



DeviceXPlorer

Client Sample Guide

OPC DataAccess2.05A/3.0

User's Manual
Revision P

Forward

Caution

Read this manual carefully and understand all contents before operating this product.

This manual does not warrant that the quality and function of this product will meet customer's requirements. This manual intends to explain the details of this product.

TAKEBISHI Corporation (hereinafter referred to as "TAKEBISHI") does not grant Customer to reproduce the whole or a part of this manual.

The author reserves the exclusive right to make changes/modifications to this manual.

Copyright and license

Copyright of all SOFTWARE and online manual included in CD shall belong to TAKEBISHI.

TAKEBISHI forbids to copy the SOFTWARE included in CD and transfer the right of and sell and distribute SOFTWARE (including provides through the network of computer communication.) to third party.

About Trademark

All other brands may be trademarks of their respective owners.

Table of Contents

1 OPC Client.....	4
1.1 Prog.ID	4
1.2 Test Client Program (OPC Client.exe).....	5
1.2.1 Operation	5
1.3 Visual C++ Sample (VcSampleOpc).....	11
1.3.1 Operation	11
1.4 Visual Basic 6.0 Sample (for Automation Interface)	14
1.4.1 Operation	14
1.4.2 Setting of Development Environment	15
1.4.3 Program Example	16
1.5 Visual Basic .NET Sample (for OPC Automation Interface)	20
1.5.1 Operation	20
1.5.2 Setting of Development Environment	21
1.5.3 Note in building sample on Windows for x64.....	22
1.5.4 Program Example	23
1.6 Visual Basic .NET Sample (for OPC Custom Interface)	25
1.6.1 Operation	25
1.6.2 Setting of Development Environment	26
1.6.3 Note in building sample on Windows for x64.....	27
1.6.4 Program Example	28
1.7 Visual C# Sample (for OPC Custom Interface)	32
1.7.1 Operation	32
1.7.2 Setting of Development Environment	33
1.7.3 Note in building sample on Windows for x64.....	34
1.7.4 Program Example	35
1.8 DXP Development Support Tool.....	39
1.8.1 Client Component (for .NET)	39
1.8.2 Simple API (for .NET)	48
1.9 EXCEL Communication Support Tool (OPCFunction)	49
1.9.1 Operation	49
1.10 EXCEL Sample (OPCDataAccess).....	53
1.10.1 Operation	53
1.10.2 Setting of Development Environment	56
1.10.3 Program Example	56
1.10.4 How to uninstall.....	57
2 DDE Client.....	58
2.1 Test Client Program (DDE Test Client).....	58
2.1.1 Operation	58

1 OPC Client

1 OPC Client

1.1 Prog.ID

Prog.ID of our OPC Server is as follows.

OPC Server	Version	Prog.ID
DeviceXPlorer OPC Server 6	Ver6	Takebishi.Dxp or Takebishi.Dxp.6
DeviceXPlorer OPC Server 5	Ver5	Takebishi.Dxp or Takebishi.Dxp.5
DeviceXPlorer OPC Server	Ver4	Takebishi.Dxp.1
MELSEC OPC Server	Ver3	Takebishi.Melsec.1

1.2 Test Client Program (OPC Client.exe)

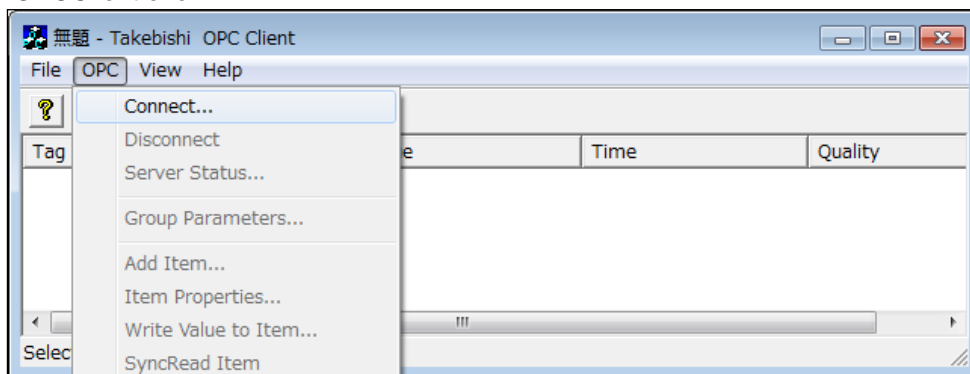
1.2 Test Client Program (OPC Client.exe)

The use of the OPC client program appended to the product is shown.

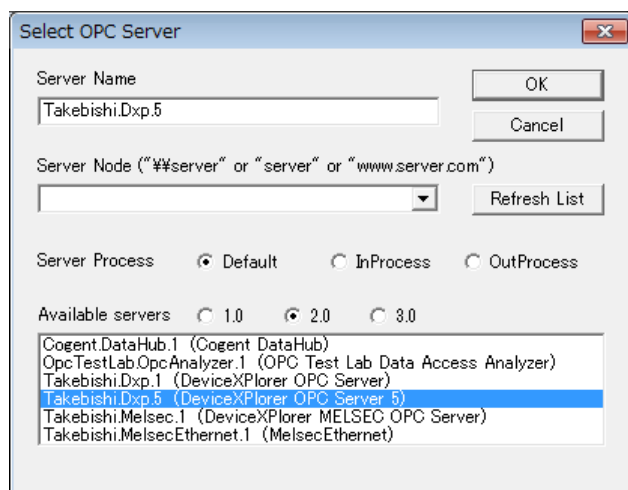
1.2.1 Operation

[Connection of Server]

To start OPC Client, from “C:\Program Files\TAKEBISHI\DeviceXPlorer OPC Server 5\Bin”, start “OPCClient.exe”.



On the OPC menu select "Connect".



[Server Process] is to select the method of managing the process of OPC Server. When “Default” is selected, it operates initial of OPC Server.

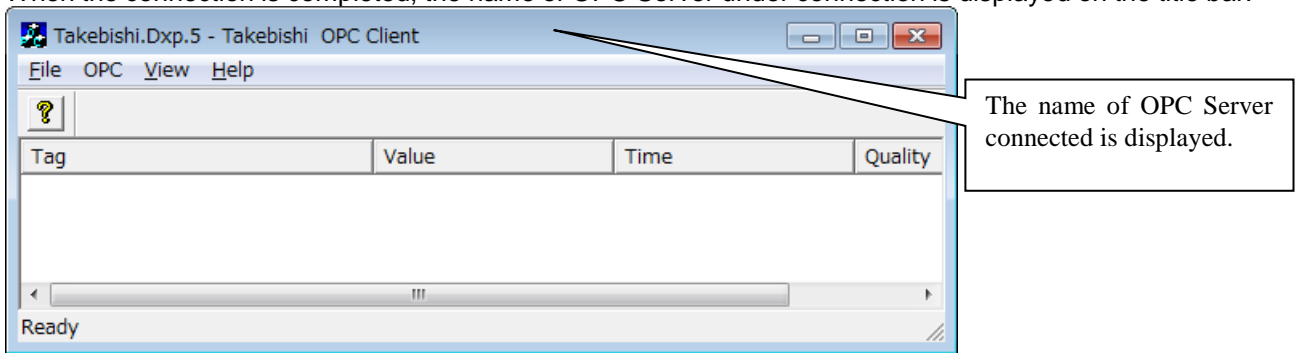
When the version of OPCDA is selected with [Available Server], the OPC Server to correspond with the version is Listed

Prog.ID of DeviceXPlorer OPC Server is “Takebishi.Dxp”. Other OPC Server’s Prog.ID is written at 1.1

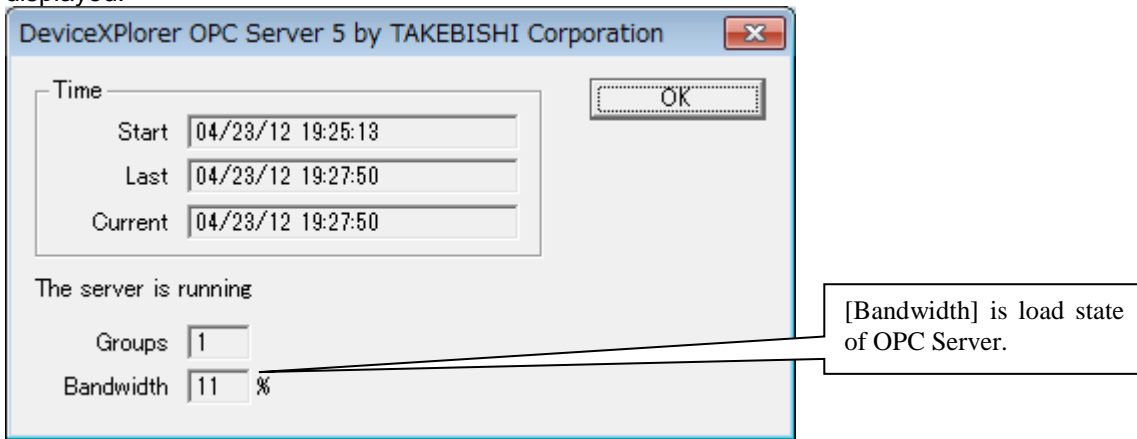
If Prog.ID of OPC Server is not listed, the server needs to be registered. This can be done from the register.bat file located in the installation directory.

1.2 Test Client Program (OPC Client.exe)

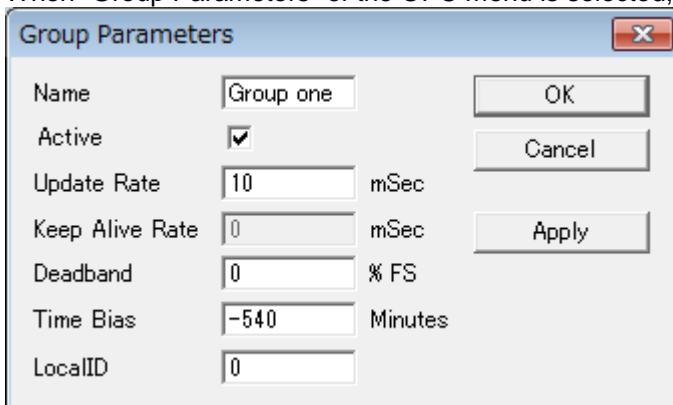
When the connection is completed, the name of OPC Server under connection is displayed on the title bar.



When "Server Status" of the OPC menu is selected, the state of OPC Server under connection is displayed.



When "Group Parameters" of the OPC menu is selected, the parameter of group is displayed.



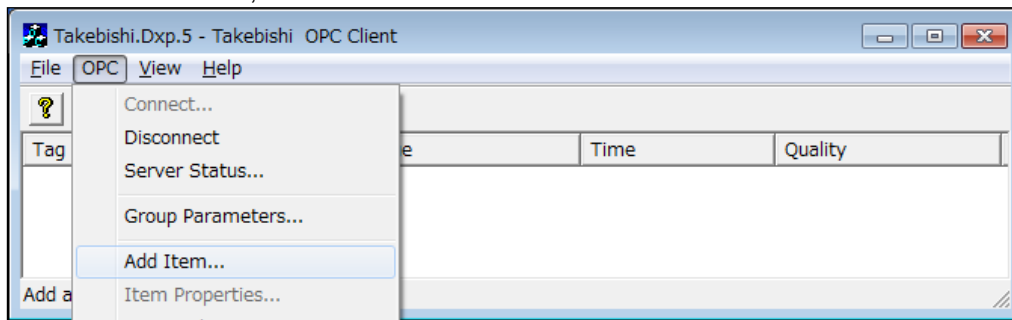
- [Update Rate]
Define how often the data will be updated. The minimum value for the OPC Server is 10 msec.
- [Active]
This check box controls the communication state between OPC Client and the server. If unchecked, communication will be inactive.
- [Dead Band]
The percent by which the data group must change before the OPC Server notifies this program.

To disconnect, select "Disconnect" from the OPC menu.

1.2 Test Client Program (OPC Client.exe)

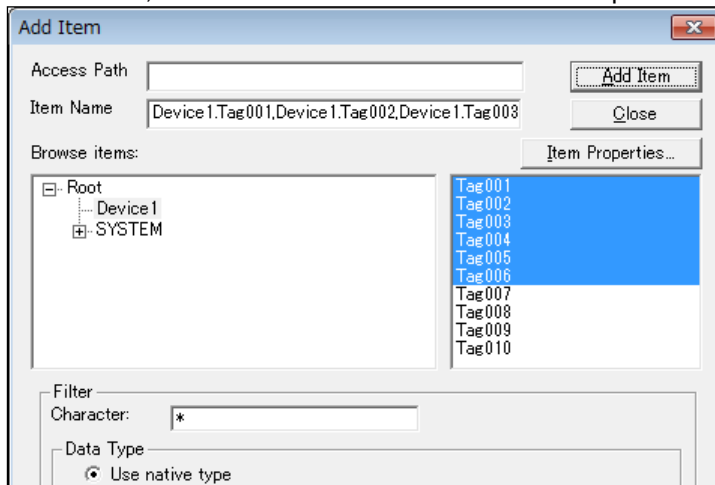
[Add Items]

From the OPC menu, select "Add Item.."

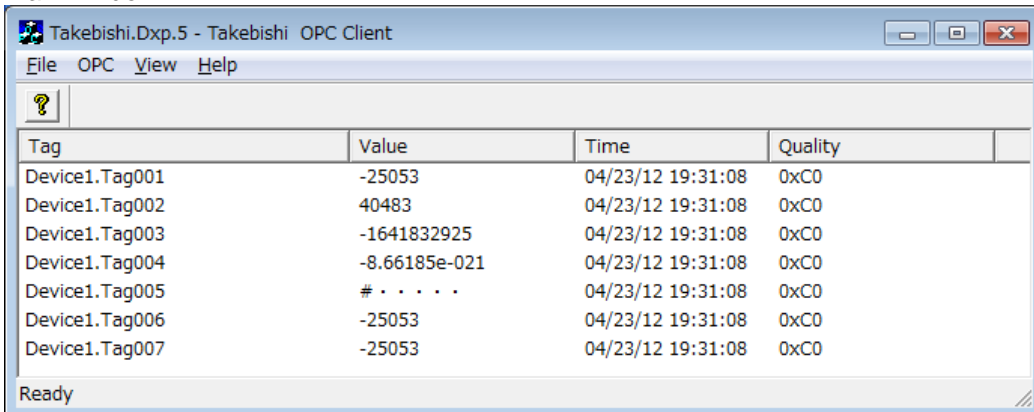


The defined devices, groups, and tag appear in the item browser.

To add item, select item from the item browser or input item in [Item Name], and click [Add Item] button.



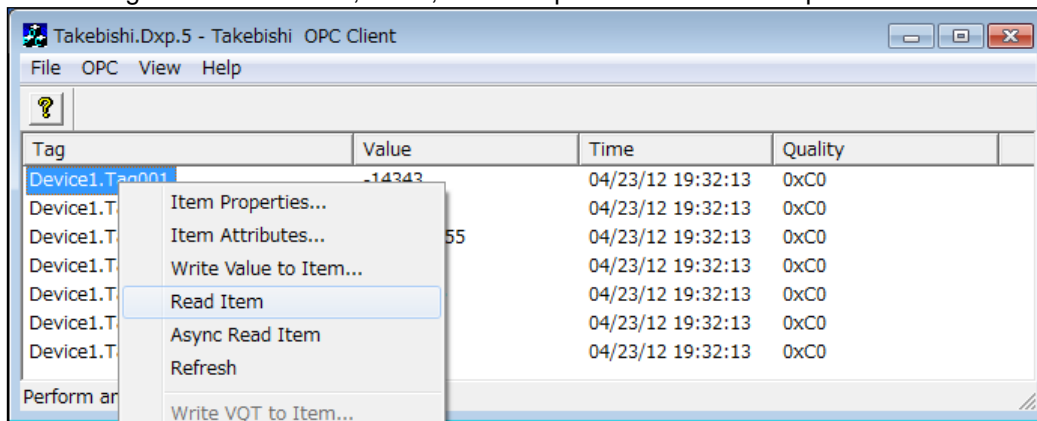
Once you have completed adding items, click the done button. The newly added items will be added to the main window.



1.2 Test Client Program (OPC Client.exe)

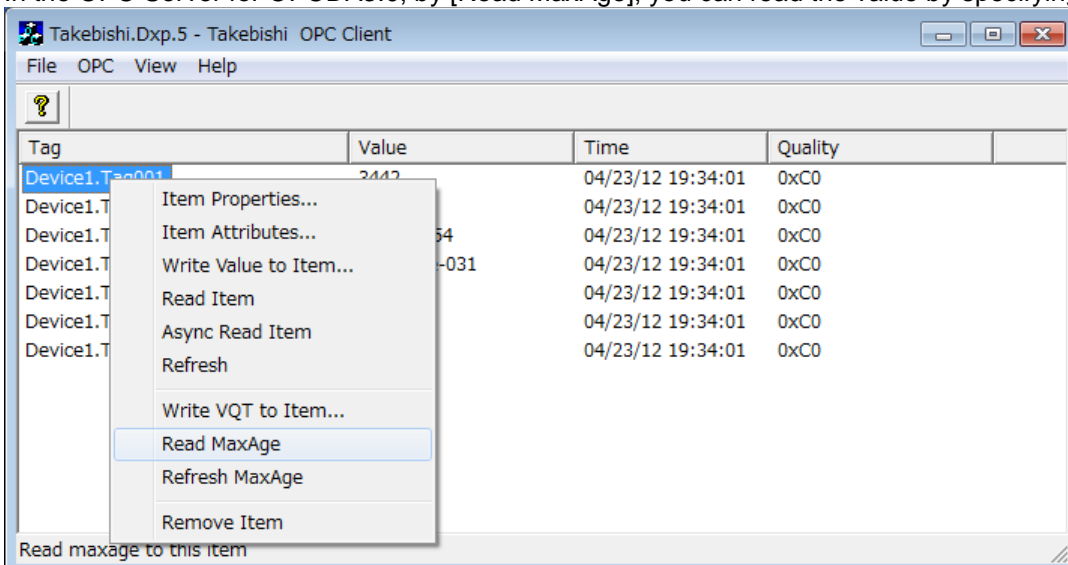
[Read and Write of Item]

Data change occurred in item, value, timestamp on window will be update.

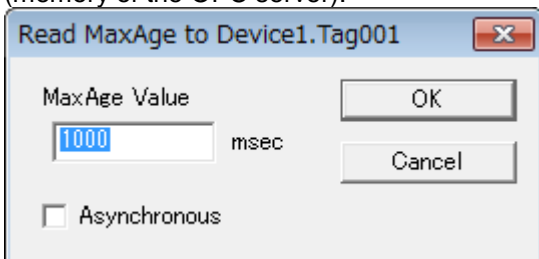


When you need to read value, select item, make it show OPC menu or item menu using right click and execute Read Item.

In the OPC Server for OPCDA3.0, by [Read MaxAge], you can read the value by specifying "Old" of data.



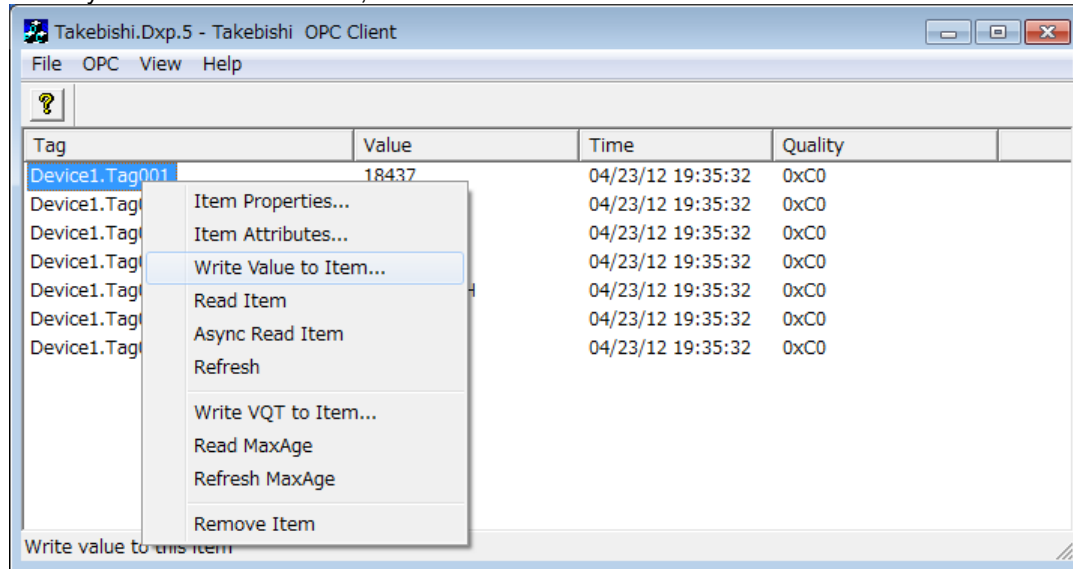
Specify time of "Old" to [MaxAge Value] and execute it. For example, If you specify 1000msec, when the data maintained in the OPC Server is older than 1000msec, the OPC Server acquires the value from device (PLC). And when the data is newer than 1000msec, the OPC Server returns the value of cache (memory of the OPC server).



