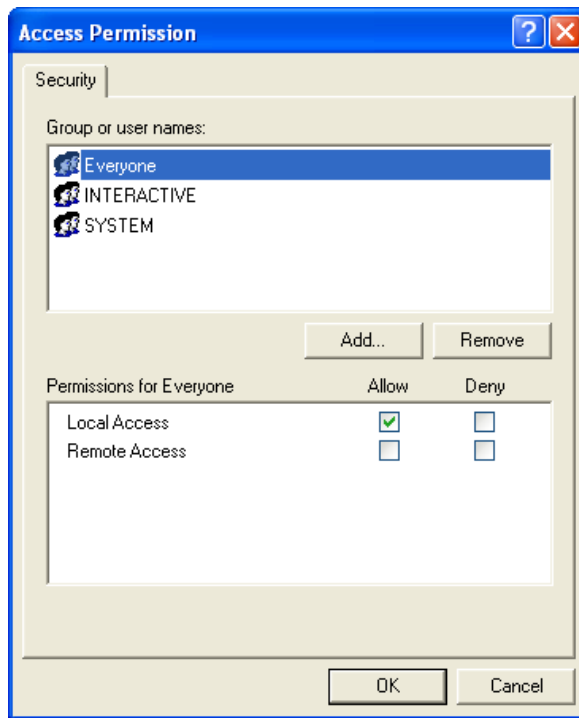


"Security" tab is selected, the check is put in "Customize" of "Launch and Activation Permissions" and "Access Permissions", and "Edit..." button is pressed.



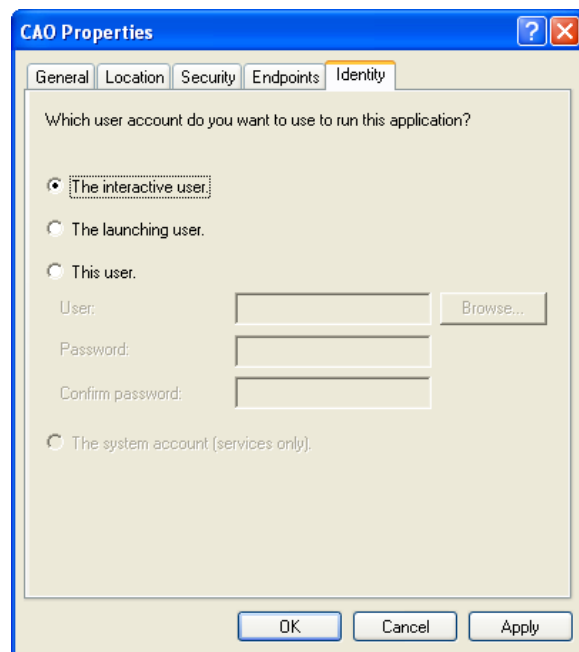
**Figure5–15 Set security of each application**

"Add..." button is pressed, the user named "Everyone", "Interactive", and "SYSTEM" is added, and the check is put in "Access permission".



**Figure5-16 Set security access permission of each application**

"Identify" The tab is selected, and the check is put in "The interactive user".



**Figure5-17 Setting ID of each application**

The CAO provider that uses it similarly sets it.

**5.1.5. Set item in Windows XP SP2**

As for Windows XP Service Pack 2(SP2), the security item such as DCOM is strengthened to establish stronger security, and the function of the firewall is more effective in default.

It cannot access the CAO provider remotely for that.

**5.1.5.1. Setting of Windows firewall**

When PC has been protected enough by the firewall etc. , the setting of the Windows firewall can be invalidated. When the setting of the firewall is invalidated, the CAO provider can be remotely used even if the setting described as follows is not done.

"Windows Firewall" is started from "Setting" → "Control Panel" of the start menu.

