OPC UA Multiple providers
OPC UA gateway

Version 1.1.0

User's Guide

July 1, 2024

NOTE:

[Revision History]

Version	Date	Content
1.0.0	2022-07-29	First edition
1.1.0	2024-07-01	Update of internally used library.

[Compatible devices]

Model	Version	Notes

Table of Contents

1. Introduction 4
2. Provider Overview 5
2.1. Overview 5
2.2. Method properties 7
2.2.1. CaoWorkspace::AddController method 7
2.2.2. CaoController::AddVariable method 10
2.2.2.1. Node ID 11
2.2.3. CaoController::property get_VariableNames 13
2.2.4. CaoVariable::property get_DateTime 13
2.2.5. CaoVariable::get_Value Property 13
3. Command Reference 14
3.1. Controller classes 14
3.1.1. CaoController::Execute("MultipleRead") Command 14
4. Error code 15
5. About Certificate Files 16
5.1. Application certificate 16
5.2. Certificate for Certificate Authentication 16
5.3. Preparation of certificates 17
6. Sample program 18
6.1. CaoOPCUA 18
6.1.1. Access path usage 18
6.1.2. Use node ID 21

1. Introduction

This document is a user's guide for CAO providers for connecting to OPC UA (batch transmission/reception) that provides a means of accessing PLCs (Programmable Logic Controller) from CAO to OPC UA(OLE for Process Control Unified Architecture servers (hereinafter simply referred to as OPC UA Multiple providers).

With this provider, CAO-enabled tools can access not only robots, but also PLCs and displays with OPC UA servers as well as robots.

OPC UA Multiple providers discussed in this document provide security features that allow you to connect securely to OPC UA servers and provide access to the info that the connected devices have.

2. Provider Overview

2.1. Overview

OPC UA Multiple providers are genuine ORiN providers that connect to ORiN CAO, as well as OPC UA clients that conform to OPC UA standard (Fig. 2-1). The providers establish connectivity to servers that conform to OPC UA standard, and can similarly access robotics, PLCs, and various devices that conform to OPCUA.