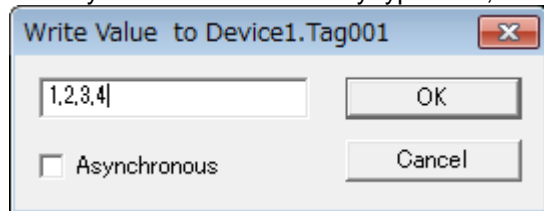
Select "Asynchronous" to execute asynchronously.

1.2 Test Client Program (OPC Client.exe)

When you need to write value, execute "Write Value to Item".

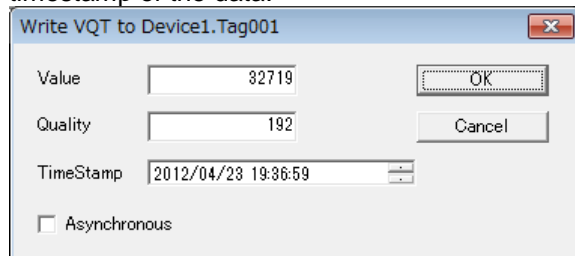


When you write value on array type data, values should be input separated by comma.



Select "Asynchronous" to execute asynchronously.

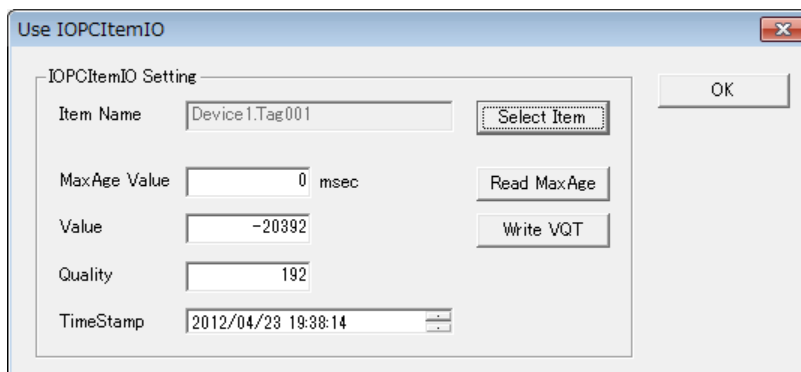
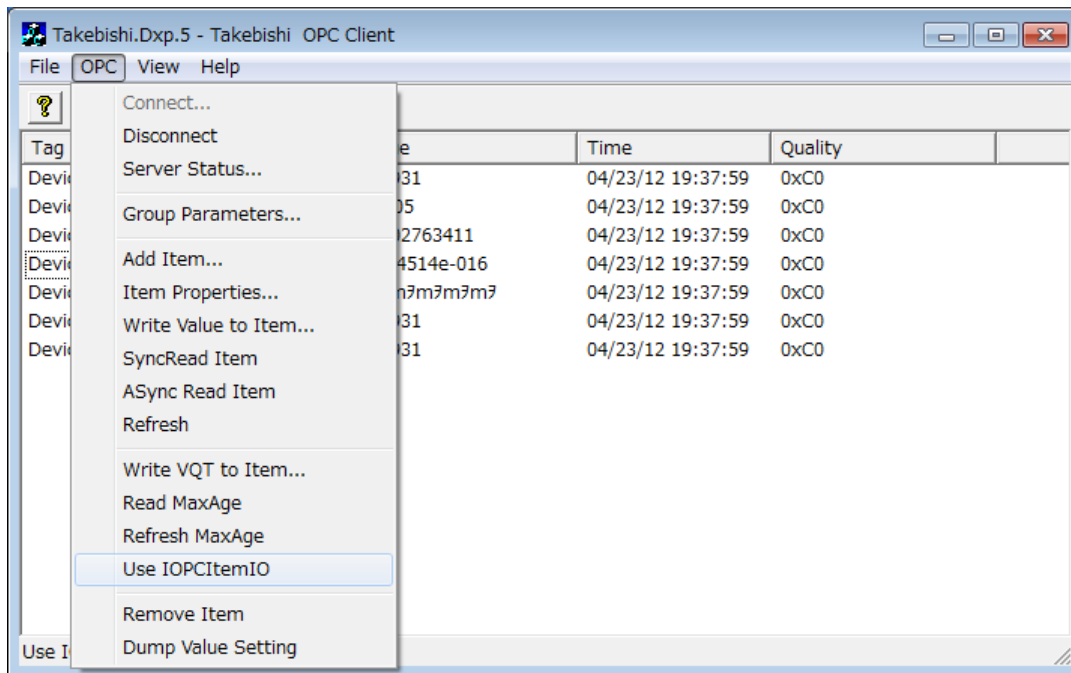
In the OPC Server for OPCDA3.0, by [Write VQT], you can written the value, the quality, and the timestamp of the data.



Select "Asynchronous" to execute asynchronously.

1.2 Test Client Program (OPC Client.exe)

In the OPC Server for OPCDA3.0, you can use the IOPCItemIO interface. When you use this interface, you can access the item directly doing neither the registration of the group nor the addition of the item.



- [Select Item] Select the item.
- [Read MaxAge] Specify time of "Old" and read the value.
- [Write VQT] Write the value, the quality, and the timestamp of the data.

1.3 Visual C++ Sample (VcSampleOpc)

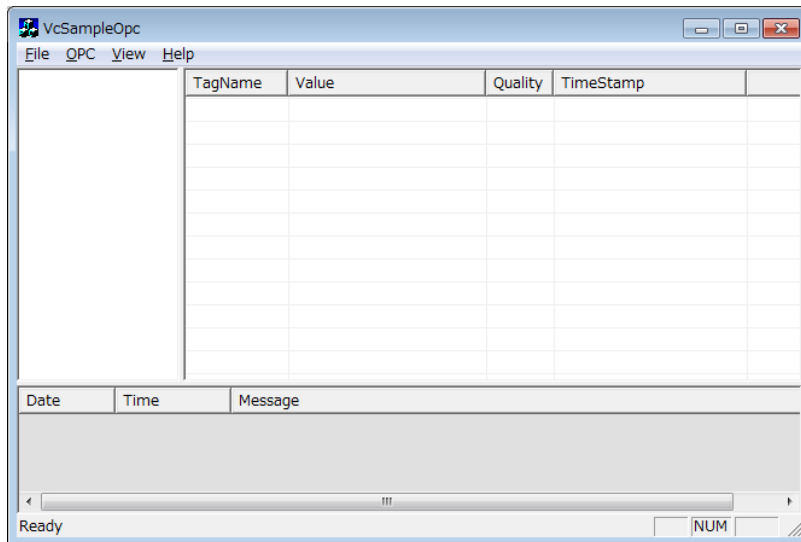
1.3 Visual C++ Sample (VcSampleOpc)

This sample program is to make the OPC client in the development environment of Visual C++.

1.3.1 Operation

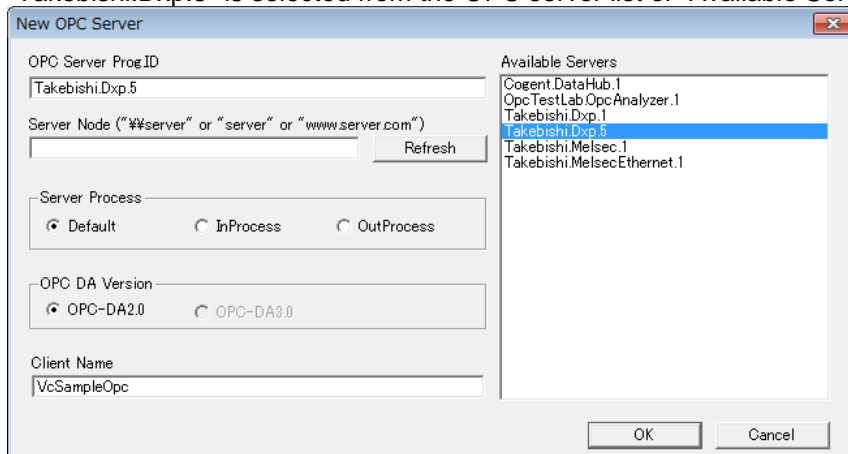
The sample program for Visual C++ is stored in the "\\Sample\VcSampleOpc" folder at the installation destination. When you execute "VcSampleOpc.exe", the following screen is displayed.

You can execute at "OPC Client" at "program" at Start Menu of Windows, too.



[Connection of Server]

When "Connect" is selected from the OPC menu of the OPC test client, the screen below is displayed. "Takebishi.Dxp.5" is selected from the OPC server list of "Available Servers" and "OK" is pushed.



[Server Process] is to select the method of managing the process of OPC Server. When "Default" is selected, it operates initial of OPC Server.

OPC Servers are Listed at [Available Server].

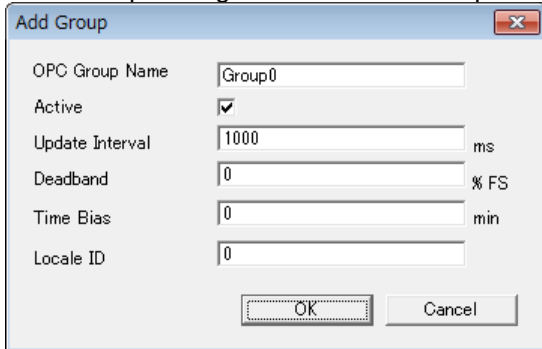
Prog.ID of DeviceXPlorer OPC Server is "Takebishi.Dxp". Other OPC Server's Prog.ID is written at 1.1

If Prog.ID of OPC Server is not listed, the server needs to be registered. This can be done from the register.bat file located in the installation directory.

1.3 Visual C++ Sample (VcSampleOpc)

[Adding OPC Group]

When "AddGroup" is selected from the OPC menu of the OPC test client, the screen below is displayed. .
OPC Group setting is done and "OK" is pushed.

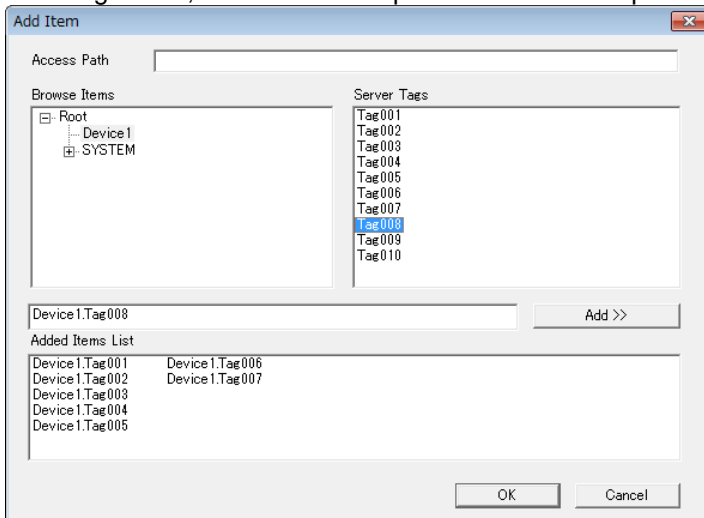


The "Add Group" dialog box contains the following fields and controls:

- OPC Group Name: Text box with "Group0" entered.
- Active: Check box, checked.
- Update Interval: Spin box with "1000" and unit "ms".
- Deadband: Spin box with "0" and unit "% FS".
- Time Bias: Spin box with "0" and unit "min".
- Locale ID: Spin box with "0".
- Buttons: "OK" and "Cancel".

[Adding Tags]

When "Add Item" is selected from the OPC menu of the OPC test client, the screen below is displayed. A hierarchical tree of "Browse Items" is opened, it selects from the definition ending tag displayed in the list of the right side, and "Add >>" is pushed and "OK" is pushed.

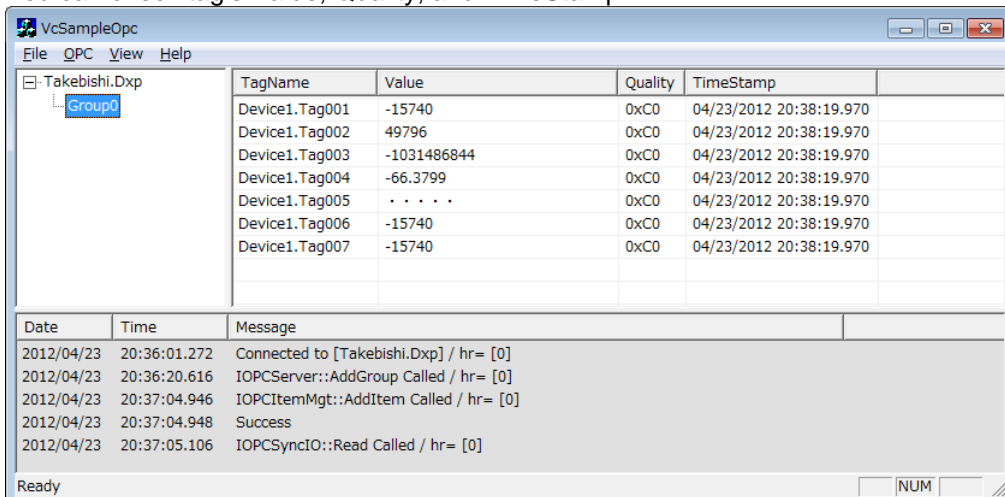


The "Add Item" dialog box contains the following elements:

- Access Path: Text box.
- Browse Items: Tree view showing "Root" > "Device1" > "SYSTEM".
- Server Tags: List box showing "Tag001" through "Tag010", with "Tag008" selected.
- Device1.Tag008: Text box showing the selected tag path.
- Add >>: Button.
- Added Items List: List box showing "Device1.Tag001" through "Device1.Tag005".
- Buttons: "OK" and "Cancel".

[Monitor the value]

You can check tag's Value, Quality, and TimeStamp.



The VcSampleOpc main window displays the following data:

TagName	Value	Quality	TimeStamp
Device1.Tag001	-15740	0xC0	04/23/2012 20:38:19.970
Device1.Tag002	49796	0xC0	04/23/2012 20:38:19.970
Device1.Tag003	-1031486844	0xC0	04/23/2012 20:38:19.970
Device1.Tag004	-66.3799	0xC0	04/23/2012 20:38:19.970
Device1.Tag005	0xC0	04/23/2012 20:38:19.970
Device1.Tag006	-15740	0xC0	04/23/2012 20:38:19.970
Device1.Tag007	-15740	0xC0	04/23/2012 20:38:19.970

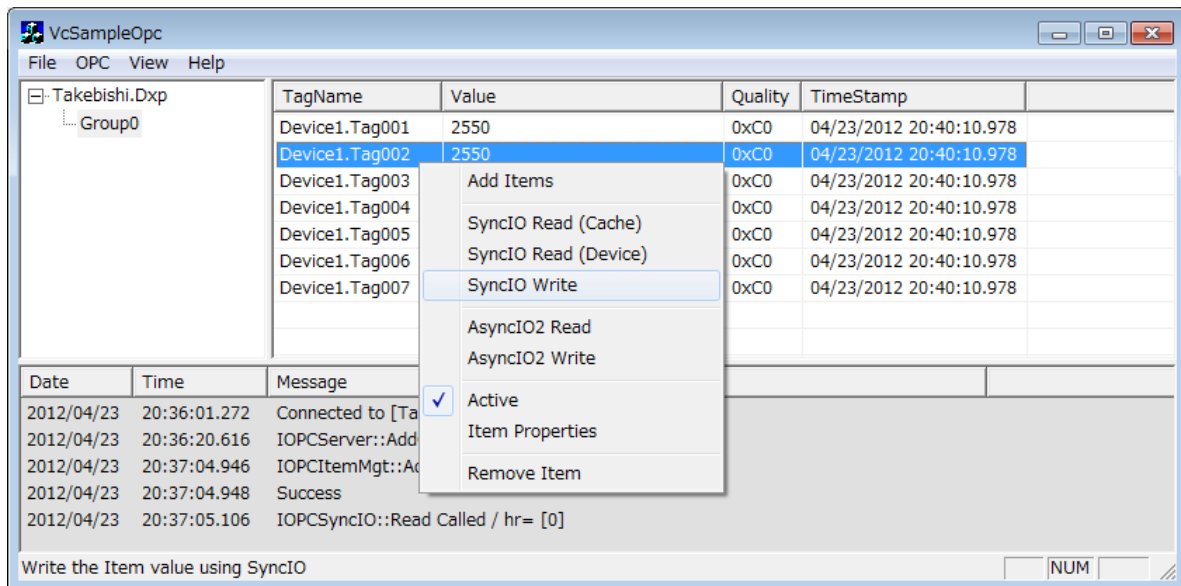
Date	Time	Message
2012/04/23	20:36:01.272	Connected to [Takebishi.Dxp] / hr= [0]
2012/04/23	20:36:20.616	IOPCServer::AddGroup Called / hr= [0]
2012/04/23	20:37:04.946	IOPCItemMgt::AddItem Called / hr= [0]
2012/04/23	20:37:04.948	Success
2012/04/23	20:37:05.106	IOPCSyncIO::Read Called / hr= [0]

Ready NUM

1.3 Visual C++ Sample (VcSampleOpc)

[Write Value to tag]

You can operate OPC function such as read item value and write item value from right-click menu.



1.4 Visual Basic 6.0 Sample (for Automation Interface)

1.4 Visual Basic 6.0 Sample (for Automation Interface)

This sample program is to make the OPC client in the development environment of Visual Basic 6.0.

1.4.1 Operation

The sample program for Visual Basic 6.0 is stored in the "\\OPC Sample\\VB6DAutoSample" folder at the installation destination. When you execute "Sample.exe", the following screen is displayed.

Item Name	Value	Date/Time	Quality
Device1.D0		TimeStamp	Quality
Device1.D1		TimeStamp	Quality
Device1.D2		TimeStamp	Quality
Device1.D3		TimeStamp	Quality
Device1.D4		TimeStamp	Quality
Device1.D5		TimeStamp	Quality
Device1.D6		TimeStamp	Quality
Device1.D7		TimeStamp	Quality

The function of each part is as follows.

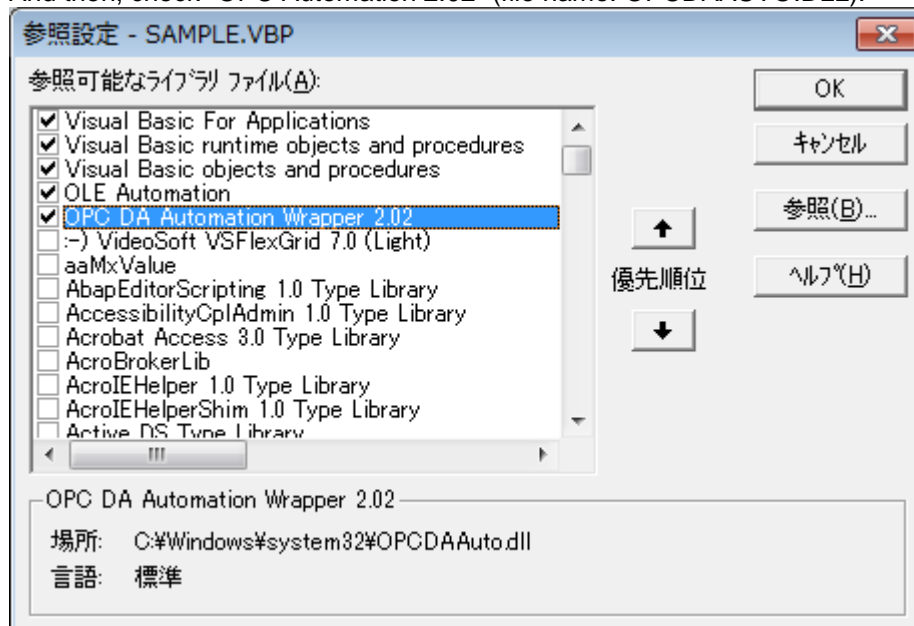
Function	Contents
Node Name	Input a target node name (machine name or IP address).
OPCServer Name	Select a OPC server name to connect. Prog.ID of DeviceXPlorer OPC Server is "Takebishi.Dxp". Other OPC Server's Prog.ID is written at 1.1
Update Rate	Input a cycle reading of the PLC data. The unit is "msec".
Item Name	Input a pre-defined device name and item name. (format is "device name.item name".) The default settings are between "Device1.D0" and "Device1.D7".
Connect	Connects with the OPC server. And the button will change to "Disconnect".
Advise	The group and data report become valid. The data in OPC server is read out every "Update Rate" cycle. Changing report will be sent to OPC client. And when you execute "AsyncRead" and "AsyncWrite" method, "Advise" must become valid.
Sync Read	Read and the value, the quality, and the timestamp of the tag will be retrieved from the OPC server and displayed.
Sync Write	Writes value with "Value" to the OPC server.
Async Read	It demands asynchronous reading data to OPC server. It is notified for the OPC server to complete asynchronous reading.
Async Write	It demands asynchronous writing data to OPC server. It is notified for the OPC server to complete asynchronous writing.
DisConnect	Disconnects with the OPC server.

When OPC server finishes within connecting, the message box "Server Shutdown" will appears.