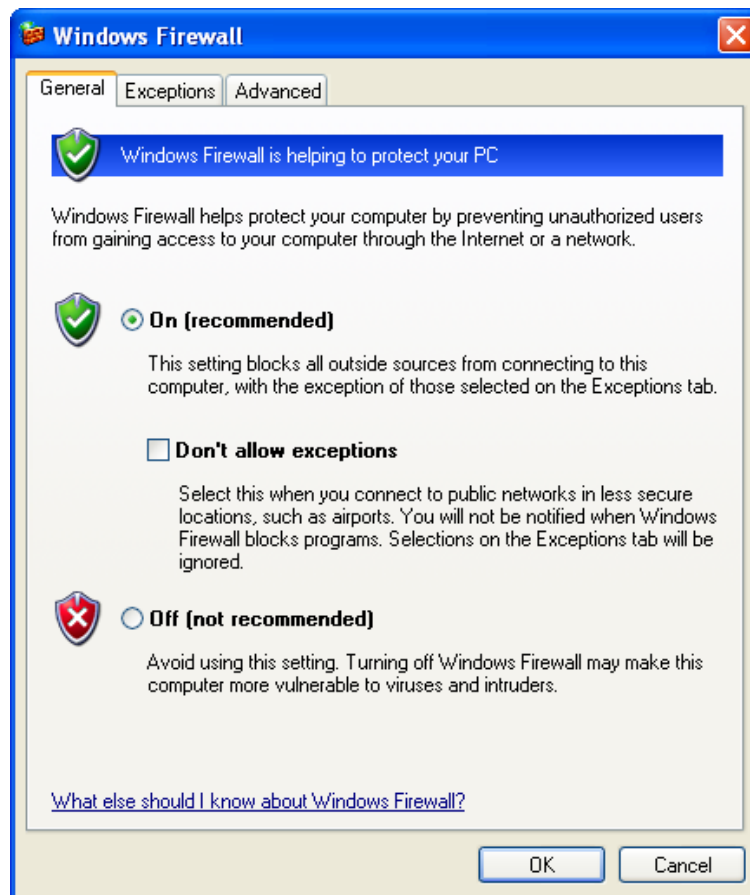


Figure5-18 Windows firewall setting

### 5.1.5.2. Exception setting of firewall

"Windows Firewall" is started from "Setting" → "Control Panel" of the start menu, and "Exceptions" tab is selected.

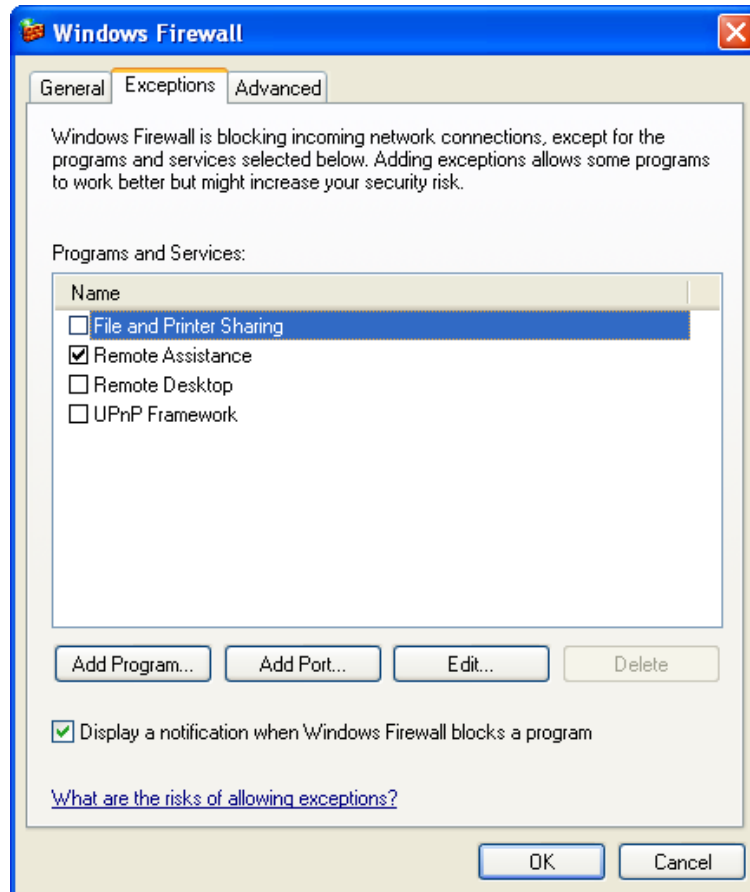


Figure5-19 Addition of exception handling

When you press "Add Program..." button of the exception tab Figure5-20 The shown dialog is displayed, and add the following programs, please. Please press "Browse..." button when the program doesn't exist in the list.

< Windows installation directory > %WINDOVS%system32\dllhost.exe

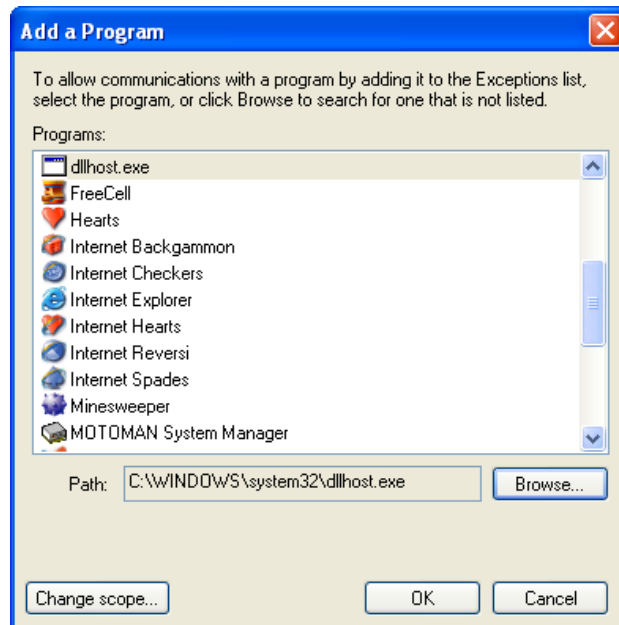


Figure5-20 Addition of exception program

Next, the addition of the port is selected in "Exceptions" tab of "Windows Firewall".