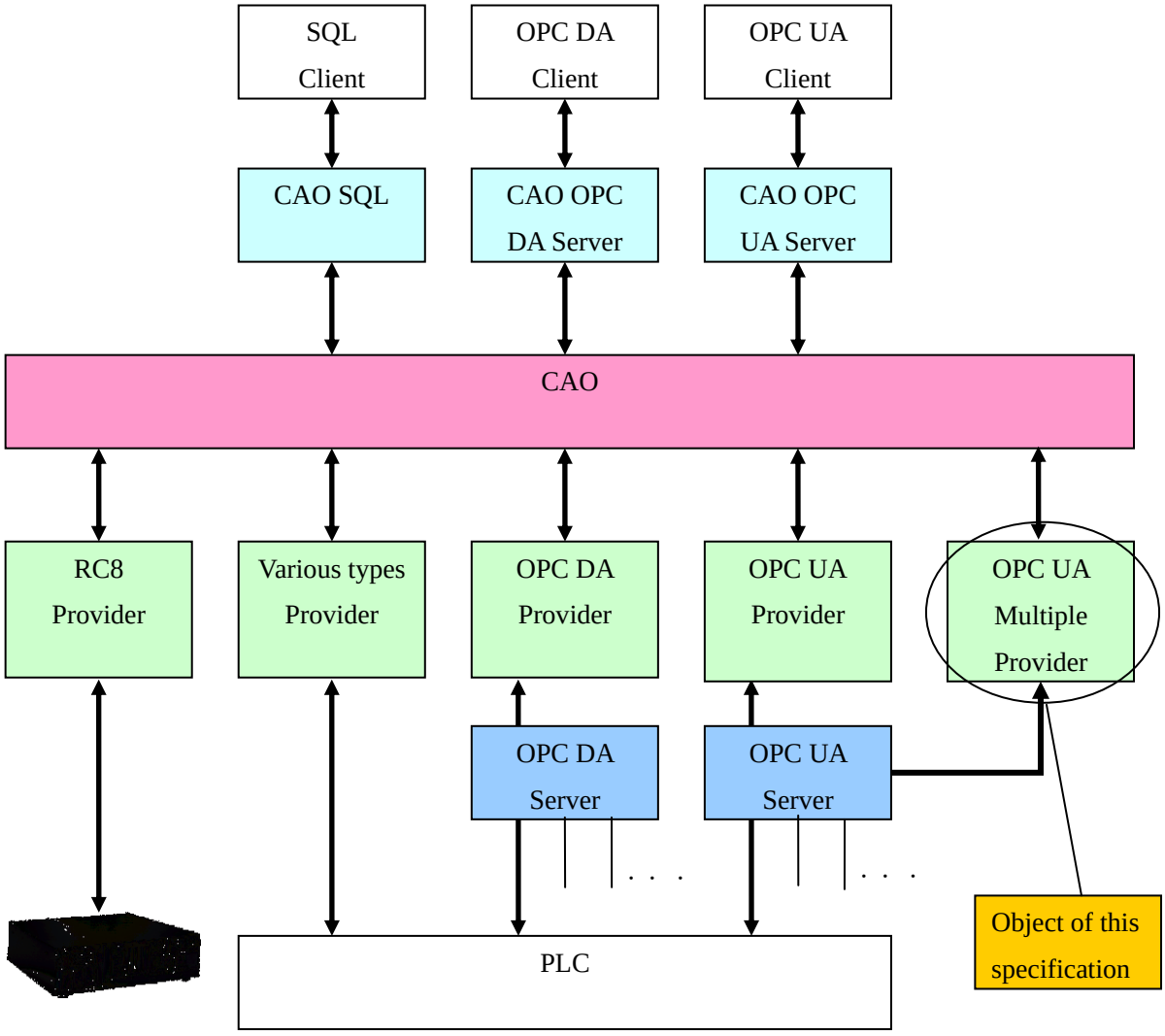


Fig. 2-1 CAO-providers for OPC UA connectivity

This OPC UA Multiple provider allows OPC UA servers to refer to Item values from CaoVariable objects. The difference between OPC DA providers and OPC UA Multiple providers is the same, except that they are not standards-compatible.

Table 2-1 OPC UA Multiple Providers

File name	CaoProvOPCUAMultiple.dll
ProgID	CaoProv.OPCUAMultiple
Registry registration[1]	regsvr32 CaoProvOPCUAMultiple.dll
Deletion of Registry Registration	regsvr32 /u CaoProvOPCUAMultiple.dll

[1] If it is installed by ORiN SDK, it does not need to be registered/deleted manually.

2.2. Method properties

2.2.1. CaoWorkspace::AddController method

Specifications for AddController methods for CaoWorkspace classes of CAOs are shown below.

```
AddController
(
    "<Controller>",                 // Controller name
    "CaoProv.OPCUAMultiple",            // Provider name. Fixed.
    "<machine name>",                   // The name of the provider's running machine.
    "<Option>"                      // Option string (OPC UA Multiple option)
)
```

Where <provider name> is fixed and <execution machine name> is equivalent to other providers.

The format of <option> is as follows:

<Option>::= Server=<OPC UA server URL>[,AccessPath=<access path>][,Security=<security policy>:<security mode>][,Der=<security certificate file name>][,Pem=<security private key file name>][,Password=<security password>][,User=<user name>:<password>][,Certificate=<user authentication certificate file name>][,PrivateKey=<user authentication private key file name>][,TrustServer=<True|False>]

Specifying <OPC UA server URL> is mandatory and you enter the URL of OPC UA server. Parameters enclosed in "[]" are not mandatory, otherwise the default value is set. The interpretation of the settings depends on OPC UA servers being called. See Table 2-2 below for additional options.

Format AddController(<bstrCtrlName:BSTR>,<bstrProvName>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

<bstrCtrlName>	:	[in] Controller name (VT_BSTR) Specify an arbitrary character string.
<bstrProvName>	:	[in] Provider ProgID (VT_BSTR) Specify a fixed string of "CaoProv.OPCUAMultiple"
<bstrPCName>	:	[in] PC name (VT_BSTR) Specify the PC name for remote connection. For a normal local connection, specify a blank string ("").
<bstrOption>	:	[in] Option string (VT_BSTR) Specify an option string to ensure the connection. Each option is specified in the following format, separated by commas (,): <OptionName>=<Value>,<OptionName>=<Value>,..