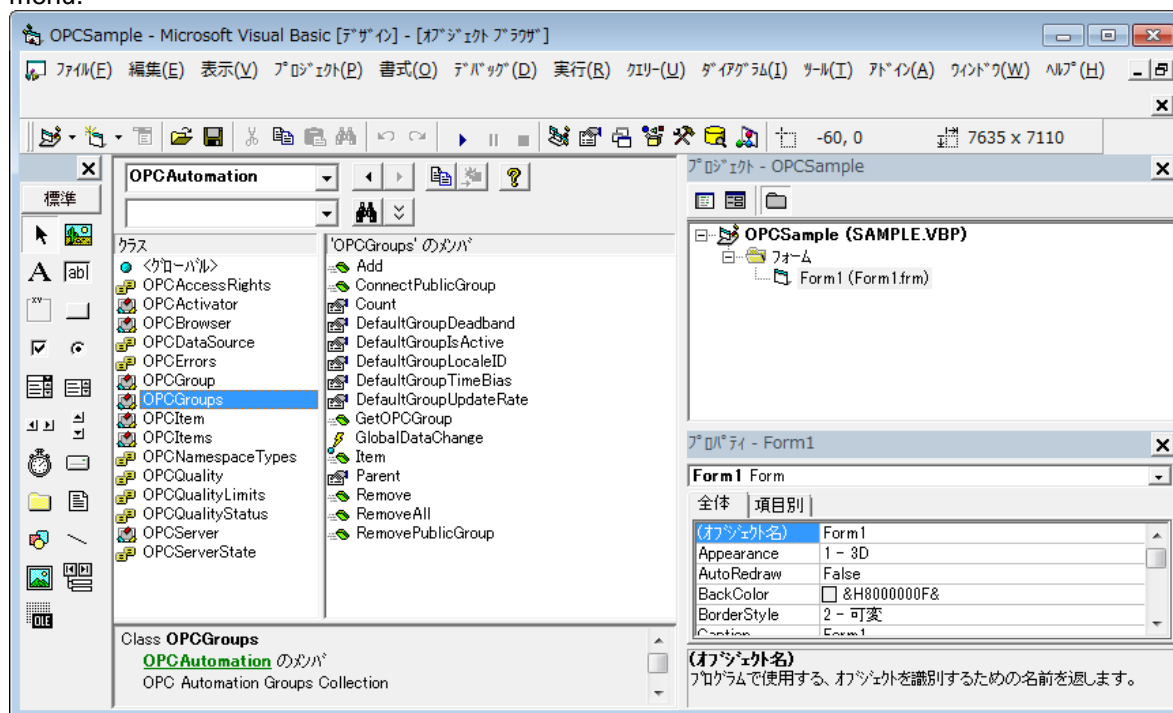
1.4 Visual Basic 6.0 Sample (for Automation Interface)

1.4.2 Setting of Development Environment

To make modifications to Visual Basic 6.0, select "Reference" from "Project" menu. And then, check "OPC Automation 2.02" (file name: OPCDAAUTO.DLL).



The properties of the OPC server can be viewed from the object browser ("F2" key) located in the view menu.



1.4 Visual Basic 6.0 Sample (for Automation Interface)

1.4.3 Program Example

Here are general descriptions that uses OPC Automation Interface with Visual Basic 6.0.

Refer to the attached sample programs and the specifications of OPCDA2.0 for details about programs and OPC Automation Interface.

OPC Automation Interface doesn't correspond to OPCDA3.0.

❖ DEFINED PART

Define objects to use in OPC server.

You should appoint "WithEvents" in objects including event (OPCServer / OPCGroups / OPCGroup etc.).

Then you can get event information.

<i>Dim WithEvents OPCMyServer As OPCServer</i>	<i>' OPCServer object</i>
<i>Dim WithEvents OPCMyGroups As OPCGroups</i>	<i>' OPCGroups correction</i>
<i>Dim WithEvents OPCMyGroup As OPCGroup</i>	<i>' OPCGroup object</i>
<i>Dim OPCMyItems As OPCItems</i>	<i>' OPCItems correction</i>
<i>Dim OPCMyItem As OPCItem</i>	<i>' OPCItem object</i>

❖ CONNECT WITH SERVER

Create a server object and execute "Connect" method. Set Prog.ID of the OPC sever. This example shows that it connects to MELSEC OPC Server.

<i>Set OPCMyServer = New OPCServer</i>	<i>' Make a OPCServer</i>
<i>OPCMyServer.Connect "Takebishi.Dxp.5", ""</i>	<i>' Connect with OPC server</i>

❖ DISCONNECT WITH THE SERVER

For disconnection with the OPC server, execute "Remove" method of OPCGroups, and then execute "Disconnect" method. Don't forget to delete objects after using.

<i>OPCMyGroups.Remove OPCMyGroup.ServerHandle</i>	
<i>Set OPCMyItems = Nothing</i>	<i>' Delete OPCItems correction</i>
<i>Set OPCMyItem = Nothing</i>	<i>' Delete OPCItem objects</i>
<i>Set OPCMyGroups = Nothing</i>	<i>' Delete OPCGroups correction</i>
<i>Set OPCMyGroup = Nothing</i>	<i>' Delete OPCGroup objects</i>
<i>OPCMyServer.Disconnect</i>	<i>' Disconnect with OPC server</i>
<i>Set OPCMyServer = Nothing</i>	<i>' Delete OPCServer object</i>

❖ CREATE A GROUP

Get "OPCGroups" property of server object for adding a group. This example shows that group's name is "Group1".

<i>Set OPCMyGroups = OPCMyServer.OPCGroups</i>	<i>' Make a OPCGroups</i>
<i>Set OPCMyGroup = OPCMyGroups.Add("Group1")</i>	<i>' Add a OPCGroup</i>

❖ ADD ITEMS

Then, add items. This example shows that an item "Device1.D0" is added in OPC server.

When you define an item name ("OPCItemID"), divide the device, group and tag with period (.).

You should define a unique value to "ClientHandle" for client.

"ItemServerHandles" and "Errors" are set in OPC server.

You can register plural items together.

<i>Dim ItemServerHandles() As Long</i>	<i>' Item handle</i>
<i>Dim ClientHandles(1) As Long</i>	<i>' Client handle</i>
<i>Dim OPCItemIDs(1) As String</i>	<i>' Item ID (PLC device name)</i>
<i>Dim Errors() As Long</i>	<i>' Item error</i>
<i>Set OPCMyItems = OPCMyGroup.OPCItems</i>	<i>' Item correction</i>
<i>OPCItemIDs(1) = "Device1.D0"</i>	
<i>ClientHandle(1) = 1</i>	
<i>OPCMyItems.AddItem 1, OPCItemIDs, ClientHandles, ItemServerHandles, Errors</i>	
<i>If Errors(1) <> 0 Then</i>	<i>' If there are any error in item registration</i>

1.4 Visual Basic 6.0 Sample (for Automation Interface)

```
MsgBox "Error"  
End If
```

❖ SET FOR AUTOMATIC READING OUT

When setting "IsActive" property of OPCGroup to TRUE, OPC server reads out registered items. When setting "IsSubscription" property to TRUE, "DataChange" event is called back with change of read out data. Only the items that occurred in those groups are called back. Refer to "ClientHandle" when you would like to know items change or not.

```
OPCMyGroup.IsActive = True  
OPCMyGroup.IsSubscribed = True  
  
Private Sub OPCMyGroup_DataChange(ByVal TransactionID As Long, ByVal NumItems As Long, _  
ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date)  
  
    ' The number of informed items is set to "NumItems".  
    ' Handle of the changed items is set to "ClientHandles".  
  
End Sub
```

❖ SYNC READ

This example shows that all items that entered in OPCItems are read out with device. You can get quality and timestamp from OPC server.

```
Dim anItem As OPCItem  
For Each anItem In OPCMyGroup.OPCItems  
    anItem.Read OPCDevice  
    Debug.Print anItem.Value, anItem.TimeStamp, anItem.Quality  
    Set anItem = Nothing  
Next anItem  
  
Dim Values() As Variant  
Dim Errors() As Long  
Dim Qualities As Variant  
Dim TimeStamps As Variant  
Dim i As Integer  
Dim ItemServerHandles(ItemMax) As Long      ' Server Handle  
For i = 1 To ItemMax  
    ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle ' Get Server Handle  
Next i  
  
Call OPCMygroup.SyncRead(OPCDataSource.OPCDevice, ItemMax, ItemServerHandles, Values, Errors,  
Qualities, TimeStamps)  
  
For i = 1 To ItemMax  
    Debug.Print Values(i), Qualities(i), TimeStamps(i), Qualities(i)  
Next i
```

❖ SYNC WRITE

This example shows that value 123 is wrote in all items registered in OPCItems.

```
Dim Values(ItemMax) As Variant  
Dim Errors() As Long  
Dim i As Integer  
Dim ItemServerHandles(ItemMax) As Long      ' Server Handle  
For i = 1 To ItemMax  
    ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle ' Get Server Handle  
    Values(i) = 123  
Next i
```

1.4 Visual Basic 6.0 Sample (for Automation Interface)

Call OPCMygroup.SyncWrite(ItemMax, ItemServerHandles, Values, Errors)

❖ ASYNCREAD

This example shows that all items that entered in OPCItems are read out with device. You can get quality and timestamp from OPC server. When the client calls AsyncRead, this request is received by the server. AsyncReadComplete event is called back from the server as the result. Refer to "ClientHandle" when you would like to know items change or not.

```
Private Sub ASYNCREAD_Button_Click()  
    Dim i As Integer  
    Dim TransactionID As Long  
    Dim CancelID As Long  
    Dim Values() As Variant  
    Dim Errors() As Long  
  
    TransactionID = 100 ' set 100(an arbitrary number) in transactionID  
  
    Dim ItemServerHandles(ItemMax) As Long      ' Server Handle  
    For i = 1 To ItemMax  
        ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle ' Get Server Handle  
    Next i  
  
    Call OPCMygroup.AsyncRead(ItemMax, ItemServerHandles, Errors, TransactionID, CancelID)  
End Sub  
  
Private Sub OPCMygroup_AsyncReadComplete(ByVal TransactionID As Long, ByVal NumItems As Long,  
ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date, Errors() As  
Long)  
    If TransactionID = 100 Then  
        MsgBox "Async Read Complete"  
    End If  
  
    ' The number of items is set to "NumItems".  
    ' Handle of items is set to "ClientHandles".  
End Sub
```

❖ ASYNCSWRITE

This example shows that value 123 is wrote in all items registered in OPCItems. When the client calls AsyncWrite method, this request is received by the server. AsyncWriteComplete event is called back from the server as the result. Refer to "ClientHandle" when you would like to know items change or not.

```
Private Sub ASYNCSWRITE_Button_Click()  
    Dim i As Integer  
    Dim TransactionID As Long  
    Dim CancelID As Long  
    Dim Values(ItemMax) As Variant  
    Dim Errors() As Long  
  
    TransactionID = 101 ' set 101(an arbitrary number) in transactionID  
  
    Dim ItemServerHandles(ItemMax) As Long      ' Server Handle  
    For i = 1 To ItemMax  
        ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle ' Get Server Handle  
        Values(i) = 123  
    Next i  
  
    Call OPCMygroup.AsyncWrite(ItemMax, ItemServerHandles, Values, Errors, TransactionID, CancelID)  
End Sub  
  
Private Sub OPCMygroup_AsyncWriteComplete(ByVal TransactionID As Long, ByVal NumItems As Long,  
ClientHandles() As Long, Errors() As Long)
```

1.4 Visual Basic 6.0 Sample (for Automation Interface)

```
If TransactionID = 101 Then  
    MsgBox "Async Write Complete"  
End If  
  
    ' The number of items is set to "NumItems".  
    ' Handle of items is set to "ClientHandles".  
End Sub
```

1.5 Visual Basic .NET Sample (for OPC Automation Interface)

1.5 Visual Basic .NET Sample (for OPC Automation Interface)

This sample program is to make the OPC client that uses OPC Automation Interface in the development environment of Visual Basic .NET.

1.5.1 Operation

The sample program for Visual Basic .NET is stored in the "\\OPC Sample\\DotNetDAautoSample" folder at the installation destination. When you execute "DAAutoDotNET.exe" stored in the "bin" folder, the following screen is displayed.

Item Name	Value	Date/Time	Quality
Device1.D1			
Device1.D2			
Device1.D3			
Device1.D4			
Device1.D5			
Device1.D6			
Device1.D7			
Device1.D8			

The function of each part is as follows.

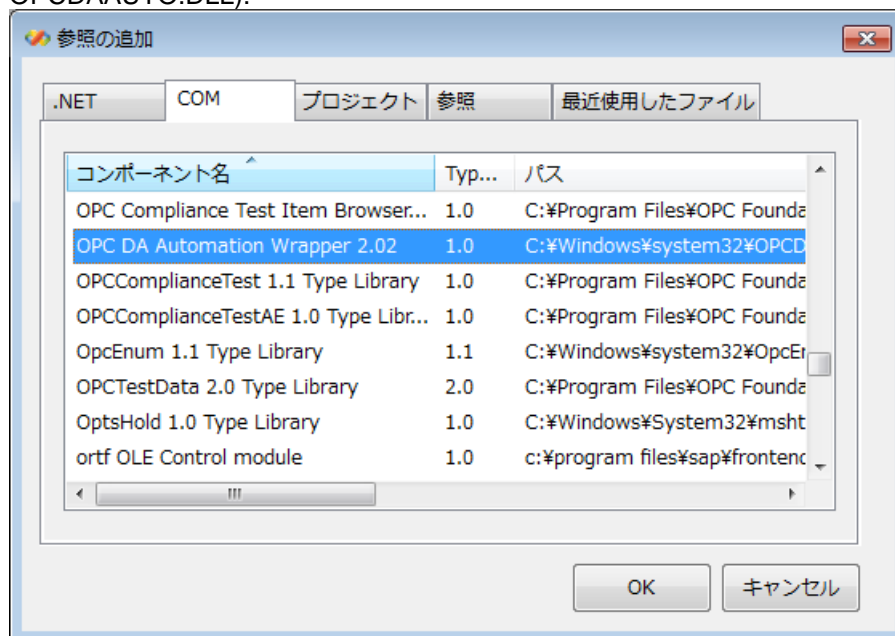
Function	Contents
Node Name	Input a target node name (machine name or IP address).
OPCServer Name	Select a OPC server name to connect. Prog.ID of DeviceXPlorer OPC Server is "Takebishi.Dxp". Other OPC Server's Prog.ID is written at 1.1
Update Rate	Input a cycle reading of the PLC data. The unit is "msec".
Item Name	Input a pre-defined device name and item name. (format is "device name.item name".) The default settings are between "Device1.D0" and "Device1.D7".
Connect	Connects with the OPC server. And the button will change to "Disconnect".
Advise	The group and data report become valid. The data in OPC server is read out every "Update Rate" cycle. Changing report will be sent to OPC client. And when you execute "AsyncRead" and "AsyncWrite" method, "Advise" must become valid.
Sync Read	Read and the value, the quality, and the timestamp of the tag will be retrieved from the OPC server and displayed.
Sync Write	Writes value with "Value" to the OPC server.
Async Read	It demands asynchronous reading data to OPC server. It is notified for the OPC server to complete asynchronous reading.
Async Write	It demands asynchronous writing data to OPC server. It is notified for the OPC server to complete asynchronous writing.
DisConnect	Disconnects with the OPC server.

1.5 Visual Basic .NET Sample (for OPC Automation Interface)

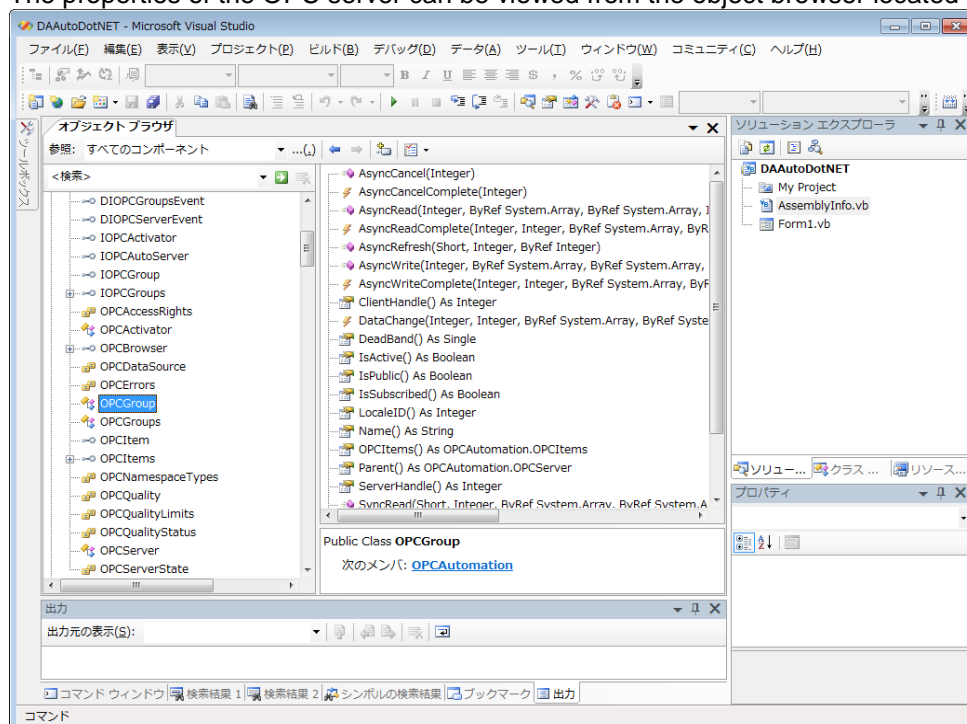
1.5.2 Setting of Development Environment

To make modifications to Visual Basic .NET, select "Reference" from "Project" menu.

And then, select "COM" tab and check "OPC DA Automation Wrapper 2.02" (file name: OPCDAAUTO.DLL).



The properties of the OPC server can be viewed from the object browser located in the view menu.

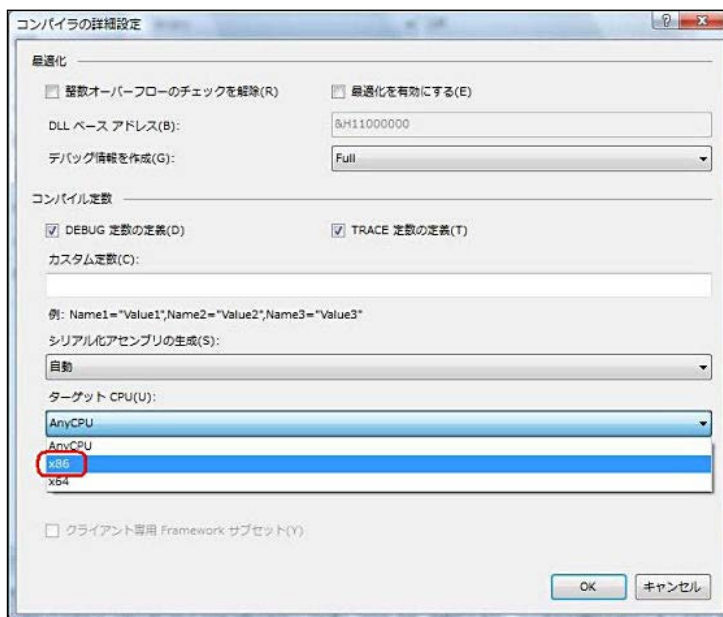


1.5 Visual Basic .NET Sample (for OPC Automation Interface)

1.5.3 Note in building sample on Windows for x64

Automation Wrapper DLL is built for 32 bits. When you build the sample of the OPC client by VisualStudio on 64 bits Windows, please select x86 of the target CPU setting. If you selected AnyCPU or x64 and built, an exception error (0x80040154) of class non-registration of COM may occur at the runtime.

The target CPU setting can be found by opening 'Project Properties' and selecting the 'Compile' tab and selecting 'Detail Compile Option'.



1.5.4 Program Example

Here are general descriptions that uses OPC Automation Interface with Visual Basic .NET.

Refer to the attached sample programs and the specifications of OPCDA2.0 for details about programs and OPC Automation Interface.

OPC Automation Interface doesn't correspond to OPCDA3.0.

❖ DEFINED PART

Define objects to use in OPC server.

You should appoint "WithEvents" in objects including event (OPCServer / OPCGroups / OPCGroup etc.).

Then you can get event information.

```
Dim WithEvents OPCMyServer As OPCAutomation.OPCServer      ' OPCServer object
Dim WithEvents OPCMyGroups As OPCAutomation.OPCGroups      ' OPCGroups correction
Dim WithEvents OPCMyGroup As OPCAutomation.OPCGroup        ' OPCGroup object
Dim OPCMyItems As OPCAutomation.OPCItems                  ' OPCItems correction
Dim OPCMyItem As OPCAutomation.OPCItem                     ' OPCItem object
```

❖ CONNECT WITH SERVER

Create a server object and execute "Connect" method. Set Prog.ID of the OPC sever. This example shows that it connects to MELSEC OPC Server.

```
Set OPCMyServer = New OPCAutomation.OPCServer      ' Make a OPCServer
OPCMyServer.Connect("Takebishi.Dxp.5", "")         ' Connect with OPC server
```

❖ CREATE A GROUP

Get "OPCGroups" property of server object for adding a group. This example shows that group's name is "Group1".

```
Set OPCMyGroups = OPCMyServer.OPCGroups           ' Make a OPCGroups
Set OPCMyGroup = OPCMyGroups.Add("Group1")         ' Add a OPCGroup
```

❖ ADD ITEMS

Then, add items. This example shows that an item "Device1.D0" is added in OPC server.

When you define an item name ("OPCItemID"), divide the device, group and tag with period (.).

You should define a unique value to "ClientHandle" for client.

"ItemServerHandles" and "Errors" are set in OPC server.