The following ports are added by an additional dialog of the port.

- Name: DCOM
- Port number: 135

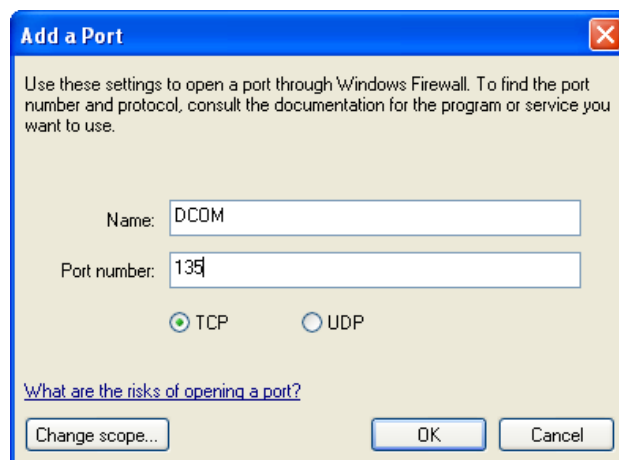


Figure5-21 Addition of exception port number

Figure5-22 It is confirmed that there are checks in "DCOM" and "dllhost.exe" as showing, and presses the "OK" button. The security setting of the firewall is completed above.

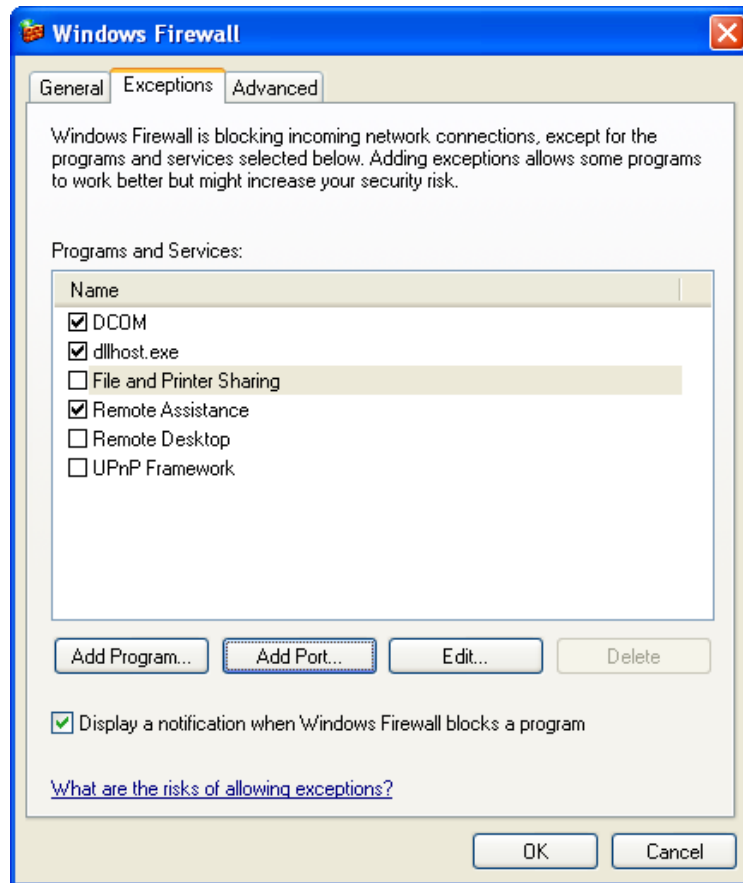


Figure5-22 Additional result of exception

### 5.1.5.3. Setting of DCOM

5.1.4Set item in Windows XP Similar is set.

### 5.1.6. Set confirmation of DCOM

The provider connection test remotely is done by using test tool "CaoTester" of the attachment for ORiN2 SDK to confirm the setting of DCOM.

Please refer to the [CaoTester user's guide](#) for detailed use of CaoTester.

This time, "DataStore provider" is used as a provider that remotely connects it. Please do above-mentioned "Setting of each application" to the DataStore provider.

CaoTester is started with PC on the client side. Figure5-23 Please input the following items to the workspace child window as .... shown.

- Controller Name : Please input an arbitrary character string.
- Provider Name : Please select CaoProv.DataStore this time.
- Machine Name : Please input the host name of the DCOM server.