Example："Server=opc.tcp://192.168.188.128:4890/CaoOPCUA, AccessPath=OPCUA.CAO/DataStore"

The following is an option list for the <bstrOption> option string.

Table 2-2 CaoWorkspace::AddController Option-String

<table>
<tr><th>Option</th><th>Meaning</th></tr>
<tr><td>Server=<OPC UA ServerURL></td><td>OPC UA Servers URL.[Mandatory]
Example : "Server=opc.tcp://192.168.188.128:4890/CaoOPCUA"</td></tr>
<tr><td>AccessPath = [Access path]</td><td>Default access path.
It can be set individually for AccessPath of AddVariable. If it is not set individually, the value set here is used.</td></tr>
<tr><td>Security = [Security Policy [:Security Mode]]</td><td>Security settings . The following values can be specified: (Default: 0:0)
Security Policy:
<table>
<tr><th>Setting</th><th>Value</th><th>Description</th></tr>
<tr><td>None</td><td>0</td><td>No security.</td></tr>
<tr><td>Basic128Rsa15</td><td>1</td><td>Deprecated. This feature is retained for compatibility.[2]</td></tr>
<tr><td>Basic256</td><td>2</td><td>Deprecated. This feature is retained for compatibility. 2[2]</td></tr>
<tr><td>Basic256Sha256</td><td>3</td><td>Average security.</td></tr>
<tr><td>Aes128Sha256RsaOaep</td><td>4</td><td>Higher security</td></tr>
<tr><td>Aes256Sha256RsaPss</td><td>5</td><td>Best security.</td></tr>
</table>
Security Mode:
<table>
<tr><th>Setting</th><th>Value</th><th>Description</th></tr>
<tr><td>None</td><td>0</td><td>No security.</td></tr>
<tr><td>Sign</td><td>1</td><td>Sign the message but do not encrypt it.</td></tr>
<tr><td>SignAndEncrypt</td><td>2</td><td>Sign and encrypt the message.</td></tr>
</table>
You can configure the following combinations:
<table>
<tr><th>Security policy</th><th>Security Mode</th></tr>
<tr><td>None</td><td>None</td></tr>
<tr><td>Basic128Rsa15</td><td>Sign</td></tr>
<tr><td>Basic128Rsa15</td><td>SignAndEncrypt</td></tr>
</table></td></tr>
</table>

[2] A vulnerability has been discovered in cryptographic algorithms. Avoid use.

<table>
<tr><td></td><td>
<table>
<tr><td>Basic256</td><td>Sign</td></tr>
<tr><td>Basic256</td><td>SignAndEncrypt</td></tr>
<tr><td>Basic256Sha256</td><td>Sign</td></tr>
<tr><td>Basic256Sha256</td><td>SignAndEncrypt</td></tr>
<tr><td>Aes128Sha256RsaOaep</td><td>Sign</td></tr>
<tr><td>Aes128Sha256RsaOaep</td><td>SignAndEncrypt</td></tr>
<tr><td>Aes256Sha256RsaPss</td><td>Sign</td></tr>
<tr><td>Aes256Sha256RsaPss</td><td>SignAndEncrypt</td></tr>
</table>

If the security policy is 0, you can omit the security mode. In this case, the security mode is 0.
</td></tr>
<tr><td>Der=[security certificate file name]</td><td>Specifies the file name of the certificate for security. (default: empty)</td></tr>
<tr><td>Pem=[security private key file name]</td><td>Specifies the file name of the private key for security. (default: empty)</td></tr>
<tr><td>Password = [security password]</td><td>Specify a security password. (default: empty)</td></tr>
<tr><td>User = [username:password]</td><td>Specify the user name and password for user authentication. (default: empty)</td></tr>
<tr><td>Certificate = [user-authentication-certificate-file-name]</td><td>Specifies the file name of the certificate when authenticating with a certificate. (default: empty)</td></tr>
<tr><td>PrivateKey = [user-authenticated private key filename]</td><td>Specifies the file name of the private key when authenticating with a certificate. (default: empty)</td></tr>
<tr><td>TrustServer=[<True|False>]</td><td>Specifies what happens if the server certificate does not exist in the certificate trust list. (Defaults to True)
True: Trust and connect.
False: Not trusted and not connected.</td></tr>
</table>