You can register plural items together.

```
Dim ItemServerHandles() As Long                    ' Item handle
Dim ClientHandles(1) As Long                      ' Client handle
Dim OPCItemIDs(1) As String                       ' Item ID (PLC device name)
Dim Errors() As Long                              ' Item error
Set OPCMyItems = OPCMyGroup.OPCItems              ' Item correction
OPCItemIDs(1) = "Device1.D0"
ClientHandle(1) = 1
OPCMyItems.AddItems(1, OPCItemIDs, ClientHandles, ItemServerHandles, Errors)
If Errors(1) <> 0 Then                              ' If there are any error in item registration
    MsgBox "Error"
End If
```

❖ SET FOR AUTOMATIC READING OUT

When setting “IsActive” property of OPCGroup to TRUE, OPC server reads out registered items. When setting “IsSubscription” property to TRUE, “DataChange” event is called back with change of read out data. Only the items that occurred in those groups are called back. Refer to “ClientHandle” when you would like to know items change or not.

```
OPCMyGroup.IsActive = True
OPCMyGroup.IsSubscribed = True

Private Sub OPCMyGroup_DataChange(ByVal TransactionID As Integer, _
    ByVal NumItems As Integer, _
    ByRef ClientHandles As System.Array, _
    ByRef ItemValues As System.Array, _
    ByRef Qualities As System.Array, _
    ByRef TimeStamps As System.Array) Handles OPCMyGroup.DataChange

    ' The number of informed items is set to "NumItems".
    ' Handle of the changed items is set to "ClientHandles".

End Sub
```

❖ READ OUT

This example shows that all items that entered in OPCItems are read out with device. OPCGroup can also read out with “SyncRead” and “AsyncRead” method. These methods enable OPCGroup to read out plural items. You can get quality and timestamp from OPC server.

```
Dim anItem As OPCAutomation.OPCItem
For Each anItem In OPCMyGroup.OPCItems
    anItem.Read(OPCAutomation.OPCDataSource.OPCDevice)
    Set anItem = Nothing
Next anItem
```

❖ WRITE IN

This example shows that value 123 is wrote in all items registered in OPCItems. OPCGroup can write in with “SyncWrite” and “AsyncWrite” method. These methods enable OPCGroup to write in plural items.

```
Dim anItem As OPCAutomation.OPCItem
For Each anItem In OPCMyGroup.OPCItems
    anItem.Write(123)
    Set anItem = Nothing
Next anItem
```

❖ DISCONNECT WITH THE SERVER

For disconnection with the OPC server, execute “Remove” method of OPCGroups, and then execute “Disconnect” method. Don’t forget to delete objects after using.

```
OPCMyGroups.Remove(OPCMyGroup.ServerHandle)
Set OPCMyItems = Nothing           ' Delete OPCItems correction
Set OPCMyItem = Nothing            ' Delete OPCItem objects
Set OPCMyGroups = Nothing          ' Delete OPCGroups correction
Set OPCMyGroup = Nothing           ' Delete OPCGroup objects
OPCMyServer.Disconnect()           ' Disconnect with OPC server
Set OPCMyServer = Nothing          ' Delete OPCServer object
```


1.6 Visual Basic .NET Sample (for OPC Custom Interface)

1.6 Visual Basic .NET Sample (for OPC Custom Interface)

This sample program is to make the OPC client that uses OPC Custom Interface in the development environment of Visual Basic .NET.

1.6.1 Operation

The sample program for Visual Basic .NET is stored in the "\OPC Sample\DotNetRcwSample" folder at the installation destination. When you execute "SampleDotNET.exe" stored in the "bin" folder, the following screen is displayed.

The screenshot shows a Windows application window titled "Form1". It contains the following elements:

- Node Name:** Text box containing "localhost".
- Server Name:** Text box containing "Takebishi.Dxp.5".
- Update Rate:** Spin box set to "1000".
- OPCDA Version:** Radio buttons for "2.0" and "3.0" (selected).
- Active:** Check box (unchecked).
- ItemActive:** Check box (unchecked).
- Buttons:** "MaxAge ON", "Read", "Write", "Advise", "Async Read", and "Async Write".
- Table:** A table with 4 columns: "Item Name", "Value", "Date/Time", and "Quality". It lists items from "Device1.D0" to "Device1.D7".
- Footer:** "DotNetRcwSample ver3.0" in the bottom right corner.

The function of each part is as follows.

Function	Contents
Node Name	Input a target node name (machine name or IP address).
OPCServer Name	Select a OPC server name to connect. Prog.ID of DeviceXPlorer OPC Server is "Takebishi.Dxp". Other OPC Server's Prog.ID is written at 1.1
Update Rate	Input a cycle reading of the PLC data. The unit is "msec".
OPCDA Version	Select OPCDA version to connect.
Active	Select Group active state.
Item Active	Select Item active state.
Item Name	Input a pre-defined device name and item name. (format is "device name.item name".) The default settings are between "Device1.D0" and "Device1.D7".
Connect	Connects with the OPC server. And the button will change to "Disconnect".
Advise	The group and data report become valid. The data in OPC server is read out every "Update Rate" cycle. Changing report will be sent to OPC client. And when you execute "AsyncRead" and "AsyncWrite" method, "Advise" must become valid.
MaxAge On	It reads the value by specifying "Old" of data every second. In this sample, the "MaxAge" setting time (specified time of "Old") is 10000msec. When the data maintained in the OPC Server is older than 10000msec, the OPC Server acquires the value from device (PLC). And when the data is newer than 10000msec, the OPC Server returns the value of cache (memory of the OPC

1.6 Visual Basic .NET Sample (for OPC Custom Interface)

	server).
Read	Read and the value, the quality, and the timestamp of the tag will be retrieved from the OPC server and displayed.
Write	Writes value with "Value" to the OPC server.
Async Read	It demands asynchronous reading data to OPC server. It is notified for the OPC server to complete asynchronous reading.
Async Write	It demands asynchronous writing data to OPC server. It is notified for the OPC server to complete asynchronous writing.
DisConnect	Disconnects with the OPC server.

1.6.2 Setting of Development Environment

OPC server works based on COM(Component Object Model). Therefore when you access OPC server from .NET environment, the conversion from COM to .NET is needed.

Please install OPC Core Components SDK beforehand.

OPC Core Components SDK can be downloaded from the homepage of OPC Foundation.

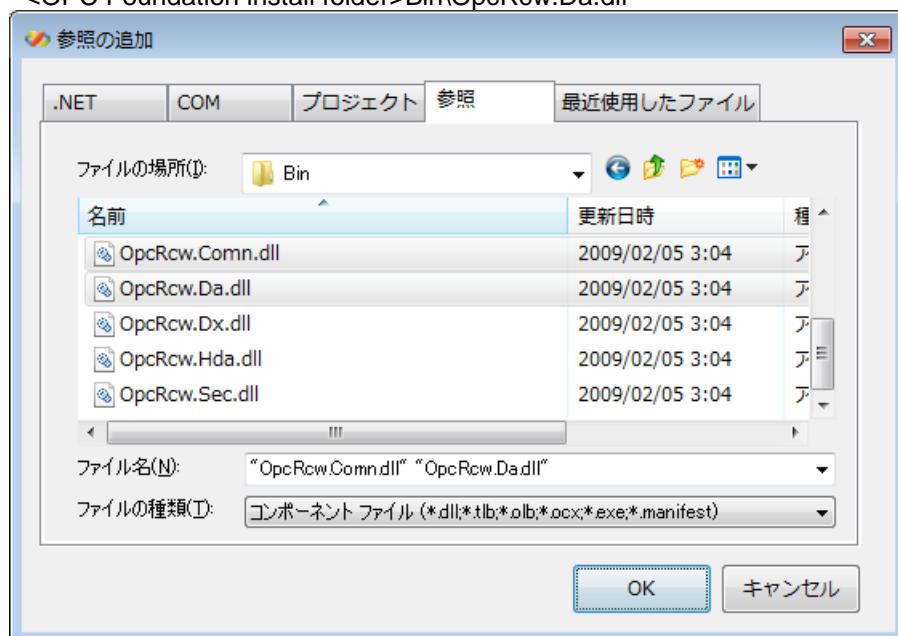
<http://www.opcfoundation.org/>

To make modifications to Visual Basic .NET, select "Reference" from "Project" menu.

And then, click "Reference" button and select the following files.

- <OPC Foundation install folder>Bin\OpcRcw.Comn.dll

- <OPC Foundation install folder>Bin\OpcRcw.Da.dll

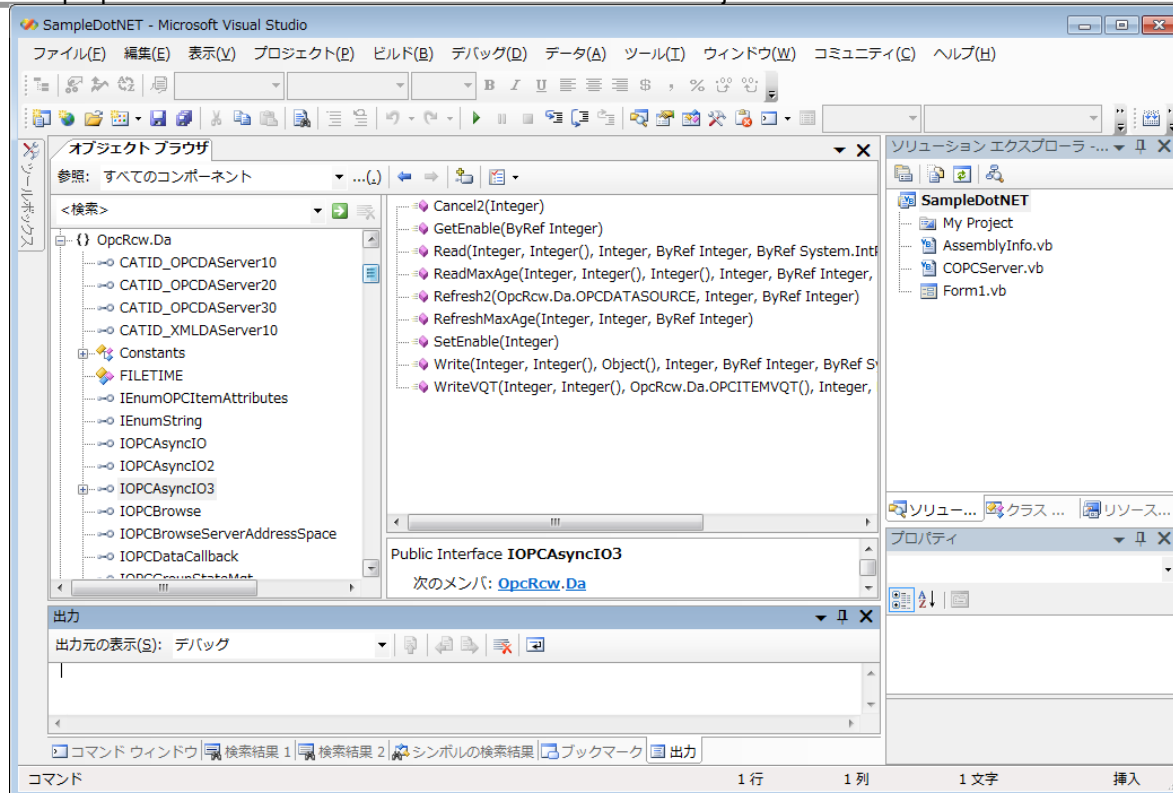


The following components become valid.

- OpcRcw.Comn
- OpcRcw.Da

1.6 Visual Basic .NET Sample (for OPC Custom Interface)

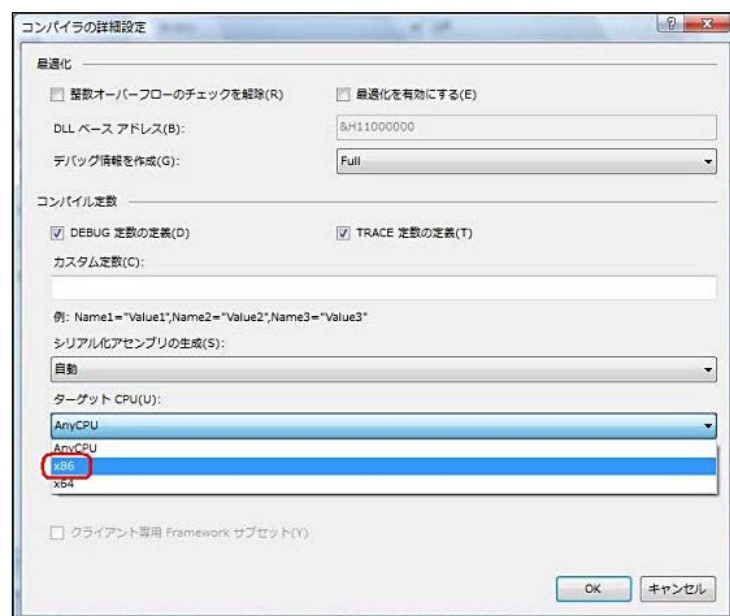
The properties of the OPC server can be viewed from the object browser located in the view menu.



1.6.3 Note in building sample on Windows for x64

RCW DLL is built for 32 bits. When you build the sample of the OPC client by VisualStudio on 64 bits Windows, please select x86 of the target CPU setting. If you selected AnyCPU or x64 and built, application can not work normally at the runtime.

The target CPU setting can be found by opening 'Project Properties' and selecting the 'Compile' tab and selecting 'Detail Compile Option'.



1.6 Visual Basic .NET Sample (for OPC Custom Interface)

1.6.4 Program Example

Here are general descriptions that uses OPC Custom Interface with Visual Basic .NET. It explains the COPCServer class that uses it by the sample program in this paragraph.

Refer to the attached sample programs and the specifications of OPCDA3.0 for details about programs and OPC Automation Interface.

❖ OUTLINE

COPCServer is the user class to communicate with OPC server. And to acquire the event from the OPC server, IOPCDataCallback interface has been implemented.

❖ DEFINED PART

Define objects to use in OPC server.

```
Private m_OPCTServer As IOPCTServer           ' OPCTServer object
Private m_OPCTGroup As IOPCTGroupStateMgt2     ' OPCTGroup object
Private m_OPCTItem As IOPCTItemMgt            ' OPCTItem object
Private m_OPCTConnPointCntnr As IConnectionPointContainer ' ConnectionPointContainer object
Private m_OPCTConnPoint As IConnectionPoint    ' ConnectionPoint object
```

❖ CONNECT WITH SERVER

Create a server object and connect OPC server. Set Prog.ID of the OPC sever. This example shows that it connects to MELSEC OPC Server.

```
m_OPCTServer = CreateObject("Takebishi.Dxp.5")
```

❖ CREATE A GROUP

Execute "AddGroup" method of server object for adding a group.

Allocate the connection point in COPCCallBack class, then the callback from the server become valid.

```
guidGroupStateMgt = Marshal.GenerateGuidForType(GetType(IOPCTGroupStateMgt2))
m_OPCTServer.AddGroup("Group1", _           ' Group name
    True, _                                ' True:Active, False:Not active
    1000, _                                ' UpdateRate
    1234, _                                ' Group ID for client
    ptrTimeBias, _                         ' Pointer to TimeBias
    ptrDeadBand, _                         ' Pointer to DeadBand
    0, _                                   ' Language ID
    m_iServerGroup, _                     ' Group ID for server
    iRevisedUpdateRate, _                 ' Pointer to UpdateRate returned by server
    guidGroupStateMgt, _                  ' GUID of IOPCTGroupStateMgt
    m_OPCTGroup)                          ' OPCTGroup object
```

```
m_OPCTConnPointCntnr = CType(m_OPCTGroup, IConnectionPointContainer)
guidDataCallback = Marshal.GenerateGuidForType(GetType(IOPCTDataCallback))
m_OPCTConnPointCntnr.FindConnectionPoint(guidDataCallback, m_OPCTConnPoint)
```

❖ ADD ITEMS

Then, add items. This example shows that an item "Device1.D0" is added in OPC server.

In the OPCITEMDEF structure, Store information necessary for registration.

Delimiter at period (.) and specify the device group tag set with the OPC server for "szItemID".

Set active of each item to "bActive". Set the unique value for client to "hClient".

The unique value for server is set to "hServer".

"ppResult" and "ppErrors" are set with the OPC server. "ppResult" is a pointer of the OPCITEMRESULT structure. Delete the memory of "ppResult" and "ppErrors" at the end.

You can register plural items together.

```
itemDef(1).szItemID = "Device1.D0"           ' Item name
itemDef(1).bActive = True                    ' True:Active, False:Not active
itemDef(1).hClient = 1                      ' Item ID for client
m_OPCCItem = CType(m_OPCCGroup, IOPCCItemMgt)
m_OPCCItem.AddItem(1, itemDef, ppResult, ppErrors)
itemResult = CType(Marshal.PtrToStructure(ppResult, GetType(OPCITEMRESULT)), OPCITEMRESULT)
ServerHd = itemResult.hServer
Marshal.FreeCoTaskMem(ppResult)
Marshal.FreeCoTaskMem(ppErrors)
```

❖ MAKE CALLBACK VALID

The callback become valid by "Advise" method of ConnectionPoint object.

When you execute automatic reading, asynchronous reading, and asynchronous writing, the callback must become valid.

```
m_OPCCConnPoint.Advise(Me, m_iCallBackConnection)
```

❖ READ OUT

This example shows that the OPCGroup object is cast to OPCSyncIO2 interface and execute synchronous reading 1 item from device. The quality and the timestamp can be acquired from the OPC server besides the value. Delete the memory of "ppItemVal" and "ppErrors" at the end.

Besides this, the OPCGroup object enables to be cast to OPCAsyncIO3 interface and execute asynchronous reading.

```
OPCSyncIO = CType(m_OPCCGroup, IOPCSyncIO2)
OPCSyncIO.Read(OPCDATASOURCE.OPC_DS_DEVICE, 1, ServerHd, ppItemVal, ppErrors)
' Synchronous reading
ItemState = CType(Marshal.PtrToStructure(ppItemVal, GetType(OPCITEMSTATE)), OPCITEMSTATE)
vValue = ItemState.vDataValue                ' Acquire the value
fTimeStamp = Date.FromFileTime(ItemState.ftTimeStamp.dwHighDateTime * 2 ^ 32 + _
                                     ItemState.ftTimeStamp.dwLowDateTime) ' Acquire the time
wQuality = ItemState.wQuality                ' Acquire the quality
Marshal.FreeCoTaskMem(ppItemVal)
Marshal.FreeCoTaskMem(ppErrors)
```

❖ WRITE IN

This example shows that the OPCGroup object is cast to OPCSyncIO2 interface and execute synchronous writing 1 item. Delete the memory of "ppErrors" at the end.

Besides this, the OPCGroup object enables to be cast to OPCAsyncIO3 interface and execute asynchronous writing.

```
OPCSyncIO = CType(m_OPCCGroup, IOPCSyncIO2)
vValue = 12345
OPCSyncIO.Write(1, ServerHd, vValue, ppErrors) ' Synchronous writing
Marshal.FreeCoTaskMem(ppErrors)
```

❖ NOTIFICATION OF DATA CHANGE

When the value of the device changes in the PLC, this event is notified by OPC server. Refer to "phClientItems" to identify which item the change occurred.

```
Sub onDataChange(ByVal dwTransid As Integer, _
    ByVal hGroup As Integer, _
    ByVal hrMasterquality As Integer, _
    ByVal hrMastererror As Integer, _
    ByVal dwCount As Integer, _           ' Number of items where change occurs
    ByVal phClientItems() As Integer, _   ' Item ID for client
    ByVal pvValues() As Object, _        ' Value
    ByVal pwQualities() As Short, _      ' Quality
    ByVal pftTimeStamps() As OpcRcw.Da.FILETIME, _ ' Time
    ByVal pErrors() As Integer) _        ' Error
    Implements IOPCDataCallback.OnDataChange

    RaiseEvent DataChange(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, _
        pwQualities, pErrors)           ' Generate event
End Sub
```

❖ NOTIFICATION OF ASYNCHRONOUS READING COMPLETION

When asynchronous reading is completed, this event is notified by OPC server. Refer to "phClientItems" to identify the read item.

```
Sub OnReadComplete(ByVal dwTransid As Integer, _
    ByVal hGroup As Integer, _
    ByVal hrMasterquality As Integer, _
    ByVal hrMastererror As Integer, _
    ByVal dwCount As Integer, _           ' Number of items that reading is completed
    ByVal phClientItems() As Integer, _   ' Item ID for client
    ByVal pvValues() As Object, _        ' Value
    ByVal pwQualities() As Short, _      ' Quality
    ByVal pftTimeStamps() As OpcRcw.Da.FILETIME, _ ' Time
    ByVal pErrors() As Integer) _        ' Error
    Implements IOPCDataCallback.OnReadComplete

    RaiseEvent ReadComplete(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, _
        pwQualities, pErrors)           ' Generate event
End Sub
```

❖ NOTIFICATION OF ASYNCHRONOUS WRITING COMPLETION

When asynchronous writing is completed, this event is notified by OPC server. Refer to "phClientItems" to identify the write item.

```
Sub OnWriteComplete(ByVal dwTransid As Integer, _
    ByVal hGroup As Integer, _
    ByVal hrMastererr As Integer, _
    ByVal dwCount As Integer, _           ' Number of items that writing is completed
    ByVal phClientItems() As Integer, _   ' Item ID for client
    ByVal pErrors() As Integer) _        ' Error
    Implements IOPCDataCallback.OnWriteComplete

    RaiseEvent WriteComplete(dwTransid, dwCount, phClientItems, pErrors)
    ' Generate event
End Sub
```

❖ NOTIFICATION OF ASYNCHRONOUS PROCESS CANCELLATION

When asynchronous process is canceled, this event is notified by OPC server.

```
Sub OnCancelComplete(ByVal dwTransid As Integer, _  
                    ByVal hGroup As Integer) _  
    Implements IOPCDataCallback.OnCancelComplete  
  
    RaiseEvent CancelComplete(dwTransid)           ' Generate event  
End Sub
```

❖ DISCONNECT WITH THE SERVER

For disconnection with the OPC server, execute "RemoveGroup" method of OPCServer. Don't forget to delete objects after using.

```
If m_iServerGroup <> 0 Then  
    m_OPCTServer.RemoveGroup(m_iServerGroup, False)  
    ret = Marshal.ReleaseComObject(m_OPCTGroup)  
    m_iServerGroup = 0  
End If  
  
ret = Marshal.ReleaseComObject(m_OPCTServer)  
m_OPCTGroup = Nothing  
m_OPCTServer = Nothing
```

1.7 Visual C# Sample (for OPC Custom Interface)

1.7 Visual C# Sample (for OPC Custom Interface)

This sample program is to make the OPC client that uses OPC Custom Interface in the development environment of Visual C#.