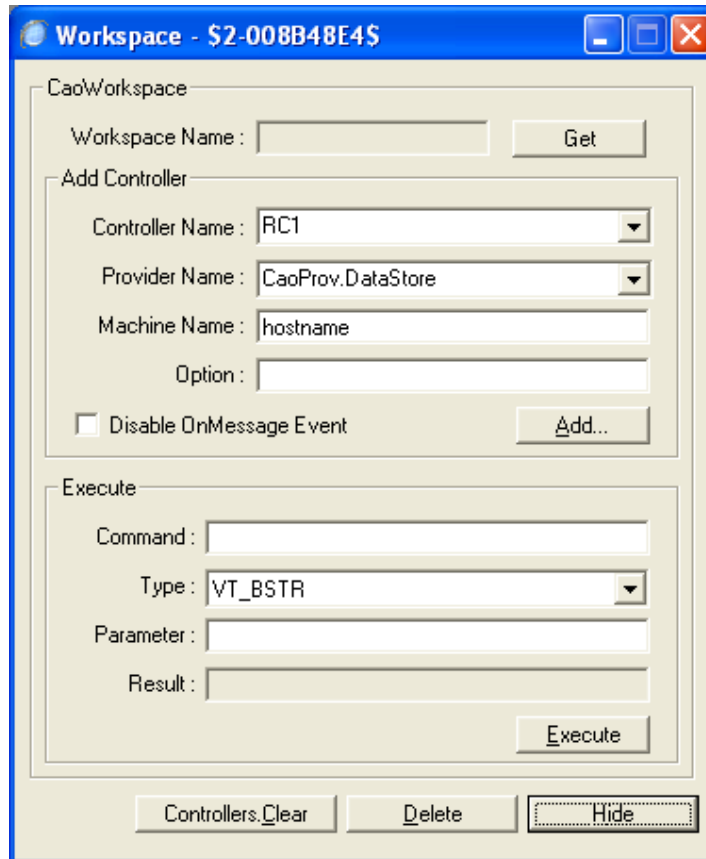
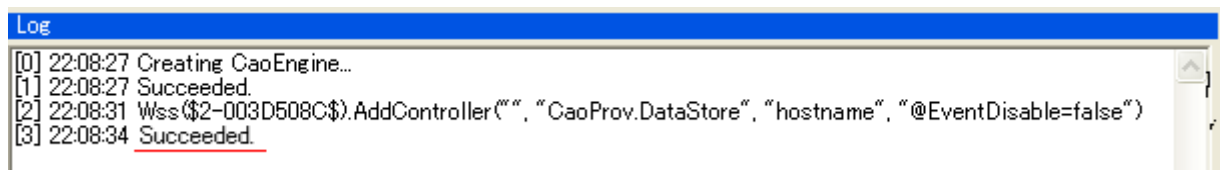


Figure5-23 Remote provider connection by DCOM

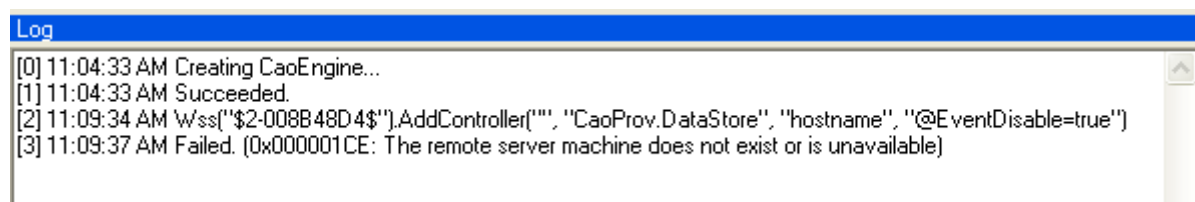


Please press the “Add” button of the workspace child window. To the display of the log under the left of CaoTester window when failing in a remote connected (Figure5-25). When succeeding in the connected (Figure 5-24).



```
Log
[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)**



```
Log
[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)**

## 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.

CaoConfig set result is recorded in registry.

### 6.1. Screen composition

#### 6.1.1. Menu

##### 6.1.1.1. File Menu

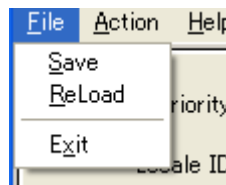


Figure 6–1 CaoConfig File Menu

##### [Preservation] – Save

Information on CaoConfig is set to the registry.

##### [Rereading seeing] – Reload

Information on CaoConfig is acquired from the registry.

##### [Import] – Import

Import CaoConfig settings from an XML file.

##### [Export] – Export

Export CaoConfig settings to an XML file.

##### [End] – Exit

CaoConfig is ended.

##### 6.1.1.2. Action Menu



Figure 6–2 CaoConfig Action Menu

**[Start service] – Service Start**

The CAO service begins. Only when CAO was registered as service, it is possible to use it.

**[Stop service] – Service Stop**

CAO stops serving. Only when CAO was registered as service, it is possible to use it.

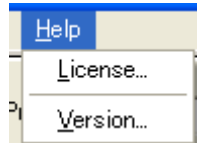
**6.1.1.3. Help Menu**

Figure 6–3 CaoConfig Help Menu

**[License] – License**

The screen of the management of the license of CAO is displayed.