Usage

example

```
Private CCaoController _caoController = null;

_caoController = _caoWorkspace.AddController("OpcUa", "CaoProv.OPCUAMultiple", "",
      "Server=opc.tcp://192.168.188.128:4890/CaoOPCUA,AccessPath=OPCUA.CAO/DataStore");
```

2.2.2. CaoController::AddVariable method

The specifications of CaoController::AddVariable methods of CAOs are shown below.

```
CaoController::AddVariable
(
    "<variable name>",                          // Variable Name
    "<Option>"              // Option string
)
```

If you are using OPC UA Multiple providers, set this parameter as follows:

<variable name>::=<item ID>

<Option>::=[[AccessPath = <Access Path>][,RequestType = <Receive Variable Type>] [,NamespaceIndex = <Namespace Index>,IdentifierType = <Identifier Type>,Identifier = <Identifier>]]

Parameters enclosed in "[]" are not mandatory, otherwise the default value is set. The interpretation of the settings depends on OPC UA servers being called. The receiving variable type is VT_TYPE. See Table 2-4 for VT_TYPE and its numeric value. See Table 2-3 below for additional options.

See 2.2.3 for the available system variables.

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

Table 2-3: CaoController::AddVariable Option-String

<table>
<tr><th>OPC UA Multiple Optional Name</th><th>Description</th></tr>
<tr><td>AccessPath [=<access path>]</td><td>Access path settings. (Default: AccessPath option-value of the parent controller)
Overwrites the values set by AccessPath options in AddController.</td></tr>
<tr><td>RequestType [=<receiving variable type>]</td><td>Setting the receiving variable type. (Default: VT_EMPTY)</td></tr>
<tr><td>NamespaceIndex [=<namespace index>]</td><td>Specifies the namespace of the node ID defined on the server. (Default: Not specified)
(See 2.2.2.1)</td></tr>
<tr><td>IdentifierType [=<identifier type>]</td><td>Specifies the data type of the identifier. (Default: Not specified)
<table>
<tr><th>Setting</th><th>Value</th><th>Description</th></tr>
<tr><td>Numeric value</td><td>0</td><td>Specifies an identifier as a number.</td></tr>
<tr><td>Character</td><td>1</td><td>Specifies the identifier as a string.</td></tr>
</table>
</td></tr>
</table>

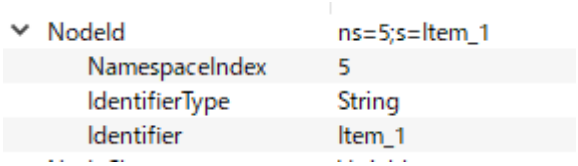
	string		
	(See 2.2.2.1)		
Identifier [=<identifier>]	Specifies the identifier of the node. (Default: Not specified) (See 2.2.2.1)		

2.2.2.1. Node ID

If you specify a namespace index identifier type identifier, specify it by node ID. When specifying by node ID, if all three of these types are not specified, an error will occur.

When specifying by node ID, searching by item name and access path is not performed.

Example)



For the items set above on OPCUA servers,

NamespaceIndex=5,IdentifierType=1,Identifier=Item_1

It becomes.

Table 2-4: VARIANT Type Datatypes (Some)

Data type	Specified value	In bytes	Description
VT_I2	2	2	Single-precision (16-bit) integer
VT_I4	3	4	Double (32-bit) integer
VT_R4	4	4	Single precision (32-bit) floating point
VT_R8	5	8	Double (64-bit) floating point
VT_CY	6	8	Same currency as VT_UI8.
VT_DATE	7	8	The total date and time since 1899/12/30, which is the same as VT_R8.
VT_BSTR	8	Variable	Character string UNICODE characters and NULL terminators
VT_BOOL	11	2	Same as VT_I2 0:FALSE,-1::TRUE
VT_VARIANT	12	Variable	Variant type (used only for variant type arrays)
VT_UI1	17	1	Unsigned character
VT_ARRAY	8192	Variable	One-dimensional array of the above data types

Usage

example

```
Private CCaoVariable _caoVariable1 = null;
Private CCaoVariable _caoVariable2 = null;
Private CCaoVariable _caoVariable3 = null;

_caoVariable1 = _caoController.AddVariable("Item_I4", "");
_caoVariable2 = _caoController.AddVariable("Item_R8", "");
_caoVariable3 = _caoController.AddVariable("Item_BSTR", "");
```

2.2.3. CaoController::property get_VariableNames

This get_VariableNames property retrieves a list of system variables.