1.7.1 Operation

The sample program for Visual C# is stored in the "\\OPC Sample\\VCDotNetRcwSample" folder at the installation destination. When you execute "VCDotNetRcwSample.exe" stored in the "\\bin\\Release" folder, the following screen is displayed.

Item Name	Value	Date/Time	Quality
Device1.D0			
Device1.D1			
Device1.D2			
Device1.D3			
Device1.D4			
Device1.D5			
Device1.D6			
Device1.D7			
Device1.D8			
Device1.D9			

The function of each part is as follows.

Function	Contents
Node Name	Input a target node name (machine name or IP address).
OPCServer Name	Select a OPC server name to connect. Prog.ID of DeviceXPlorer OPC Server is "Takebishi.Dxp". Other OPC Server's Prog.ID is written at 1.1
Update Rate	Input a cycle reading of the PLC data. The unit is "msec".
OPCDA Version	Select OPCDA version to connect.
Group Active	Select Group active state.
Item Name	Input a pre-defined device name and item name. (format is "device name.item name".) The default settings are between "Device1.D0" and "Device1.D9".
Connect	Connects with the OPC server. And the button will change to "Disconnect".
Advise	The group and data report become valid. The data in OPC server is read out every "Update Rate" cycle. Changing report will be sent to OPC client. And when you execute "AsyncRead" and "AsyncWrite" method, "Advise" must become valid.
MaxAge On	It reads the value by specifying "Old" of data every second. In this sample, the "MaxAge" setting time (specified time of "Old") is 10000msec. When the data maintained in the OPC Server is older than 10000msec, the OPC Server

1.7 Visual C# Sample (for OPC Custom Interface)

	acquires the value from device (PLC). And when the data is newer than 10000msec, the OPC Server returns the value of cache (memory of the OPC server).
Read	Read and the value, the quality, and the timestamp of the tag will be retrieved from the OPC server and displayed.
Write	Writes value with "Value" to the OPC server.
Async Read	It demands asynchronous reading data to OPC server. It is notified for the OPC server to complete asynchronous reading.
Async Write	It demands asynchronous writing data to OPC server. It is notified for the OPC server to complete asynchronous writing.
DisConnect	Disconnects with the OPC server.

1.7.2 Setting of Development Environment

OPC server works based on COM(Component Object Model). Therefore when you access OPC server from .NET environment, the conversion from COM to .NET is needed.

Please install OPC Core Components SDK beforehand.

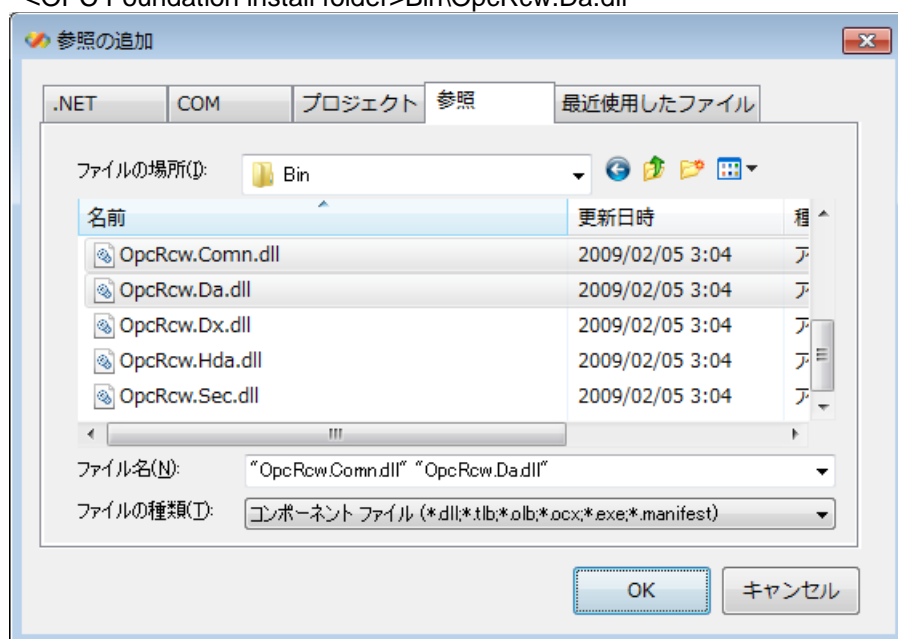
OPC Core Components SDK can be downloaded from the homepage of OPC Foundation.

<http://www.opcfoundation.org/>

To make modifications to Visual C#.NET, select "Reference" from "Project" menu.

And then, click "Reference" button and select the following files.

- <OPC Foundation install folder>Bin\OpcRcw.Comn.dll
- <OPC Foundation install folder>Bin\OpcRcw.Da.dll

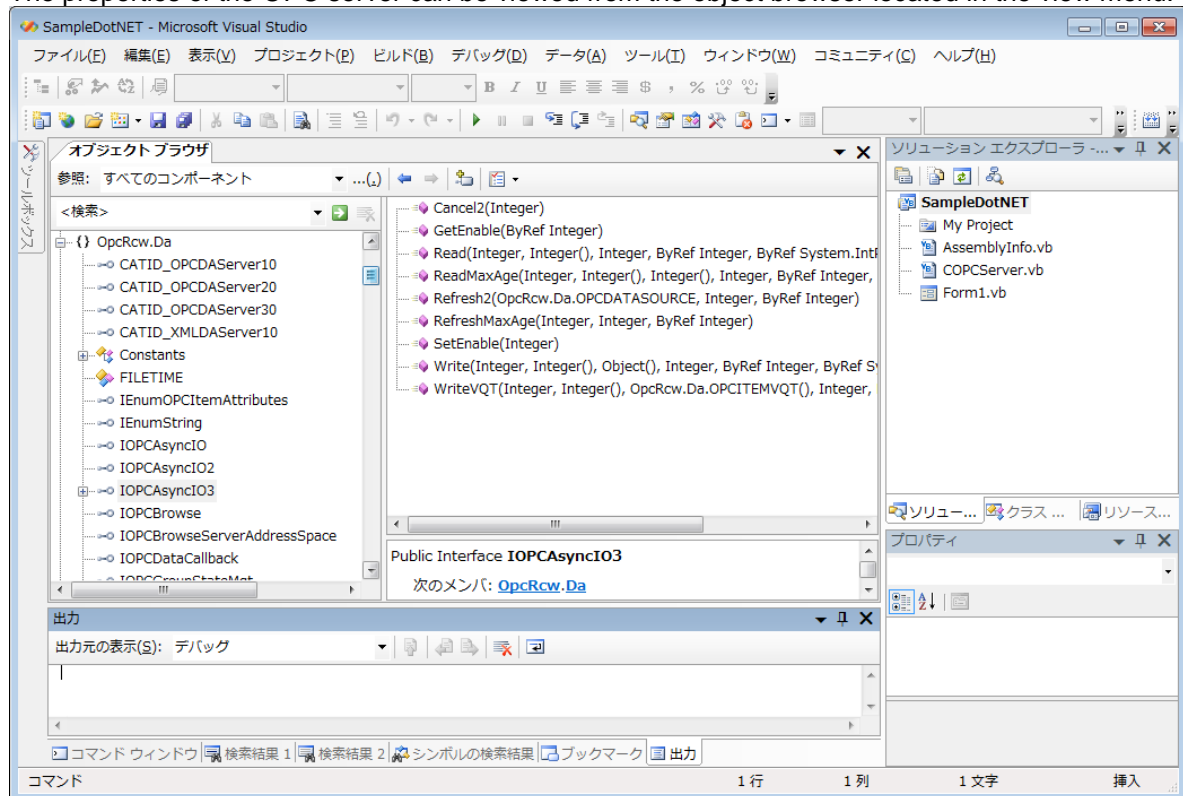


The following components become valid.

- OpcRcw.Comn
- OpcRcw.Da

1.7 Visual C# Sample (for OPC Custom Interface)

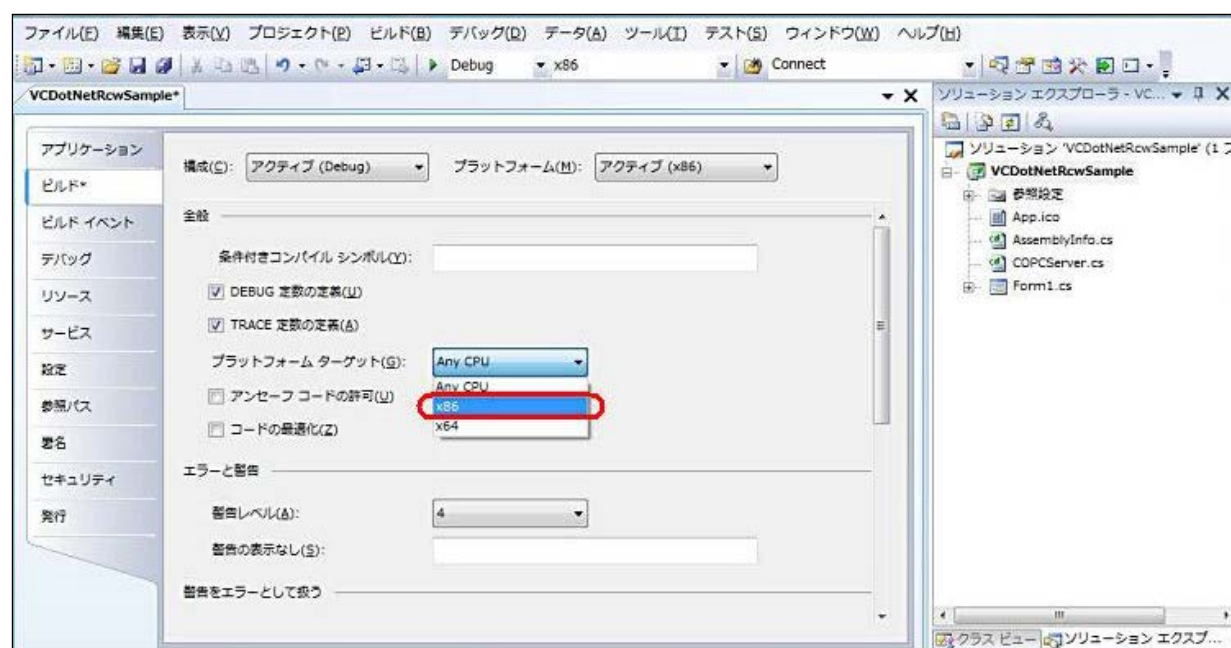
The properties of the OPC server can be viewed from the object browser located in the view menu.



1.7.3 Note in building sample on Windows for x64

RCW DLL is built for 32 bits. When you build the sample of the OPC client by VisualStudio on 64 bits Windows, please select x86 of the platform target setting. If you selected AnyCPU or x64 and built, application can not work normally at the runtime.

The platform target setting can be found by opening 'Project Properties' and selecting the 'Build' tab.



1.7 Visual C# Sample (for OPC Custom Interface)

1.7.4 Program Example

Here are general descriptions that uses OPC Custom Interface with Visual C#. It explains the COPCServer class that uses it by the sample program in this paragraph.

Refer to the attached sample programs and the specifications of OPCDA3.0 for details about programs and OPC Automation Interface.

❖ OUTLINE

COPCServer is the user class to communicate with OPC server. And to acquire the event from the OPC server, IOPCDataCallback interface has been implemented.

❖ DEFINED PART

Define objects to use in OPC server.

```
private IOPCServer          m_OPCTServer;    // OPCTServer object
private IOPCGroupStateMgt2  m_OPCTGroup;    // OPCTGroup object
private IOPCItemMgt         m_OPCTItem;     // OPCTItem object
private IConnectionPointContainer m_OPCTConnPointCntnr;
                                   // ConnectionPointContainer object
private IConnectionPoint    m_OPCTConnPoint; // ConnectionPoint object
```

❖ CONNECT WITH SERVER

First of all, create a list of OPC Servers.

Next, extract a CLSID based on Prog.ID of OPC Server from the list, and create a OPC Server object based on CLSID and connect.

This example shows that it connects to MELSEC OPC Server.

```
IOPCTServerList svrList = (IOPCTServerList)CreateInstance(CLSID_SERVERLIST, null);
                                   // Create the instance of OPCTServerList
Guid clsidList;               // CLSID extracted from OPCTServerList
svrList.CLSIDFromProgID("Takebishi.Dxp.5", out clsidList);
                                   // CLSID is extracted based on ProgID.
m_OPCTServer = (IOPCTServer)CreateInstance(clsidList, null);
                                   // Create the instance of OPCTServer.
```

❖ CREATE A GROUP

Execute "AddGroup" method of server object for adding a group.

Allocate the connection point in COPCCallBack class, then the callback from the server become valid.

```
guidGroupStateMgt = Marshal.GenerateGuidForType(typeof(IOPCTGroupStateMgt2));
m_OPCTServer.AddGroup(sGrpName,
    1, // 1:Active, 0:Not active
    1000, // UpdateRate
    1234, // Group ID for client
    ptrTimeBias, // TimeBias
    ptrDeadBand, // DeadBand
    0, // Language ID
    out m_iServerGroup, // Group ID for server
    out iRevisedUpdateRate, // UpdateRate returned by server
    ref guidGroupStateMgt, // GUID of IOPCTGroupStateMgt
    out group); // OPCTGroup object
m_OPCTGroup = (IOPCTGroupStateMgt2)group;
m_OPCTGroup.SetKeepAlive(iKeepAliveTime, out iKeepAliveTime);

m_OPCTConnPointCntnr = (IConnectionPointContainer)m_OPCTGroup;
guidDataCallback = Marshal.GenerateGuidForType(typeof(IOPCTDataCallback));
m_OPCTConnPointCntnr.FindConnectionPoint(ref guidDataCallback, out m_OPCTConnPoint);
                                   // Allocation of connection point
```

❖ ADD ITEMS

Then, add items. This example shows that an item "Device1.D0" is added in OPC server. In the OPCITEMDEF structure, Store information necessary for registration. Delimit at period (.) and specify the device group tag set with the OPC server for "szItemID". Set active of each item to "bActive". Set the unique value for client to "hClient". The unique value for server is set to "hServer". "ppResult" and "ppErrors" are set with the OPC server. "ppResult" is a pointer of the OPCITEMRESULT structure. Delete the memory of "ppResult" and "ppErrors" at the end. You can register plural items together.

```
itemDef[1].szItemID = "Device1.D0";           // Item name
itemDef[1].bActive = 1;                       // 1:Active, 0:Not active
itemDef[1].hClient = 1;                       // Item ID for client
m_OPCCItem = (IOPCCItemMgt)m_OPCCGroup;
m_OPCCItem.AddItems(1, itemDef, out ppResult, out ppErrors); // Add OPCItem

itemResult = (OPCITEMRESULT)Marshal.PtrToStructure(ppResult, typeof(OPCITEMRESULT));
ServerHd = itemResult.hServer;                // Item ID for server
Marshal.FreeCoTaskMem(ppResult);
Marshal.FreeCoTaskMem(ppErrors);
```

❖ MAKE CALLBACK VALID

The callback become valid by "Advise" method of ConnectionPoint object. When you execute automatic reading, asynchronous reading, and asynchronous writing, the callback must become valid.

```
m_OPCCConnPoint.Advise(this, out m_iCallBackConnection);
```

❖ READ OUT

This example shows that the OPCGroup object is cast to OPCSyncIO2 interface and execute synchronous reading 1 item from device. The quality and the timestamp can be acquired from the OPC server besides the value. Delete the memory of "ppItemVal" and "ppErrors" at the end. Besides this, the OPCGroup object enables to be cast to OPCAsyncIO3 interface and execute asynchronous reading.

```
OPCSyncIO = (IOPCSyncIO2)m_OPCCGroup;        // Cast as IOPCSyncIO2
OPCSyncIO.Read(OPCDATASOURCE.OPC_DS_DEVICE, 1, ServerHd, out ppItemVal, out ppErrors);
                                                // Synchronous reading

ItemState = (OPCITEMSTATE)Marshal.PtrToStructure(ppItemVal, typeof(OPCITEMSTATE));
vValue = ItemState.vDataValue;                // Get value
fTime = ItemState.ftTimeStamp;                // Get timestamp
wQuality = ItemState.wQuality;                // Get quality
Marshal.FreeCoTaskMem(ppItemVal);
Marshal.FreeCoTaskMem(ppErrors);
```

❖ WRITE IN

This example shows that the OPCGroup object is cast to OPCSyncIO2 interface and execute synchronous writing 1 item. Delete the memory of "ppErrors" at the end. Besides this, the OPCGroup object enables to be cast to OPCAsyncIO3 interface and execute asynchronous writing.

```
OPCSyncIO = (IOPCSyncIO2)m_OPCCGroup;        // Cast as IOPCSyncIO2
vValue = 12345;
OPCSyncIO.Write(1, ServerHd, vValue, out ppErrors); // Synchronous writing
Marshal.FreeCoTaskMem(ppErrors);
```

❖ NOTIFICATION OF DATA CHANGE

When the value of the device changes in the PLC, this event is notified by OPC server. Refer to "phClientItems" to identify which item the change occurred.

```
public void OnDataChange(
    int                dwTransid,
    int                hGroup,
    int                hrMasterquality,
    int                hrMastererror,
    int                dwCount,           // Number of items where change occurs
    int[]              phClientItems,     // Item ID for client
    object[]           pvValues,         // Value
    short[]            pwQualities,      // Quality
    OpcRcw.Da.FILETIME[] pftTimeStamps, // Timestamp
    int[]              pErrors)          // Error
{
    DataChange(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, pwQualities, pErrors);
    // Generate event
}
```

❖ NOTIFICATION OF ASYNCHRONOUS READING COMPLETION

When asynchronous reading is completed, this event is notified by OPC server. Refer to "phClientItems" to identify the read item.

```
public void OnReadComplete(
    int                dwTransid,
    int                hGroup,
    int                hrMasterquality,
    int                hrMastererror,
    int                dwCount,           // Number of items that reading is completed
    int[]              phClientItems,     // Item ID for client
    object[]           pvValues,         // Value
    short[]            pwQualities,      // Quality
    OpcRcw.Da.FILETIME[] pftTimeStamps, // Timestamp
    int[]              pErrors)          // Error
{
    ReadComplete(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, pwQualities, pErrors);
    // Generate event
}
```

❖ NOTIFICATION OF ASYNCHRONOUS WRITING COMPLETION

When asynchronous writing is completed, this event is notified by OPC server. Refer to "phClientItems" to identify the write item.

```
public void OnWriteComplete(
    int                dwTransid,
    int                hGroup,
    int                hrMastererror,
    int                dwCount,           // Number of items that writing is completed
    int[]              phClientItems,     // Item ID for client
    int[]              pErrors)          // Error
{
    WriteComplete(dwTransid, dwCount, phClientItems, pErrors); // Generate event
}
```

❖ NOTIFICATION OF ASYNCHRONOUS PROCESS CANCELLATION

When asynchronous process is canceled, this event is notified by OPC server.

```
public void OnCancelComplete(  
    int dwTransid,  
    int hGroup)  
{  
    CancelComplete(dwTransid);           // Generate event  
}
```

❖ DISCONNECT WITH THE SERVER

For disconnection with the OPC server, execute "RemoveGroup" method of OPCServer. Don't forget to delete objects after using.

```
if (m_OPCTGroup != null)  
{  
    ret = Marshal.ReleaseComObject(m_OPCTGroup); // Release the reference of OPCTGroup object  
    m_OPCTGroup = null;  
}  
if (m_OPCTConnPoint != null)  
{  
    ret = Marshal.ReleaseComObject(m_OPCTConnPoint);  
    // Release the reference of OPCTConnectionPoint object  
    m_OPCTConnPoint = null;  
}  
if (m_iServerGroup != 0)  
{  
    m_OPCTServer.RemoveGroup(m_iServerGroup, 0); // Remove OPCTGroup object  
    m_iServerGroup = 0;  
}  
ret = Marshal.ReleaseComObject(m_OPCTServer); // Release the reference of OPCTServer object
```

1.8 DXP Development Support Tool

1.8.1 Client Component (for .NET)

1.8.1.1 Introduction

Using a development support tool for exclusive use of the .NET which we provide, you can easily develop OPC client without programming.