**[Version] – Version**

Version information on CaoConfig is displayed.

**6.1.2. Cao Engine tab**

On CaoEngine tab, CAO Engine operation can be set.

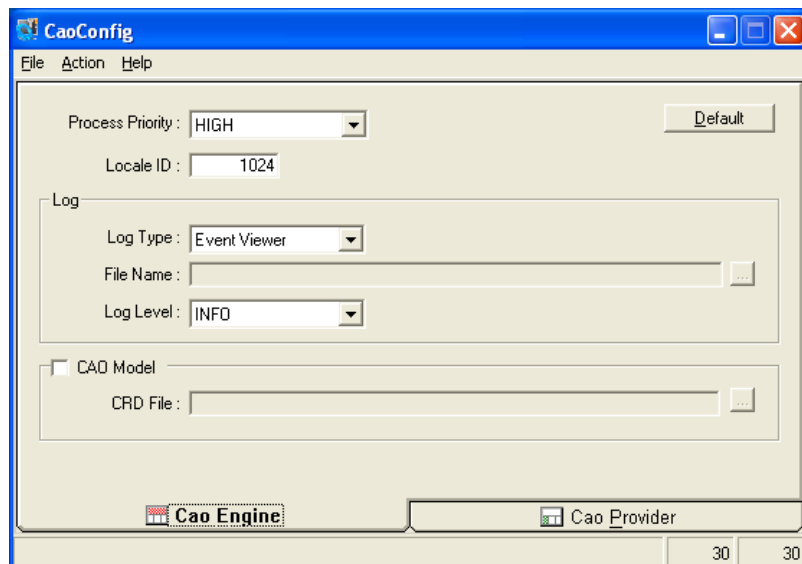


Figure6–4 Cao Engine tab

**[Process priority]**

Set CAO engine process priority. The priority can be set as following.

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

**[Local ID]**

Set language ID to be used.

**[Log type]**

Select CAO.exe log output type. Followign 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.

**[Log output file name]**

When the log type is Text File, the file path is specified here.

**[Log level]**

Set log output level. (The log more than the set level is output.) Log..level..setting..as follows..level..select.It is the highest the seriousness degree, and the more approaches "DEBUG" by FATAL "the more the seriousness degree lowers. It is set to "INFO" in the standard.

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

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

The presence of the object automatic operation registration function of the engine is set.

About details of the object automatic operation registration function 2.7.2Please refer to.

**[CRD file] – CRD File**

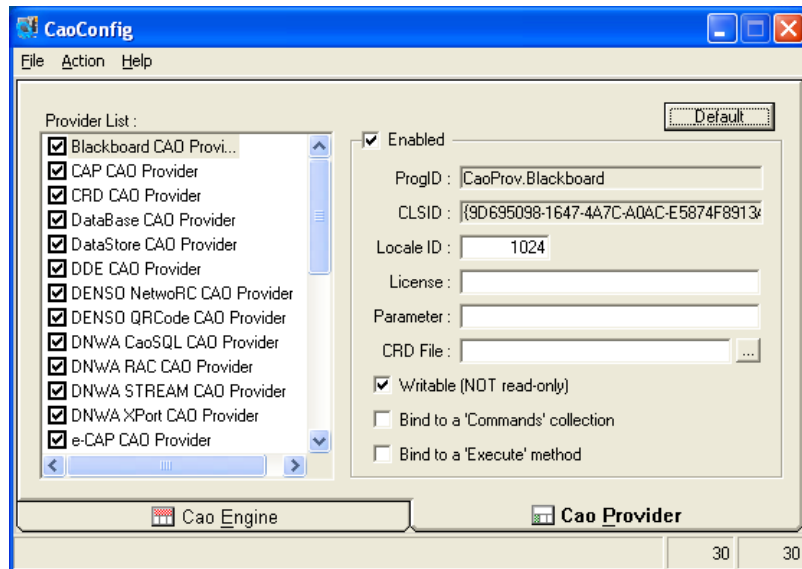
The path of the CRD file used by the object automatic operation registration of the engine is specified.

**[Default] – Default**

The setting of the engine is returned to default.

**6.1.2.1. Cao Provider tab**

The operation setting of each provider can be done in the Cao Provider tab.



**Figure6-5 Cao Provider tab**

It explains the item that can be set as follows.

**[Provider list] – Provider List**

The list of the installed provider is displayed. Enable can be switched with the check box.

**[Enable/Disable setting] – Enable**

Available/impropriety of the provider is set.

**[Program ID] – ProgID**

Program ID of the provider is displayed.

**[Class ID] – CLSID**

Class ID of the provider is displayed.

**[Local ID] – Locale ID**

Locale ID of the provider is set.

**[License] – Lisence**

The license of the provider is set.

**[Parameter] – Parameter**

The parameter of the provider is set.