2.2.4. CaoVariable::property get_DateTime

This get_DateTime property gets the timestamp of the item ID.

Note that this is the timestamp obtained when you executed the get_Value property.

2.2.5. CaoVariable::get_Value Property

This get_Value property gets the cached value in CaoVariable.

Normally, the cache value is not updated.

Use the "CaoController::Execute("MultipleRead" command to refresh the cached value to obtain the value on the most recent OPCUA server.CaoController::Execute("MultipleRead") Command

3. Command Reference

3.1. Controller classes

Table 3-1 CaoController::Execute Commands

Command	Function	Page
MultipleRead	Batch retrieves variable values.	14

3.1.1. CaoController::Execute("MultipleRead") Command

Batch retrieves variable values.

This command acquires the values of all CaoVariable on the controllers at once and updates each cache value.

Format MultipleRead()

Argument : None
Return value : None

Usage

example

```
_caoController.Execute("MultipleRead", null);
```

4. Error code

OPC UA Multiple providers define the following unique error codes: ORiN2 common errors are described in the Error Codes section of ORiN2 Programming Guide.

Table 4-1 List of unique error codes

Error name	Error Number	Description
S_OPCUA_TRUST	0x00100800	The server certificate was trusted.
E_OPCUA_NOT_TRUST	0x80100800	The server certificate was not trusted.
E_OPCUA_PKI_NOTFOUND	0x80100801	PKI (Public Key Infrastructure) was not found.
E_OPCUA_FAILED_USERCERTIFICATION	0x80100802	User authentication failure
E_OPCUA_FAILED_START	0x80100803	Failed to start client application.
E_OPCUA_FAILED_CONNECTSESSION	0x80100804	Failed to connect session.
E_OPCUA_FAILED_GETENDPOINT	0x80100805	Failed to get endpoint.
E_OPCUA_FAILED_OPENUSERCERTIFICATE	0x80100806	Could not open user certificate.
E_OPCUA_FAILED_LOADUSERPRIVATEKEY	0x80100807	Failed to load user private key.
E_OPCUA_ENDPOINT_NOTFOUND	0x80100808	The specified endpoint did not exist.
E_OPCUA_FAILED_ADDSESSION	0x80100809	Failed to add session.
E_OPCUA_CERTIFICATE_SETTINGNOW	0x8010080a	An attempt was made to open another instance while a separate certificate is being configured.
E_OPCUA_OTHERINSTANCE_EXIST	0x8010080b	An attempt was made to set up a separate certificate with other instances present.
E_OPCUA_INVALID_ACCESSPATH	0x8010080c	The access path is invalid.
E_OPCUA_VARIABLENAME_NOTEXIST	0x8010080d	A variable name that does not exist.
E_OPCUA_TYPEMISMATCH	0x8010080e	The types do not match.
E_OPCUA_BADVARTYPE	0x8010080f	Invalid type.
E_OPCUA_SECURITY_CHECKS_FAILED	0x80100810	An error occurred while validating security.
E_OPCUA_NODEID_NOTEXIST	0x80100811	Node ID that does not exist.
E_OPCUA_NODEID_NOTENOUGHOPTIONS	0x80100812	The option string for specifying the node ID is insufficient.
E_OPCUA_BAD_MULTIPLE_READ_ERROR	0x80100813	An error occurred during batch acquisition.

5. About Certificate Files

OPC UA Multiple providers have the following certificates:

- Application certificate
- Certificate for Certificate Authentication

This section describes these certificates and how to create them.

5.1. Application certificate

The application's certificate and private key file must exist for OPC UA Multiple providers to start. This certificate is also used for security connections.