DxpClientComponent is the .NET component which binds items of the DeviceXPlorer for the controls (e.g. TextBox) on the Form in client development using Visual Studio. You can show realtime data at the controls automatically.

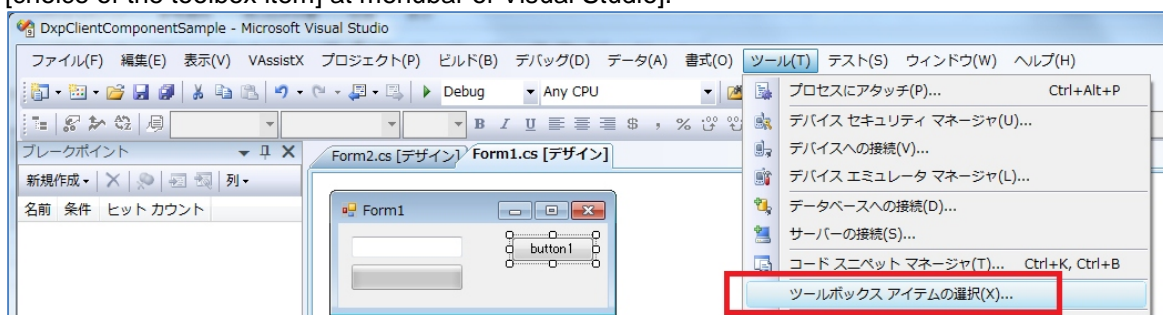
Important

Only user who buy Enterprise license of DeviceXPlorer can use DxpClientComponent.
So if you don't buy license and you buy only standard license, don't use DxpClientComponent.

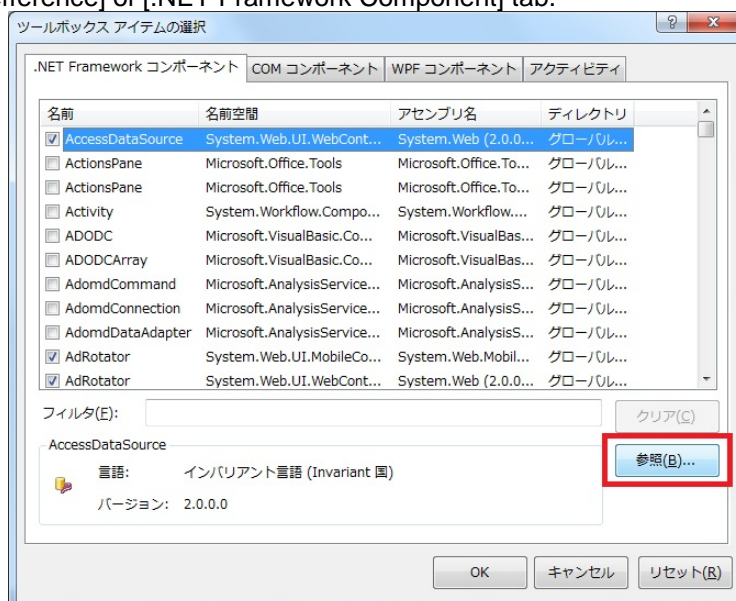
1.8.1.2 Setting of Development Environment

Procedure to show components on the ToolBox of Visual Studio

I explain a procedure to register a component to the ToolBox of Visual Studio 2008. At first, click [tool] > [choice of the toolbox item] at menubar of Visual Studio.

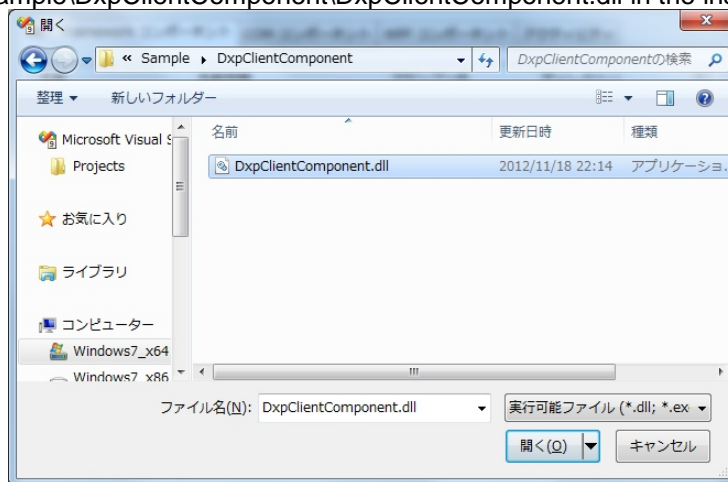


Click [Reference] of [.NET Framework Component] tab.

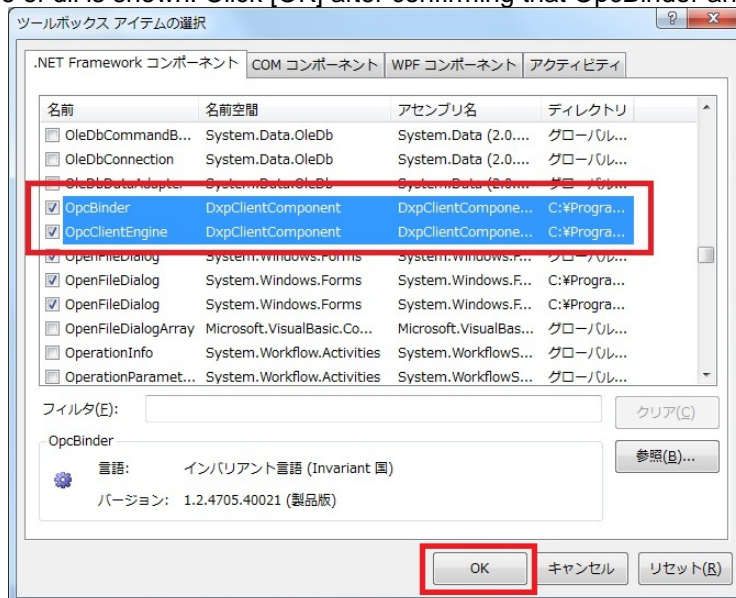


1.7 Visual C# Sample (for OPC Custom Interface)

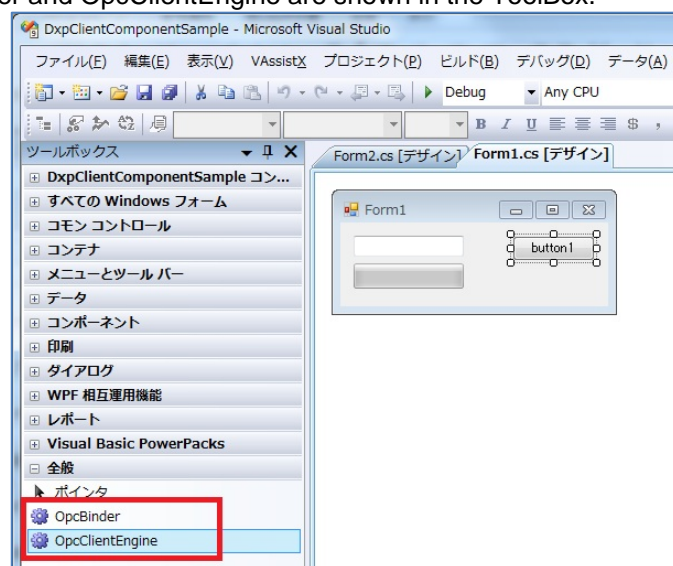
Select Sample\DxpClientComponent\DxpClientComponent.dll in the install directory.



The name of dll is shown. Click [OK] after confirming that OpcBinder and OpcClientEngine are checked.



OpcBinder and OpcClientEngine are shown in the Toolbox.

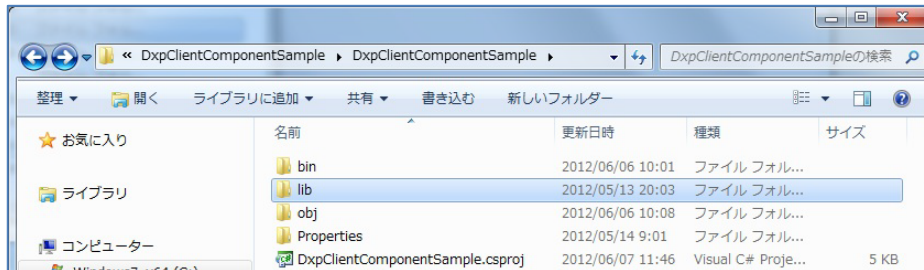


1.7 Visual C# Sample (for OPC Custom Interface)

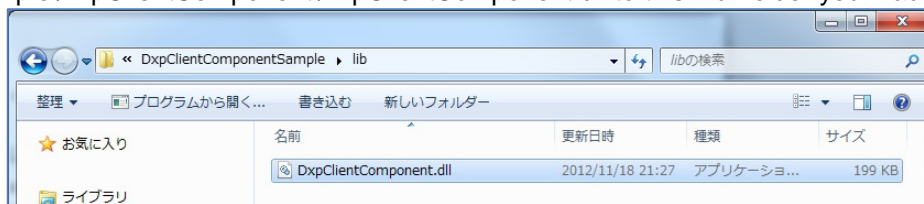
Procedure to register reference of project

Register DxpClientEngine.dll to the reference of project for the Visual C# or Visual Basic.NET.

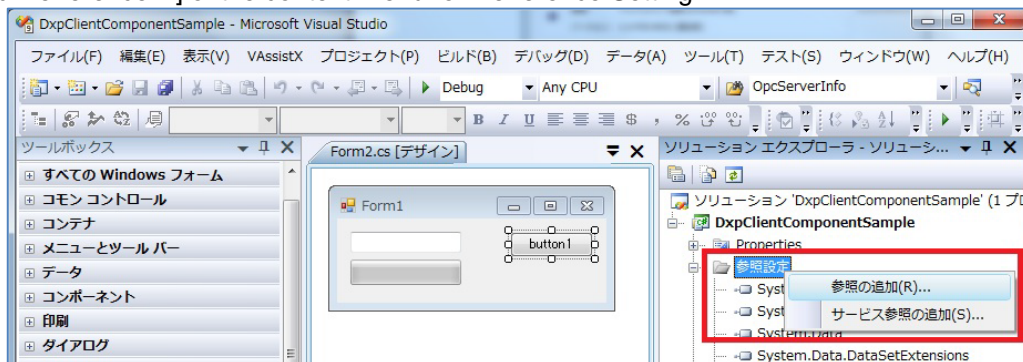
At first, make new solution and project of the Visual C#. Next, make "lib" folder at the project folder of Visual Studio.



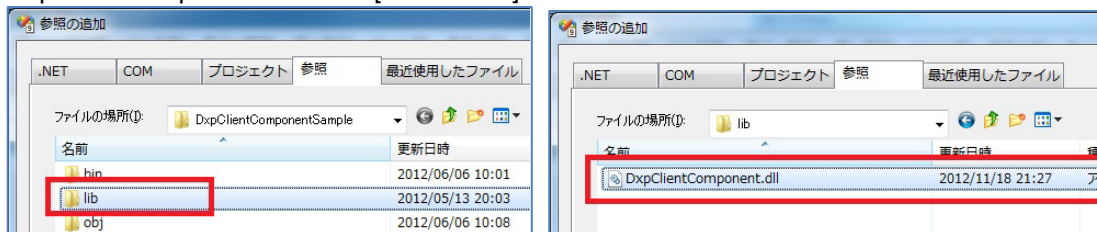
Copy Sample\DxpClientComponent\DxpClientComponent.dll to this "lib" folder you made.



Click [Add Reference...] of the context-menu for Reference Setting.



Select lib\DxpClientComponent.dll in the [Reference] tab.



Finished.

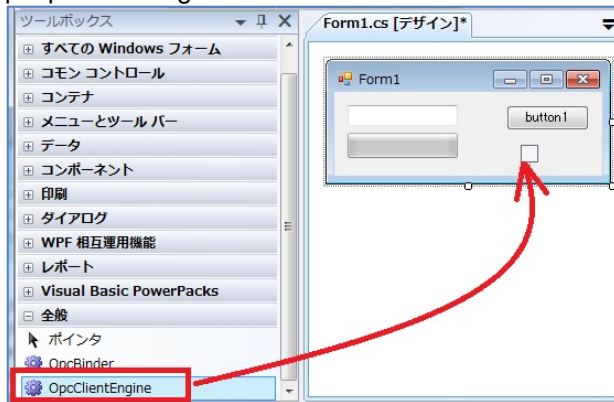


1.7 Visual C# Sample (for OPC Custom Interface)

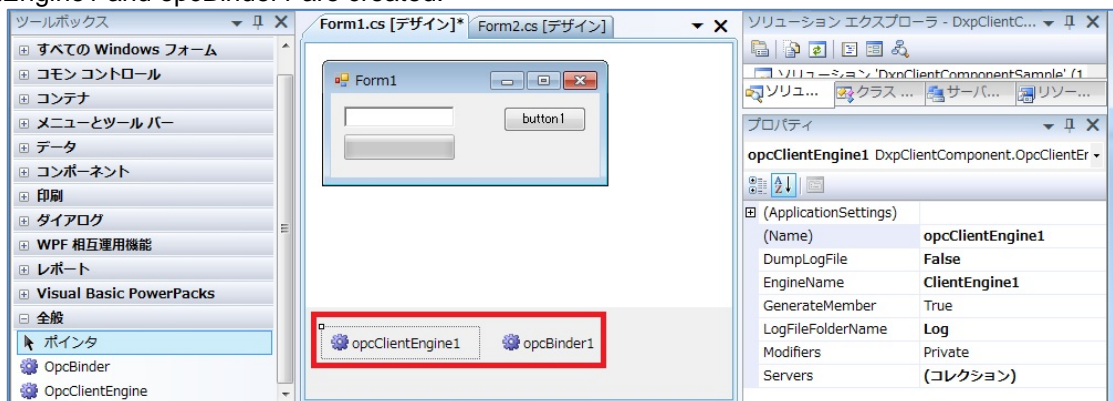
1.8.1.3 How to develop client

Procedure to use the component

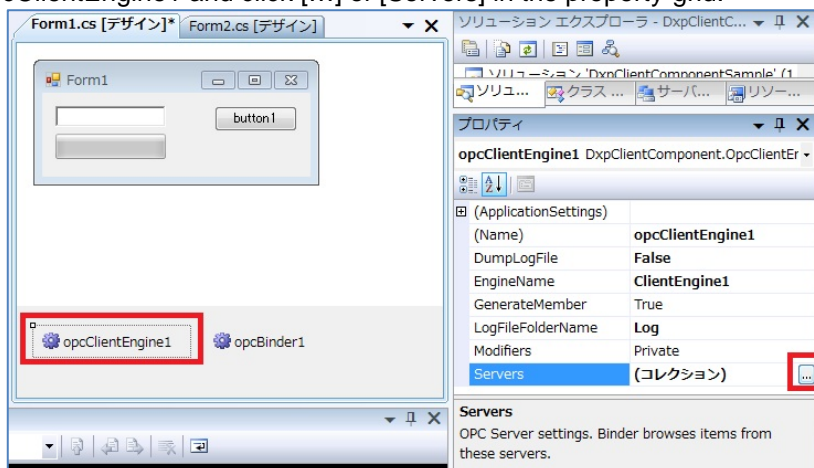
Drag&drop OpcClientEngine to the Form from ToolBox of Visual Studio.



Drag&drop OpcBinder to the Form from ToolBox of Visual Studio, too.
opcClientEngine1 and opcBinder1 are created.

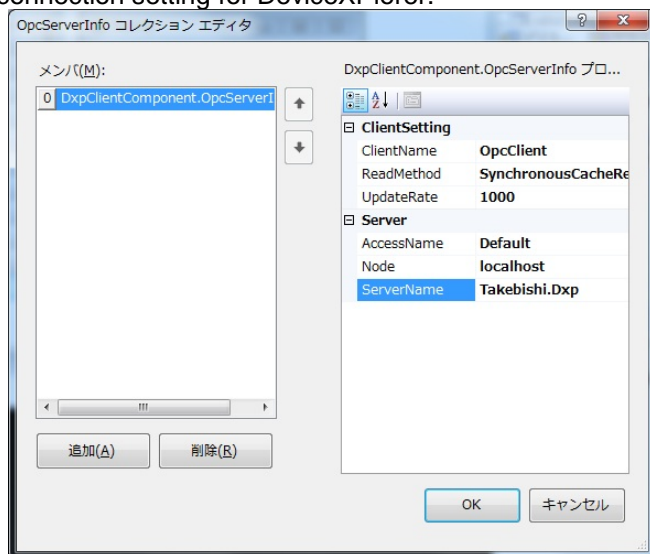


Select opcClientEngine1 and click [...] of [Servers] in the property-grid.

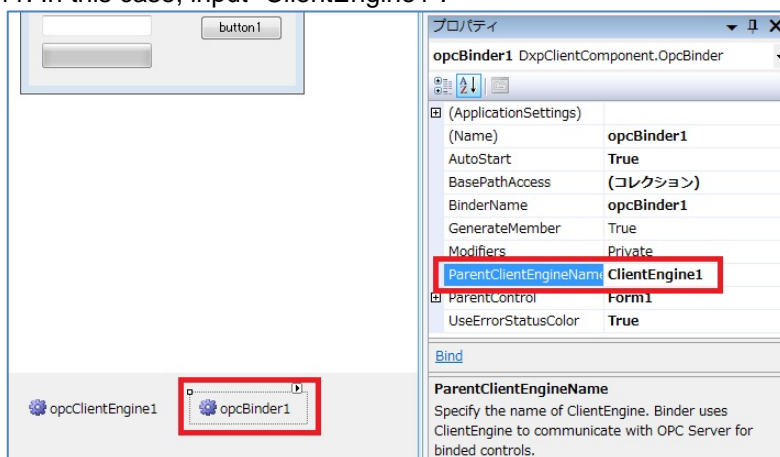


1.7 Visual C# Sample (for OPC Custom Interface)

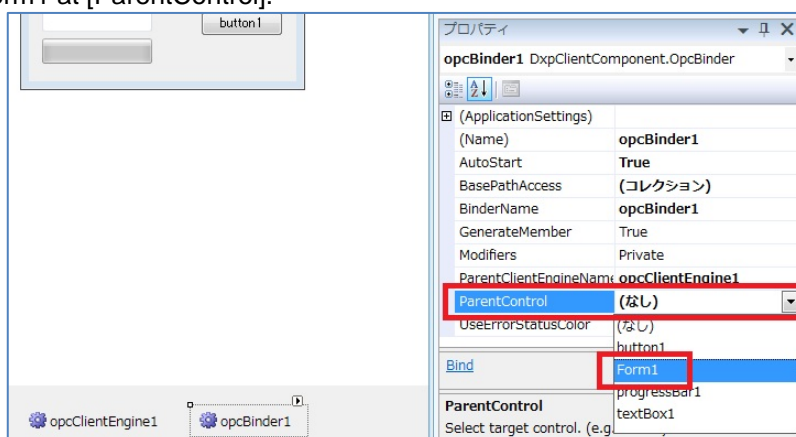
Confirm connection setting for DeviceXPlorer.



Input the string which is shown at [EngineName] of opcClientEngine1 to the [ParentClientEngineName] of opcBinder1. In this case, input "ClientEngine1".

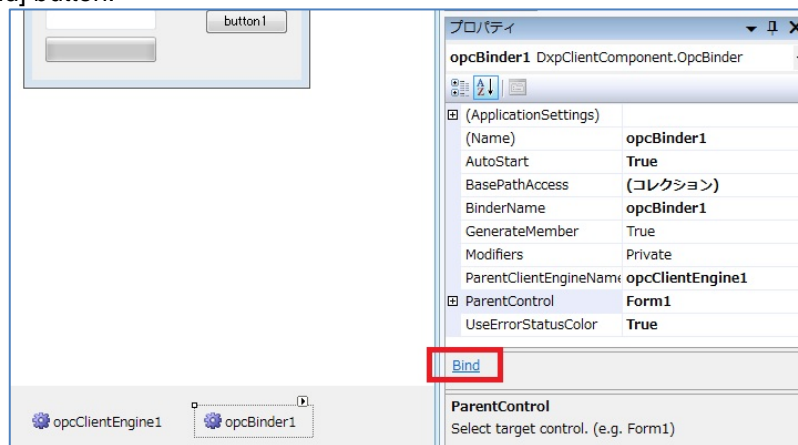


Select Form1 at [ParentControl].

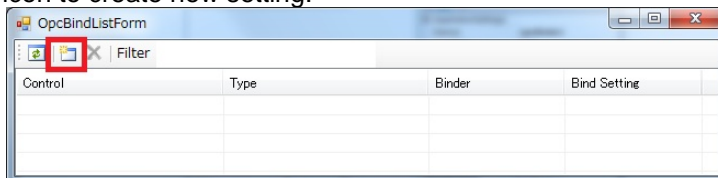


1.7 Visual C# Sample (for OPC Custom Interface)

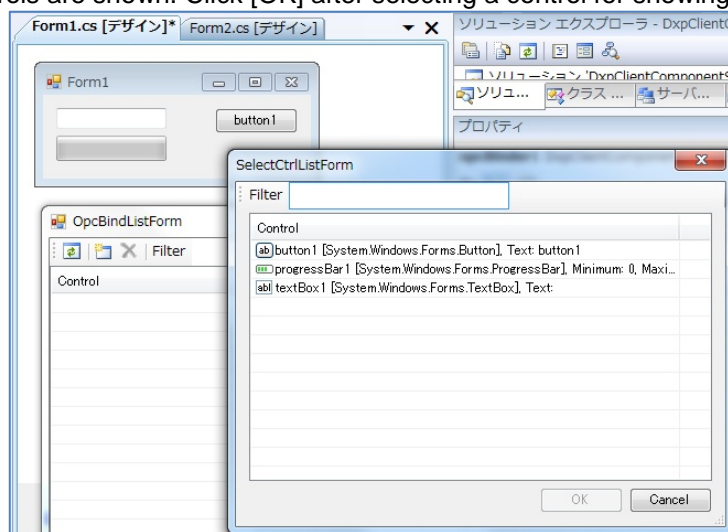
Click [Bind] button.