**[CRD file] – CRD File**

The path to the CRD file used at the CRD file switch function of each provider is set. About details 2.7.1Please refer to.

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

Read-only of the provider is set.

**[Commands collection method dynamic and addition functions] – Bind to a ‘Commands’ collection**

The presence of the dynamic and addition of method of CaoCommands functions is set. About detail 2.7.3Please refer to.

**[Execute method dynamic and addition functions] – Bind to a ‘Execute’ method**

The presence of the dynamic and addition of method of CaoController::Execute function function is set. About detail 2.7.3 Please refer to.

**[Default] – Default**

The setting of the provider is returned to default.

## Appendix A. Appendix

### Appendix A.1. CAO engine function list

#### ◆ CaoEngine Object-engine

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
CaoEngine (class number: 0)	EngineStatus P	Acquisition of engine status object	R		Object: IcaoEngineStatus	
	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 enhancing(reservation)
Value of a symbol	M:Method P:Property E:Event		(note 1)	<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

(note 1)The value of a symbol is as follows. Only the method and the property of W attribute : influenced from ON/OFF of the access limitation function of the CAO engine (writing limitation function) in this. R-Read: Status and the configuration of the controller or the provider or the engine are acquired. W-Write: Controller's status and configuration are changed. However, because the Execute method depends on the content of the command, it is assumed S attribute. The writing limitation is mounted in the provider if there is a necessity. S-Setup: The provider or the status of the engine and the configuration are changed.

## ◆ CaoWorkspace(s) Object-workspace

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks	
				IN	OUT RETVAL		
CaoWorkspaces (class number: 1)	Count	P	Acquisition of number of collections	R		Number of collections: Long	
	Add	M	Addition of workspace object	S	Workspace name:BSTR, [Option:BSTR]	Object: ICaoWorkspace	The workspace name is an arbitrary character string. When the workspace name is omitted (NULL specification), a unique name is automatically named.
	Clear	M	Liberating 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 (class number: 2)	Controllers	P	Acquisition of controller collection	R		Collection: ICaoControllers	
	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)	The option is a filter condition etc.
	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 enhancing(reservation)



Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
Value of a symbol	M:Method P:Property E:Event			<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BS TR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

◆ CaoController(s) Object-controller

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
CaoControllers (class number: 3)	Count	P	Acquisition of number of collections	R		Number of collections: Long
	Add	M	Addition of controller object	S	Controller-name:BSTR, Provider name:BSTR, [Machine name:BSTR], [Option:BSTR]	Object: ICaoController Controller-name is an arbitrary character string. Time-out value and event permission, etc. of option. Event permission: @EventEnable[=True/False] Default value is "True"
	Clear	M	Liberating 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 (class number: 4)	Attribute	P	Acquisition of attribute	R		Attribute: Long
	CommandNames	P	Acquisition of enhancing board name list	R	[Option:BSTR]	Command name list: VARIANT(VT_VARIANT VT_ARRAY) The option is a filter condition etc.
	Commands	P	Acquisition of command collection	R		Collection: ICaoCommands
	ExtensionNames	P	Acquisition of command name list	R	[Option:BSTR]	Enhancing board name list: VARIANT(VT_VARIANT VT_ARRAY) The option is a filter condition etc.
	Extensions	P	Acquisition of enhancing board collection	R		Collection: ICaoExtensions
	FileNames	P	Acquisition of file name list	R	[Option:BSTR]	File name list: VARIANT(VT_VARIANT VT_ARRAY) The option is a filter condition etc.
	Files	P	Acquisition of file collection	R		Collection: ICaoFiles File collection of root directory.

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
		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)	The option is a filter condition etc.
	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)	The option is a filter condition etc.
	Tasks P	Acquisition of task collection	R		Collection: ICaoTasks	
	VariableNames P	Acquisition of variable identifier list	R	[Option:BSTR]	Variable identifier list: VARIANT(VT_VARIANT VT_ARRAY)	The option is a filter condition etc.
	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 enhancing 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		Remarks
				IN	OUT RETVAL	
	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 identifier:BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Execute M	Execution of enhancing command	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function enhancing
	GetMessage M	Acquisition of message	R		Message: ICaoMessage	The message is acquired from the message queue in the engine. Only when the OnMessage event function is invalid, it is possible to acquire it.
	OnMessage E	Message reception event	R	Message:ICaoMessage		·The call of provider (controller)→ engine → client is achieved. - A negative range of the message number has been reserved with ORiN. - The nullification of the event is specified by the option of AddController().
Value of a symbol	M:Method P:Property E:Event			<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