If you create it in the step described later, it will be created in the "certs" folder under "¥PKI¥ store" in the application installation folder and in the "private" folder.

"cert_multiple_client_self_signed.der" in the "certs" folder is the certificate file.

"private_key_multiple_client_self_signed.pem" in the "private" folder is the private key file.

It can also be specified using the Der, Pem, and Password options.

※If these files are specified, multiple controllers cannot be handled.

Example)

・If a controller exists, you cannot add a controller with an application certificate.

・You cannot add a new controller while using a controller with an application certificate.

If the correct files do not exist or are not specified correctly, the provider startup will fail.

5.2. Certificate for Certificate Authentication

Authentication can be done in the following ways:

- Authenticating with Anonymous (Anonymous) Users
- Authentication by user name
- Authentication by means of certificates

This certificate is required for Certificate Authentication.

When you create this file in the steps below, it will be created in the "certs" folder and "private" folder under "¥ UserCertificate" in the application installation folder.

"cert_multiple_client_user.der" in the "certs" folder is the certificate file.

"private_key_multple_client_user.pem" in the "private" folder is the private key file.

If the correct files do not exist or are not specified correctly, certificate authentication fails.

5.3. Preparation of certificates

This section describes how to create the above certificates.

Follow the procedure below to create.

①Launch the create_store.bat.

②It is automatically created in the setting save destination.

6. Sample program

6.1. CaoOPCUA

The following is a sample of connecting to CaoOPCUA, collectively acquiring the value of the item with the batch acquisition button, and displaying [Variable Name], [Type], and [Value] in the list view.

6.1.1. Access path usage

List 6-16 Form1.cs

```
Using System;
Using System.Windows.Forms;

Using ORiN2.ManagedCAO;

Namespace Multiple
{
    Public partial class Form1 : Form
    {
        Private CCaoEngine _caoEngine = null;
        Private CCaoWorkspace _caoWorkspace = null;
        Private CCaoController _caoController = null;
        Private CCaoVariable _caoVariable1 = null;
        Private CCaoVariable _caoVariable2 = null;
        Private CCaoVariable _caoVariable3 = null;

        Public Form1()
        {
            InitializeComponent();
        }

        Private void Form1_Load(object sender, EventArgs e)
        {
            _caoEngine = new CCaoEngine();
            If (_caoEngine != null)
            {
                _caoWorkspace = _caoEngine.Workspaces[0];
            }
            If (_caoWorkspace != null)
            {
                _caoController = _caoWorkspace.AddController("OpcUa",
                    "CaoProv.OPCUAMultiple", "",
                    "Server=opc.tcp://192.168.188.128:4890/CaoOPCUA"
                    + ",AccessPath=OPCUA.CAO/DataStore");
            }
            If (_caoController != null)
            {
                _caoVariable1 = _caoController.AddVariable("Item_I4", "");
                _caoVariable2 = _caoController.AddVariable("Item_R8", "");
                _caoVariable3 = _caoController.AddVariable("Item_BSTR", "");
            }
```

```
}

Private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    If (_caoVariable1 != null)
    {
        If (_caoController != null)
        {
            _caoController.Variables.Remove(_caoVariable1.Index);
        }
        _caoVariable1 = null;
    }
    If (_caoVariable2 != null)
    {
        If (_caoController != null)
        {
            _caoController.Variables.Remove(_caoVariable2.Index);
        }
        _caoVariable2 = null;
    }
    If (_caoVariable3 != null)
    {
        If (_caoController != null)
        {
            _caoController.Variables.Remove(_caoVariable3.Index);
        }
        _caoVariable3 = null;
    }
    If (_caoController != null)
    {
        If (_caoWorkspace != null)
        {
            _caoWorkspace.Controllers.Remove(_caoController.Index);
        }
        _caoController = null;
    }
    If (_caoWorkspace != null)
    {
        _caoWorkspace = null;
    }
    If (_caoEngine != null)
    {
        _caoEngine.Dispose();
        _caoEngine = null;
    }
}

Private void btnGet_Click(object sender, EventArgs e)
{
    LvItems.Items.Clear();
    If (_caoController != null)
    {
        _caoController.Execute("MultipleRead", null);
    }
    Var item1 = lvItems.Items.Add(_caoVariable1.Name);
```

```
                Item1.SubItems.Add(_caoVariable1.Value.GetType().ToString());
                Item1.SubItems.Add(_caoVariable1.Value.ToString());
                Var item2 = lvItems.Items.Add(_caoVariable2.Name);
                Item2.SubItems.Add(_caoVariable2.Value.GetType().ToString());
                Item2.SubItems.Add(_caoVariable2.Value.ToString());
                Var item3 = lvItems.Items.Add(_caoVariable3.Name);
                Item3.SubItems.Add(_caoVariable3.Value.GetType().ToString());
                Item3.SubItems.Add(_caoVariable3.Value.ToString());
            }
        }
    }
```