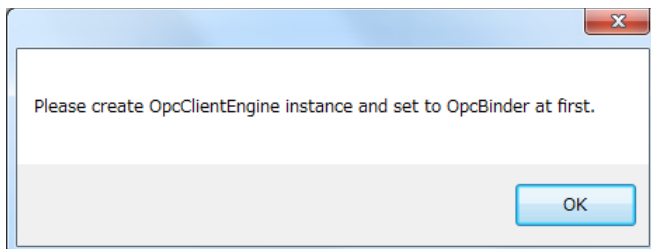
Click the icon to create new setting.



The controls are shown. Click [OK] after selecting a control for showing value.

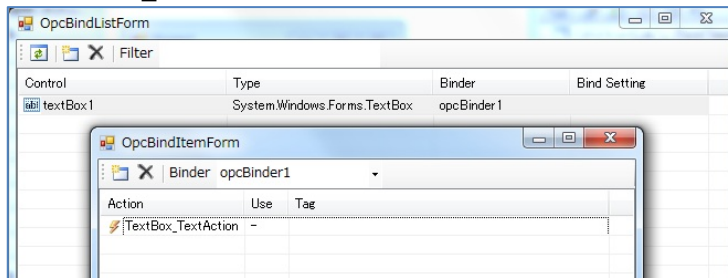


If the following message-box is shown, [ParentClientEngineName] of opcBinder1 is incorrect. Please modify.

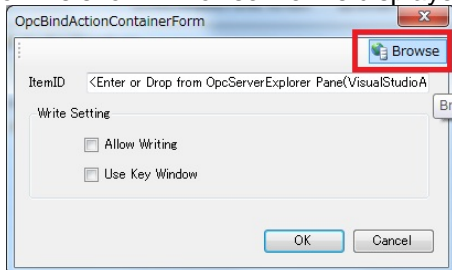


1.7 Visual C# Sample (for OPC Custom Interface)

TextBox_TextAction is shown. The “-“ means not-used at the column of [Use].
Double-click TextBox_TextAction.



Setting Form is shown. Browse-view is displayed when click [Browse] button.

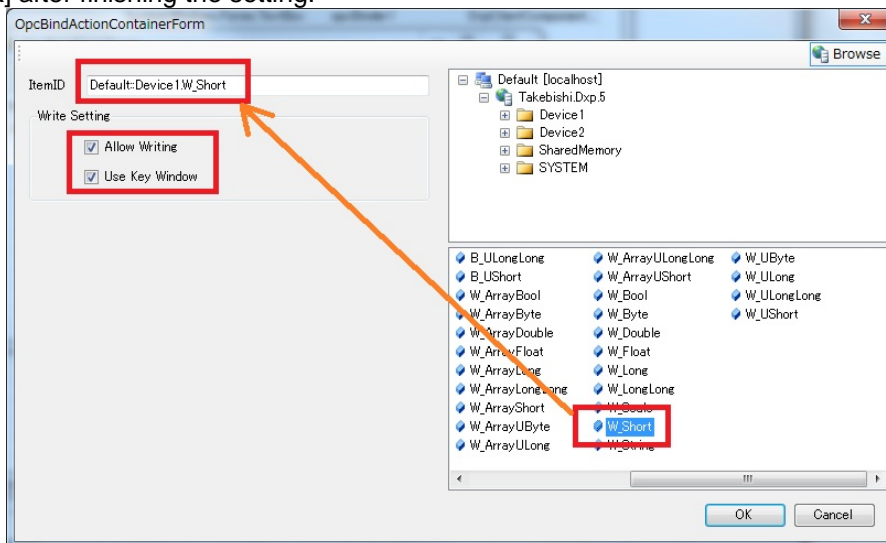


Browse-view connects to DeviceXPlorer when click [+] of the “Takebishi.Dxp” node in the Browse-view.
ItemId is filled out by double-clicking the tag.

If you want to allow writing by user, you can check [Allow Writing].

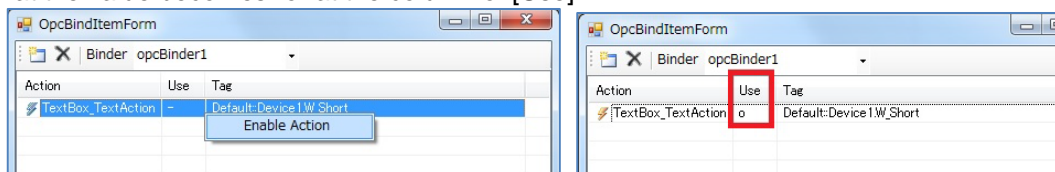
If you want to use 10-Key window, you can check [Use Key Window].

Click [OK] after finishing the setting.



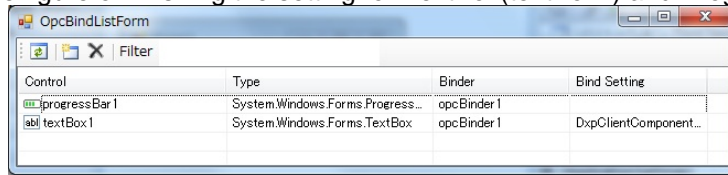
Click [Enable Action] in the context-menu by right-click for enabling the action.

Confirm that the value becomes “o” at the column of [Use].

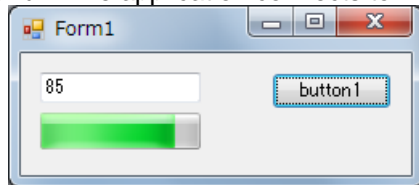


1.7 Visual C# Sample (for OPC Custom Interface)

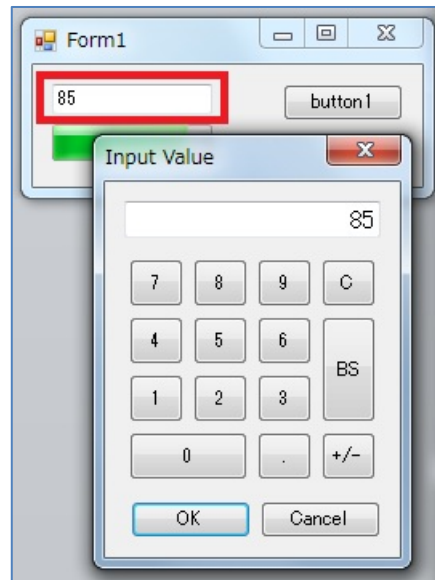
Please make the setting for ProgressBar by the same procedure.
Click [x] button or hit escape-key for closing the setting windows.
This is the figure of finishing the setting for TextBox(textBox1) and ProgressBar(progressBar1).



Build and run. The application connects to DeviceXPlorer and shows the value automatically.



You can write value to the DeviceXPlorer through the key-window by clicking TextBox.



1.8.1.4 Samples

In media disk of DeviceXPlorer, there are DxpClientComponent samples.

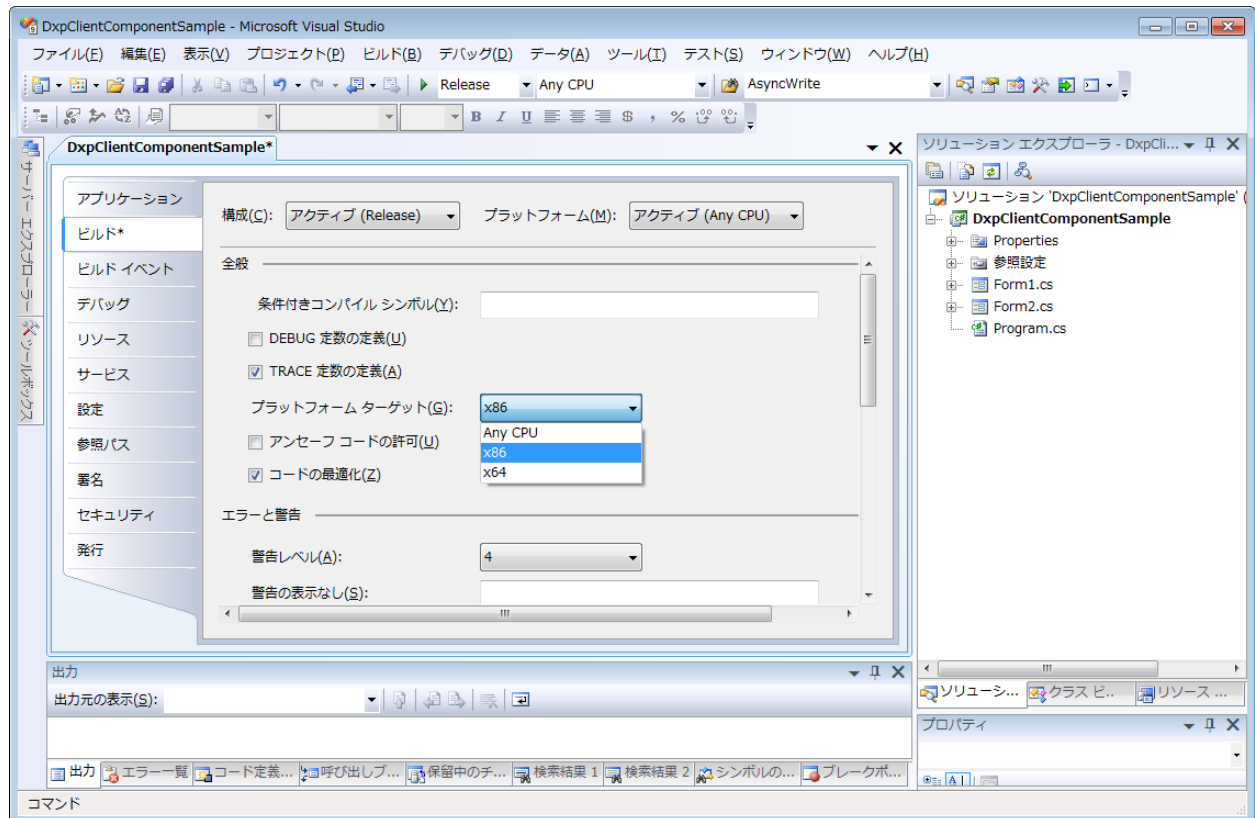
For VC# sample <Disk Directory>\Samples\DxpClientComponent\DxpClientComponentSample
DLL <Disk Directory>\Samples\DxpClientComponent\DxpClientComponent.dll

1.7 Visual C# Sample (for OPC Custom Interface)

1.8.1.5 Note in building sample on Windows for x64

DxpClientComponent DLL is built for 32 bits. When you build the sample of the OPC client by VisualStudio on 64 bits Windows, please select x86 of the platform target setting. If you selected AnyCPU or x64 and built, application cannot work normally at the runtime.

The platform target setting can be found by opening 'Project Properties' and selecting the 'Build' tab.



1.8.2 Simple API (for .NET)

1.8.2.1 Introduction

To use DxpSimpleAPI in Visual C# and Visual Basic .NET, select "Reference" from "Project" menu. And then, click "Reference" button and select the following files.

- <Install directory>\Sample\DxpSimpleAPI\DxpSimpleAPI.dll

Important

Only user who buy Enterprise license of DeviceXPlorer can use DxpSimpleAPI.
So if you don't buy license and you buy only standard license, don't use DxpSimpleAPI.

1.8.2.2 Samples

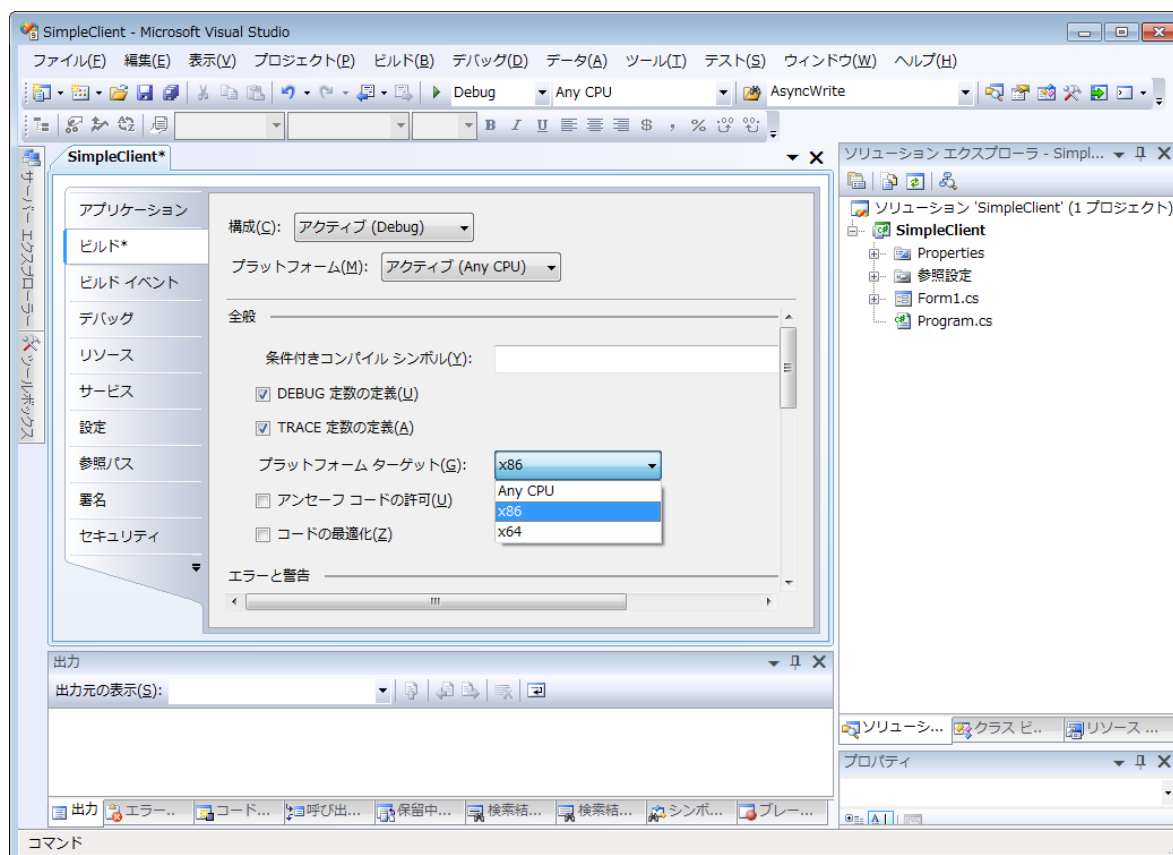
In media disk of DeviceXPlorer, there are DxpSimpleAPI samples.

For VC#	<Disk Directory>\Samples\DxpSimpleAPI\SimpleClient
For VB.NET	<Disk Directory>\Samples\DxpSimpleAPI\SimpleClientVB
DLL	<Disk Directory>\Samples\DxpSimpleAPI\DxpSimpleAPI.dll

1.8.2.3 Note in building sample on Windows for x64

DxpSimpleAPI DLL is built for 32 bits. When you build the sample of the OPC client by VisualStudio on 64 bits Windows, please select x86 of the platform target setting. If you selected AnyCPU or x64 and built, application cannot work normally at the runtime.

The platform target setting can be found by opening 'Project Properties' and selecting the 'Build' tab.



1.9 EXCEL Communication Support Tool (OPCFunction)

1.9 EXCEL Communication Support Tool (OPCFunction)

This communication support works as Add-In tool of EXCEL and you can access the OPC server only by inputting the OPC function to the cell.

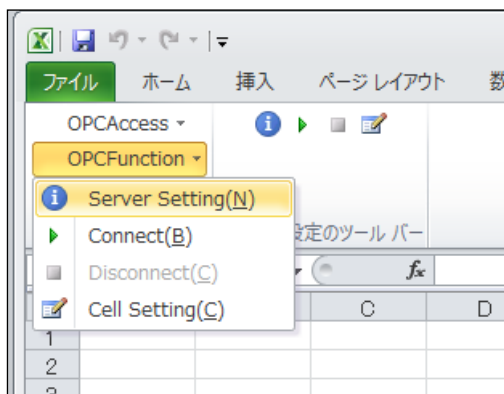
1.9.1 Operation

❖ ADD-IN

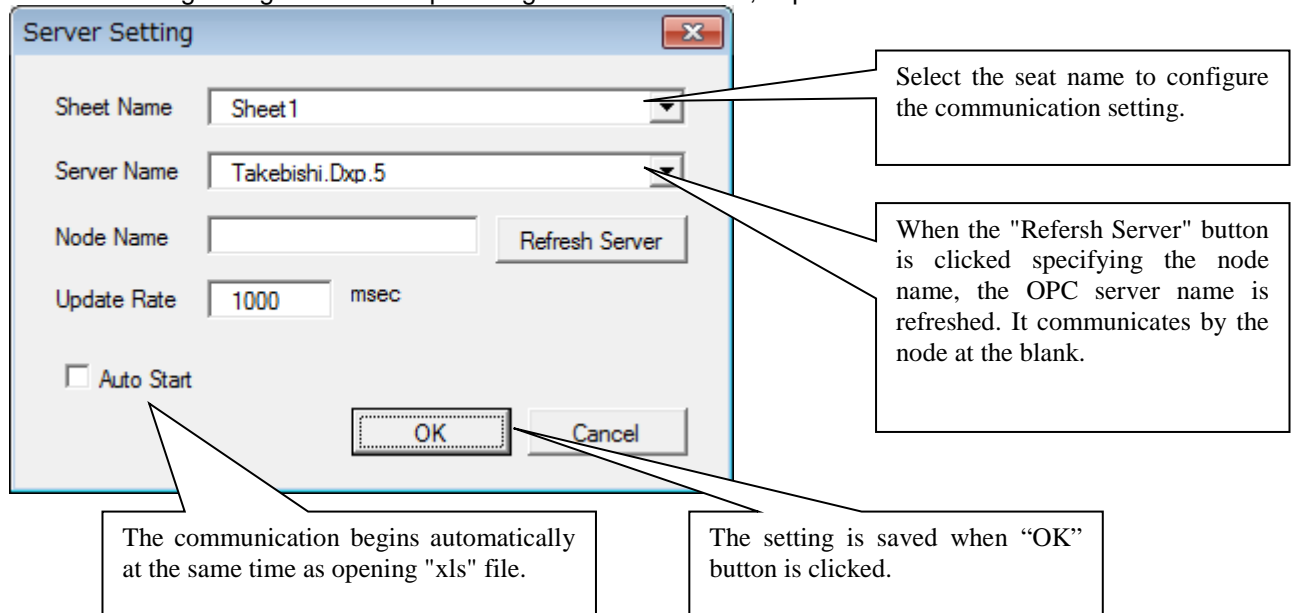
1. Execute EXCEL.
2. Select "Add-Ins" from "Tool" menu. Select the following file in Add-Ins dialog box.
<OPC Server install folder>\Sample\EXCEL\OPCFunction.xla
3. "OPCFunction" is added in menu. And the icon buttons are displayed.

❖ COMMUNICATION SETTING

1. Select "Server Setting" from "OPCFunction" menu .



2. The following dialog box shows up. Configure "Sever Name", "Update Rate" etc.



❖ INPUT FUNCTION

When the OPC function "OPCValue" is input to the cell, you can monitor the value.

=OPCValue("OPC Item ID")

exp. When you monitor the value of an item "Device1.D0".

=OPCValue("Device1.D0")

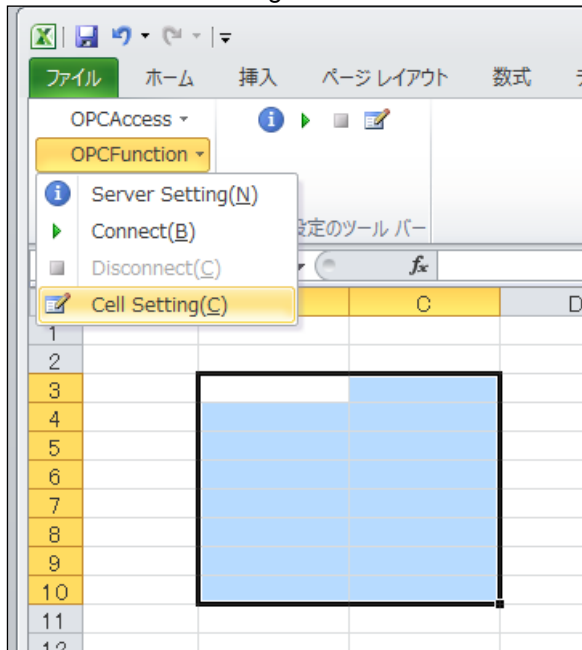
You can specify extended item type. However, the array type is not supported.