◆ CaoExtension(s) Object-enhancing 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: Long	
	Add	M	Addition of enhancing board object	S	Expansion board name: BSTR, [Option: BSTR]	Object: ICaoExtension	
	Clear	M	Liberating of all enhancing board objects	S			
	IsMember	M	Registration confirmation of enhancing board object	R	Expansion board name/Number: VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of enhancing board object	R	Expansion board name/Number: VARIANT	Object: ICaoExtension	Default member
	Remove	M	Opening of enhancing board object	S	Expansion board name/Number: VARIANT		
CaoExtension (class number: 6)	Attribute	P	Acquisition of attribute	R		Attribute: Long	
	Help	P	Help	R		Help character string: BSTR	
	ID	P	ID	R/W	ID: VARIANT	ID: VARIANT	
	Index	P	Acquisition of enhancing board number	R		Enhancing board number: Long	
	Name	P	Acquisition of enhancing board name	R		Enhancing board name: BSTR	
	Tag	P	Tag	R/S	Tag data: VARIANT	Tag data: VARIANT	
	VariableNames	P	Acquisition of variable identifier list	R	[Option: BSTR]	Variable identifier list: VARIANT(VT_VARIANT VT_ARRAY)	The option is a filter condition etc.
	Variables	P	Acquisition of variable collection	R		Collection: ICaoVariables	
	AddVariable	M	Addition of variable object	S	Variable identifier: BSTR, [Option: BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Execute	M	Execution of enhancing command	S	Command: BSTR [Parameter: VARIANT]	Result: VARIANT	For function enhancing
Value of a symbol	M: Method					The argument that can be omitted.	

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
ol	P:Property E:Event					<ul style="list-style-type: none"> <li>·The default value of the argument to which the BS TR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>

## ◆ CaoFile(s) Object-file

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
CaoFiles (class number: 7)	Count	P	Acquisition of number of collections	R		Number of collections: Long
	Add	M	Addition of file object	S	File name:BSTR, [Option:BSTR]	Object: ICaoFile  File making: @Create = numerical value default value is 0 0: Excluding 0 not made: It writes and it becomes an error in the place where the provider that makes it is read-only, except when the @Create option is 0.
	Clear	M	Liberating 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 (class number: 8)	Attribute	P	Acquisition of attribute	R		Attribute: Long
	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)  The option is a filter condition etc. 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 passing	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 identifier list	R	[Option:BSTR]	Variable identifier list: VARIANT(VT_VARIANT VT_ARRAY)	The option is a filter condition etc.
	Variables P	Acquisition of variable collection	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 identifier:BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Copy M	Copy	W	Copy destination file name:BSTR [Option:BSTR]		
	Delete M	Deletion	W	[Option:BSTR]		
	Execute M	Execution of enhancing command	S	Command:BSTR [Parameter:VARIANT]	Result: VARIANT	For function enhancing
	Move M	Movement	W	Moving destination file name:BSTR [Option:BSTR]		
	Run M	Task creation	W	[Option:BSTR]	Task name: BSTR	
Value of a symbol	M:Method P:Property E:Event			<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		



## ◆ CaoRobot(s) Object-robot

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks	
				IN	OUT RETVAL		
CaoRobots (class number: 9)	Count	P	Acquisition of number of collections	R		Number of collections: Long	
	Add	M	Addition of robot object	S	Robot name:BSTR, [Option:BSTR]	Object: ICaoRobot	
	Clear	M	Liberating 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 (class number: 10)	Attribute	P	Acquisition of attribute	R		Attribute: Long	
	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 identifier list	R	[Option:BSTR]	Variable identifier list: VARIANT(VT_VARIANT VT_ARRAY)	The option is a filter condition etc.
	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]		The axis number is □1:Acceleration of minion(Deceleration of minion), O:Acceleration of all axis (Deceleration of all axis), etc:Acceleration of specified axis (Deceleration of specified axis).	

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
	AddVariable M	Addition of variable object	S	Variable identifier:BSTR, [Option:BSTR]	Object: ICaoVariable	The same function as CaoVariables::Add().
	Cancel M	Cancellation of method of controlling robot under execution	W	Hand name:BSTR		
	Change M	Refer to the specification of the CHANGE sentence of SLIM.	W	[Option:BSTR]		
	Chuck M	Refer to the specification of the GRASP sentence of SLIM.	W	Axis Number:long, Quantity of movement:float, [Operation option:BSTR]		
	Drive M	Refer to the specification of the DRIVE sentence of SLIM.	W	Command:BSTR [Parameter:VARIANT]		The actuate is not good at the same time at two or more shafts like SLIM. In that case, MOVE is recommended to be used.
	Execute M	Execution of enhancing command	S		Result: VARIANT	For function enhancing
	GoHome M	Refer to the specification of the GOHOME sentence of SLIM.	W	[Option:BSTR]		
	Move M	Refer to the specification of the MOVE sentence of SLIM.	W	[Option:BSTR]		·The interpolation specification is 1: PTP and 2: the straight line and 3: The circular arc. - Each maker dependence for specification of pose. It becomes an array at a circular arc interpolation etc.- Operation option De fault value is ""
	Rotate M	Refer to the specification of the ROTATE sentence of SLIM.	W	Interpolation specification:long, Posed row:VARIANT, [Operation option:BSTR]		Each maker dependence for specification on rotation side.
	Speed M	Refer to the specification of the SPEED/JSPEED sentence of SLIM.	W	Surface of revolution:VARIANT, Angle:float, Rotation center:VARIANT		The axis number is □1:Speed of minion, O:All axis speeds, etc: Specified axis speed.