6.1.2. Use node ID

List 6-26 Form1.cs

```
Using System;
Using System.Windows.Forms;

Using ORiN2.ManagedCAO;

Namespace Multiple
{
    Public partial class Form1 : Form
    {
        Private CCaoEngine _caoEngine = null;
        Private CCaoWorkspace _caoWorkspace = null;
        Private CCaoController _caoController = null;
        Private CCaoVariable _caoVariable1 = null;
        Private CCaoVariable _caoVariable2 = null;
        Private CCaoVariable _caoVariable3 = null;

        Public Form1()
        {
            InitializeComponent();
        }

        Private void Form1_Load(object sender, EventArgs e)
        {
            _caoEngine = new CCaoEngine();
            If (_caoEngine != null)
            {
                _caoWorkspace = _caoEngine.Workspaces[0];
            }
            If (_caoWorkspace != null)
            {
                _caoController = _caoWorkspace.AddController("OpcUa",
                    "CaoProv.OPCUAMultiple", "",
                    "Server=opc.tcp://192.168.188.128:4890/CaoOPCUA");
            }
            If (_caoController != null)
            {
                _caoVariable1 = _caoController.AddVariable("Item1",
                    "NamespaceIndex=5,IdentifierType=1,Identifier=Item_I4");
                _caoVariable2 = _caoController.AddVariable("Item2",
                    "NamespaceIndex=5,IdentifierType=1,Identifier=Item_R8");
                _caoVariable3 = _caoController.AddVariable("Item3",
                    "NamespaceIndex=5,IdentifierType=1,Identifier=Item_BSTR");
            }
        }

        Private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            If (_caoVariable1 != null)
            {
                If (_caoController != null)
```

```
            {
                _caoController.Variables.Remove(_caoVariable1.Index);
            }
            _caoVariable1 = null;
        }
        If (_caoVariable2 != null)
        {
            If (_caoController != null)
            {
                _caoController.Variables.Remove(_caoVariable2.Index);
            }
            _caoVariable2 = null;
        }
        If (_caoVariable3 != null)
        {
            If (_caoController != null)
            {
                _caoController.Variables.Remove(_caoVariable3.Index);
            }
            _caoVariable3 = null;
        }
        If (_caoController != null)
        {
            If (_caoWorkspace != null)
            {
                _caoWorkspace.Controllers.Remove(_caoController.Index);
            }
            _caoController = null;
        }
        If (_caoWorkspace != null)
        {
            _caoWorkspace = null;
        }
        If (_caoEngine != null)
        {
            _caoEngine.Dispose();
            _caoEngine = null;
        }
    }

    Private void btnGet_Click(object sender, EventArgs e)
    {
        LvItems.Items.Clear();
        If (_caoController != null)
        {
            _caoController.Execute("MultipleRead", null);
        }
        Var item1 = lvItems.Items.Add(_caoVariable1.Name);
        Item1.SubItems.Add(_caoVariable1.Value.GetType().ToString());
        Item1.SubItems.Add(_caoVariable1.Value.ToString());
        Var item2 = lvItems.Items.Add(_caoVariable2.Name);
        Item2.SubItems.Add(_caoVariable2.Value.GetType().ToString());
        Item2.SubItems.Add(_caoVariable2.Value.ToString());
        Var item3 = lvItems.Items.Add(_caoVariable3.Name);
        Item3.SubItems.Add(_caoVariable3.Value.GetType().ToString());
```

```
                    Item3.SubItems.Add(_caoVariable3.Value.ToString());
                }
            }
        }
```