1.9 EXCEL Communication Support Tool (OPCFunction)

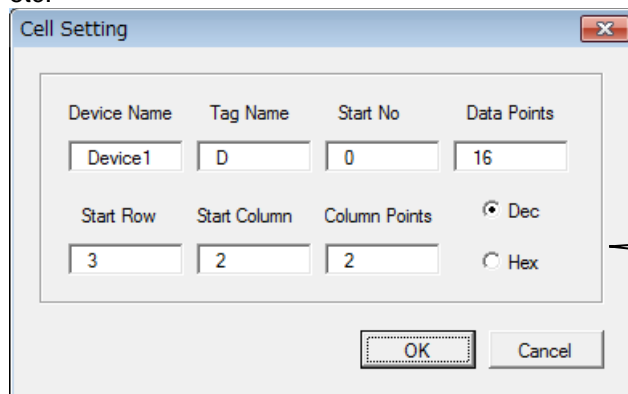
The OPC function “OPCValue” of sequence device can be set within the selected range of the cell.

1. Select cells.

2. Select “Batch Setting” from “OPCFunction” menu.



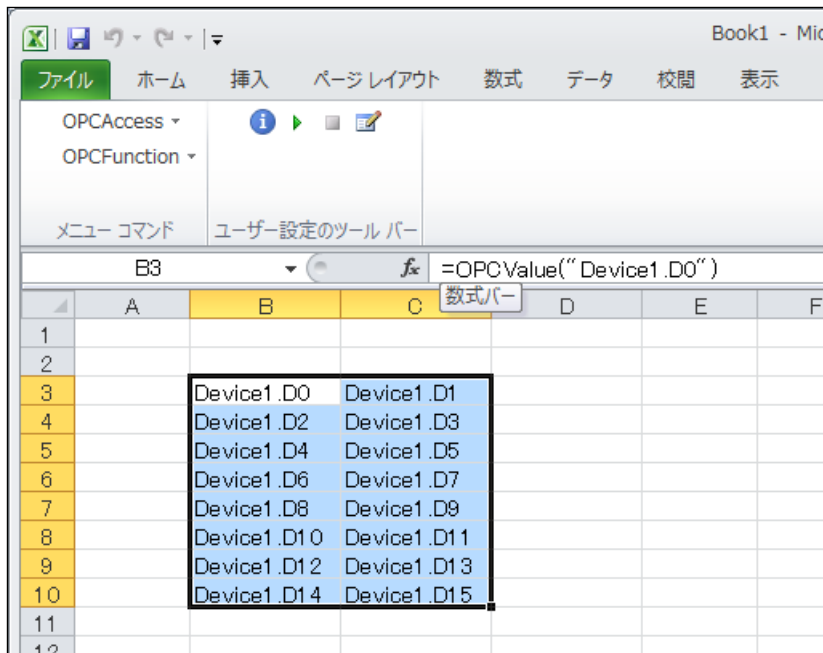
3. The following dialog box shows up. Configure “Device Name”, “Tag Name”, “Start No” and “Data Points” etc.



Select either decimal or hexadecimal about the PLC device number.

In this case, you cannot specify extended item type and 1000 points or more

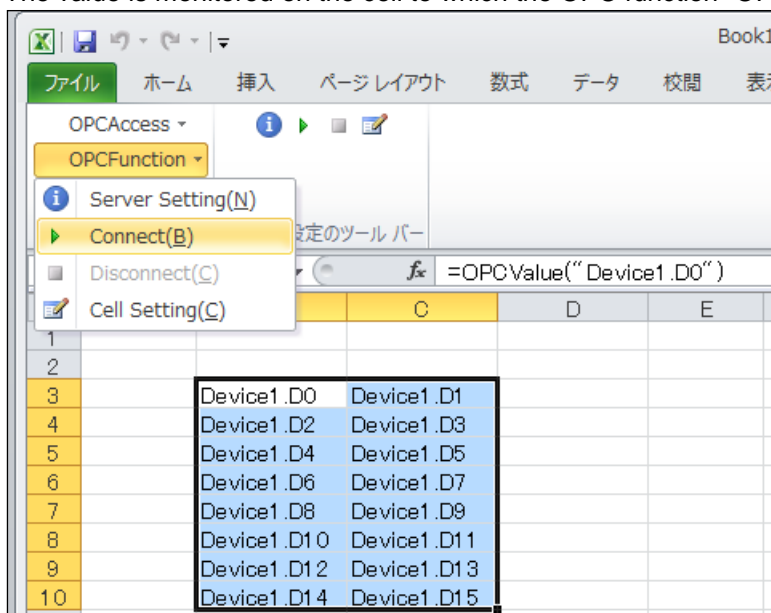
4. The OPC function “OPCValue” can be set within the selected range of the cell when “OK” button is clicked.



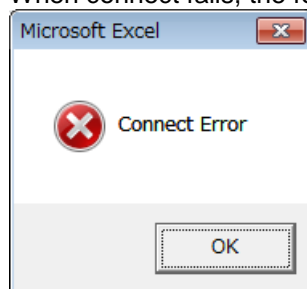
❖ STARTING AND STOPPING THE COMMUNICATION (READ OF ITEM)

To start communication select “Connect” from “OPCFunction” menu.

The value is monitored on the cell to which the OPC function “OPCValue” is set.

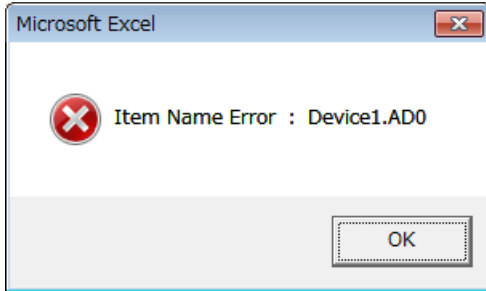


When connect fails, the following messages are displayed.



Confirm OPC server name of “Communication Setting”.

When Item ID of OPC function “OPCValue” is illegal, the following messages are displayed.

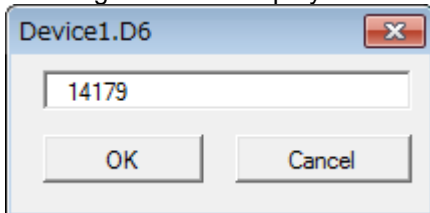


Confirm Item ID.

To stop communication select “Disconnect” from “OPCFunction”.

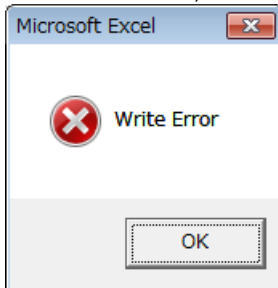
❖ WRITE OF ITEM

When you right-click on the cell to which the OPC function “OPCValue” is set during communicating, the following window is displayed.



Input the value to the box and click “OK” button. Then the value is written in PLC.

When write fails, the following messages are displayed.



1.10 EXCEL Sample (OPCDataAccess)

This sample program works as Add-In tool of EXCEL and value of devices can be monitored in selected cell easily.

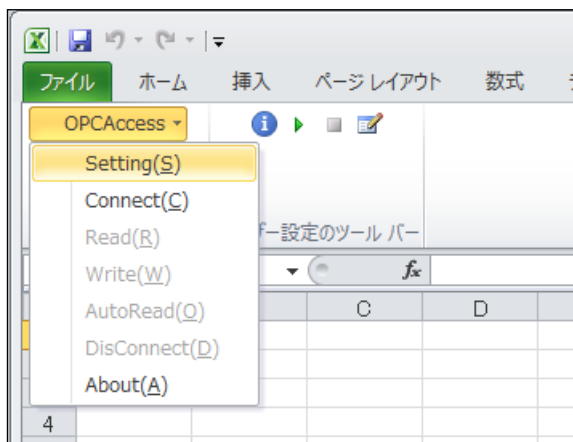
1.10.1 Operation

❖ ADD-IN

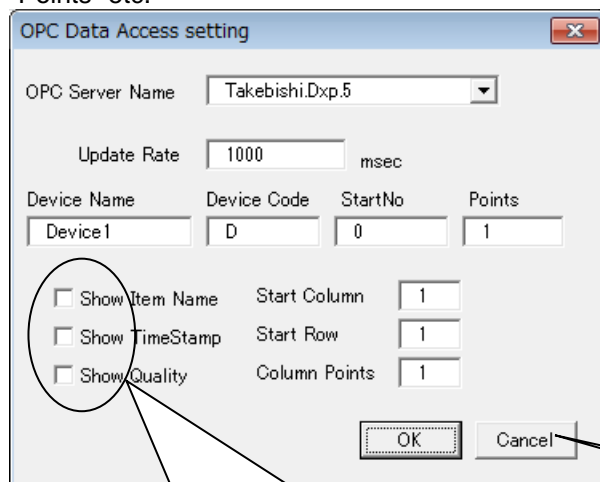
1. Execute EXCEL.
2. Select "Add-Ins" from "Tool" menu. Select the following file in Add-Ins dialog box.
<OPC Server install folder>\Sample\EXCEL\OPCDataAccess.xla
3. "OPCAccess" is added in menu.

❖ COMMUNICATION SETTING

1. Select cells.
2. Select "Setting" from "OPCAccess" menu.



3. The following dialog box shows up. Configure "OPC Sever Name", "Device Name", "Start No" and "Points" etc.



- "Show Item Name" Show item name on sheet.
- "Show Time Stamp" Show time on sheet.
- "Show Quality" Show quality on sheet.

The setting is saved when "OK" button is clicked.

1.10 EXCEL Sample (OPCDataAccess)

❖ STARTING AND STOPPING THE COMMUNICATION

To start communication select "Connect" from "OPCAccess" menu.

If connection succeeds, "Setting" and "Connect" menu become invalid and "Read", "Write", "AutoRead" and "Disconnect" become valid.

When "Disconnect" is selected, communication will be stopped.

❖ READ AND WRITE OF ITEM

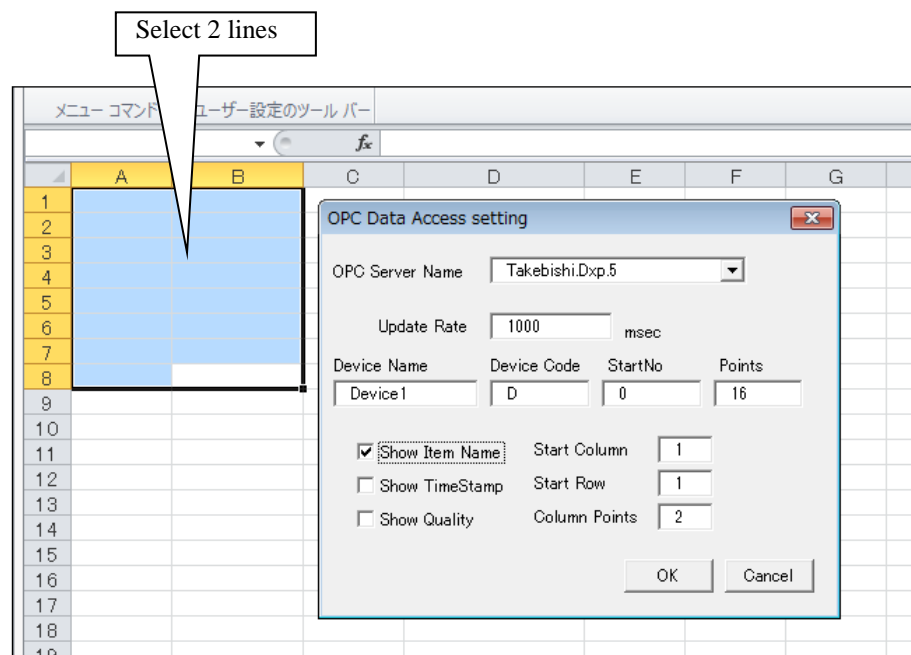
When "Read" is selected, it reads device once during communication established.

When "AutoRead" is selected, it reads device automatically during communication established.

When "Write" is selected after the value is poked, batch poke will execute during communication established.

❖ SAMPLE SCREEN

Case 1. Read 16 points from Device1.D0 after item name display becomes valid.



メニュー コマンド		ユーザー設定のツール バー			
D8		fx 1600			
	A	B	C	D	E
1	Device1.D0	100	Device1.D1	900	
2	Device1.D2	200	Device1.D3	1000	
3	Device1.D4	300	Device1.D5	1100	
4	Device1.D6	400	Device1.D7	1200	
5	Device1.D8	500	Device1.D9	1300	
6	Device1.D10	600	Device1.D11	1400	
7	Device1.D12	700	Device1.D13	1500	
8	Device1.D14	800	Device1.D15	1600	
9					

Item name

Value

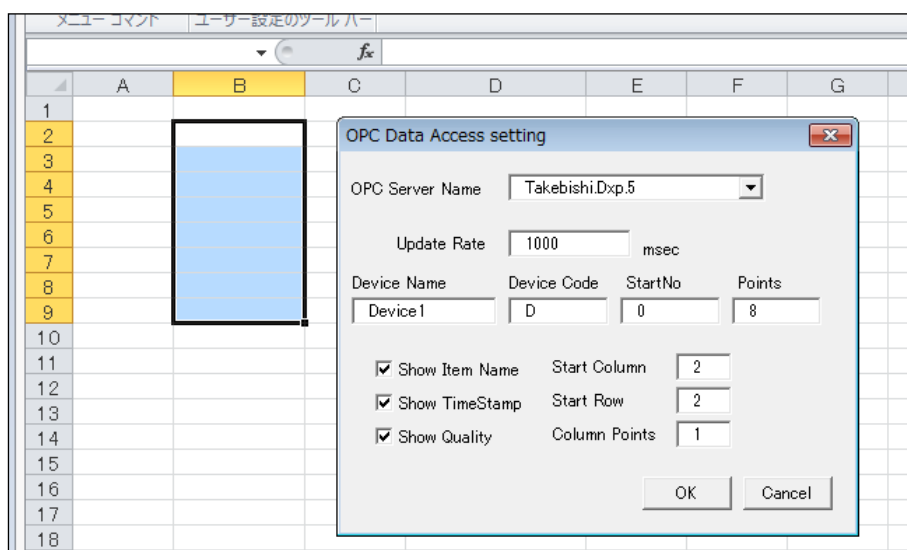
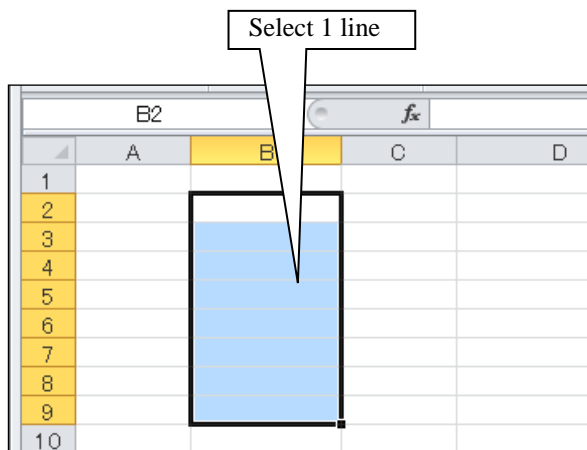
Item name

Value

Combinations of item names and values of device will be displayed in 4 lines (2 lines and 2set).

1.10 EXCEL Sample (OPCDataAccess)

Case 2. Read 8 points from Device1.D0 after item name and quality and timestamp become valid.



	A	B	C	D	E	F
1						
2		Device1.D0	19512	2012/4/23 12:16		192
3		Device1.D1	19512	2012/4/23 12:16		192
4		Device1.D2	11	2012/4/23 12:16		192
5		Device1.D3	19512	2012/4/23 12:16		192
6		Device1.D4	19512	2012/4/23 12:16		192
7		Device1.D5	19512	2012/4/23 12:16		192
8		Device1.D6	19512	2012/4/23 12:16		192
9		Device1.D7	19512	2012/4/23 12:16		192
10						

Item name Value Timestamp Quality

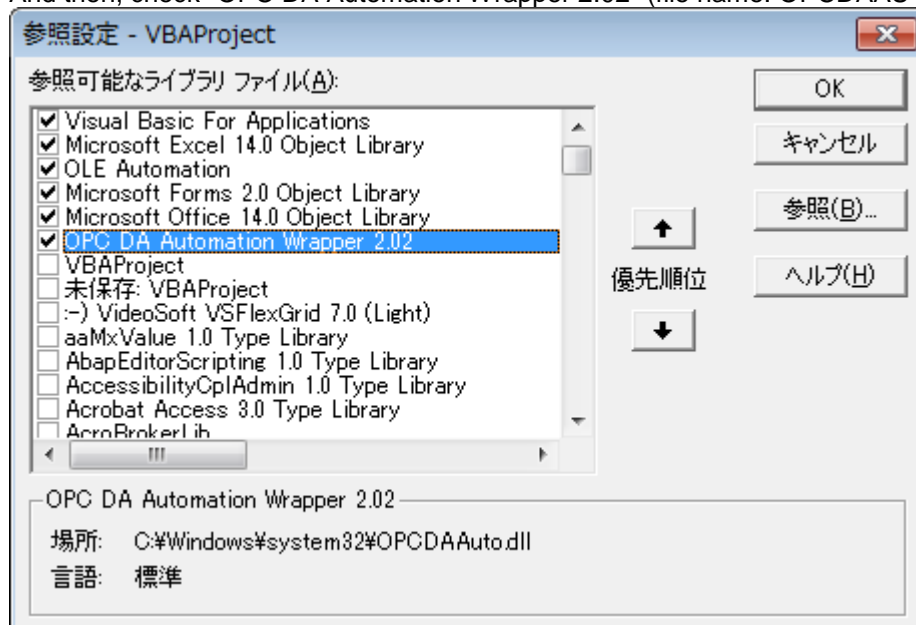
Combinations of item names, values of device, time stamp and quality will be displayed in 4 lines (2 lines and 2set).

1.10 EXCEL Sample (OPCDataAccess)

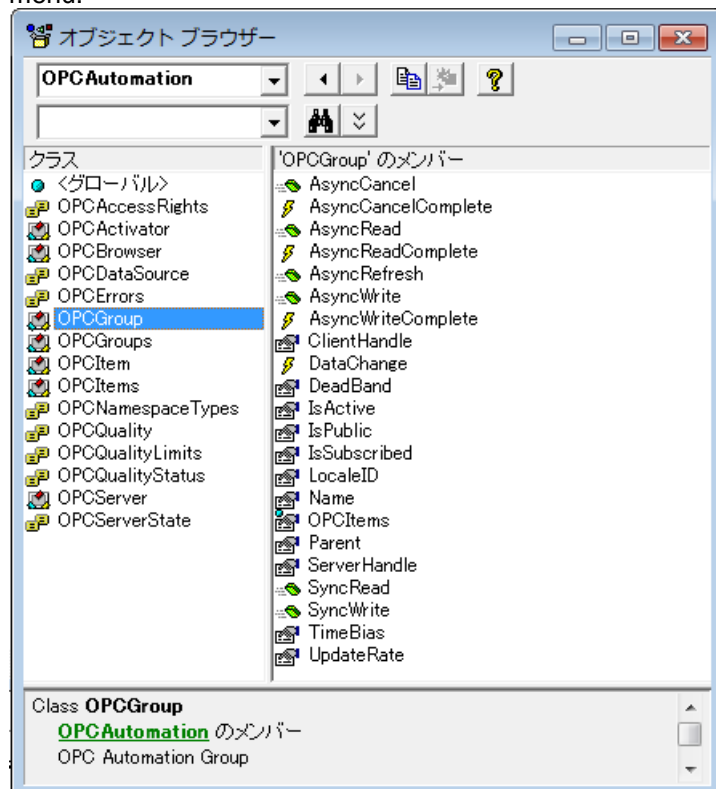
1.10.2 Setting of Development Environment

To make modifications to EXCEL, select "Macro" and "Visual Basic Editor" from "Tool" menu, then Visual Basic Editor starts up.

To use OPC Automation Interface, select "Reference" from "Tool" menu in Visual Basic Editor. And then, check "OPC DA Automation Wrapper 2.02" (file name: OPCDAAUTO.DLL).



The properties of the OPC server can be viewed from the object browser ("F2" key) located in the view menu.



1.10.3 Program Example

Refer to chapter 1.3.3 for details about program example that uses OPC Automation Interface with EXCEL.

1.10.4 How to uninstall

- Delete Add-in file

Please delete <OPC Server Install Directory>\Sample\Excel\OPCDataAccess.xla.

- Delete menu from EXCEL

<EXCEL2003>

Open "User settings" from EXCEL menu.

Select [Command] tab and Click [Rearrange Commands].

Check Toolbar button and select [Worksheet menu bar].

Select and Delete [OPC Access]

<EXCEL2007>

Select [OPC Access] on Ribbon menu of [Add-in] and right-click.

Remove the Command.

2 DDE Client

2.1 Test Client Program (DDE Test Client)

The use of the OPC client program appended to the product is shown.

2.1.1 Operation

You can execute at "DDE Client" at "program" at Start Menu of Windows, too.

Select AppName(Application Name) and Topic, and push "Connect" button. Application Name should be "DXPSV."

Enter the tagname (device name) in Item Name and press "Request" to read the value. If you press "Auto Read," it reads the value regularly.

Enter a value in Data and press "Poke" to write the value to the PLC.

Select the device type if you wish to use ASCII and double word, for example, select the device number in Number, access points in Points, and access type in Type to enable regular read, automatic read, and write.

Press "DisConnect" to disconnect from the server.

**TAKEBISHI Software Library
DeviceXPlorer
Client Sample Guide
User's manual**



TAKEBISHI Corporation

29 Mameda-cho, Nishikyogoku, Ukyo-ku, Kyoto, 615-8501 JAPAN

Phone: +81-75-325-2172

Email fa-support@takebishi.co.jp