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
				T, [Operation option:BSTRT]		
	Unchuck M	Refer to the specification of the RELEASE sentence of SLIM.	W	Axis Number:long, Speed:float		Release of SLIM changes to Chuck/Unchuck because it cannot use it by the reserved word.
Value of a symbol	M:Method P:Property E:Event			<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTRT type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

## ◆ 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	Liberating 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 identifier list	R	[Option:BSTR]	Variable identifier list: VARIANT(VT_VARIANT VT_ARRAY)	The option is a filter condition etc.
	Variables	P	Acquisition of variable collection	R		Collection: ICaoVariables	
	AddVariable	M	Addition of variable object	S	Variable identifier: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 enhancing	

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]		The mode is 1:1-cycle execution, and 2: It continuous executes, it sends by 3:1 steps, and 4:1 step return option is a starting position, etc.
	Stop M	Stop of task	W	Mode:long, [Option:BSTR]		The mode is 0: the default stop and 1: the stop at the moment and 2: the step stop and 3: The stop at the cycle. 4: Initialization stop (note) default stop is actually a how to stop of either.
Value of a symbol	M:Method P:Property E:Event			<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

◆ 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 identifier:BSTR, [Option:BSTR]	Object: ICaoVariable	
	Clear	M	Liberating of all variable objects	S			
	IsMember	M	Registration confirmation of variable object	R	Variable identifier/Number: VARIANT	Check result: VARIANT_BOOL	
	Item	M	Acquisition of variable object	R	Variable identifier/Number: VARIANT	Object: ICaoVariable	Default member
	Remove	M	Opening of variable object	S	Variable identifier/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 (micro second)	R		Time and date stamp: Long	
	Name	P	Acquisition of variable identifier	R		Variable identifier: BSTR	The system variable identifier 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				·The argument that can be omitted. ·The default value of the argument to which the BSTR type can be omitted is a dead letter character. ·The default value of the argument to which a numeric type can be omitted is 0.		

◆ CaoCommand(s) Object-command

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks	
				IN	OUT RETVAL		
CaoCommands (class number: 13)	Count	P	Acquisition of number of collections	R		Number of collections: Long	
	Add	M	Addition of command object	S	Command name:BSTR, [Option:BSTR]	Object: ICaoCommand	
	Clear	M	Liberating 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 (class number: 14)	Attribute	P	Acquisition of attribute	R		Attribute: Long	
	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 is 0: standing. ,1: executing it.. etc : Provider dependence.
	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 command when being executing it				

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
	Execute M	Execution of command		Mode:long		The mode is 0: synchronous execution and 1: Asynchronous execution. When asynchronously executing it, S_FALSE is returned.
Value of a symbol	M:Method P:Property E:Event			<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BS TR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		



## ◆ CaoMessage Object-message

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
CaoMessage (class number: 15)	DateTime P	At the date	R		At the date: VARIANT.	
	Description P	Explanation	R		Explanation: BSTR	
	Destination P	Mailing address	R		Mailing address: BSTR	
	Number P	Message number	R		Message number: Long	
	SerialNumber P	Message sequential number	R		Message sequential number: Long	Sequential number to 0- LONG_MAX automatically added with engine. 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)	<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

## ◆ CaoEngineStatus Object-engine status

Class	Property, method, and event	Explanation	R/W	Argument of function		Remarks
				IN	OUT RETVAL	
CaoEngineStatus (class number: 16)	CurrentDateTime P	Present date	R		Date now: 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	(reservation 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)	<ul style="list-style-type: none"> <li>·The argument that can be omitted.</li> <li>·The default value of the argument to which the BSTR type can be omitted is a dead letter character.</li> <li>·The default value of the argument to which a numeric type can be omitted is 0.</li> </ul>		

## 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)  
Client..application..offer..robot..controller..operate..interface.CAO API "points at 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)  
The 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)  
It is 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)  
It is SOAP message defined by CAP.
- CRD (Controller Resource Definition)  
It is Dataskema 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)  
It is a resource of the robot expressed by CRD.
- CRD data(CRD Data)

In the resource data of the robot described according to CRD, it is described by XML.

■ CRD Dataskema(CRD Data Schema)

It is a schema that defines the format of the CRD data.

■ CRD file(CRD File)

It is XML file where the CRD data is described.

■ CRD provider(CRD Provider)

It is a provider to access the CRD data.

■ RAC

It is a standard to send and receive the command between the client application and the robot.