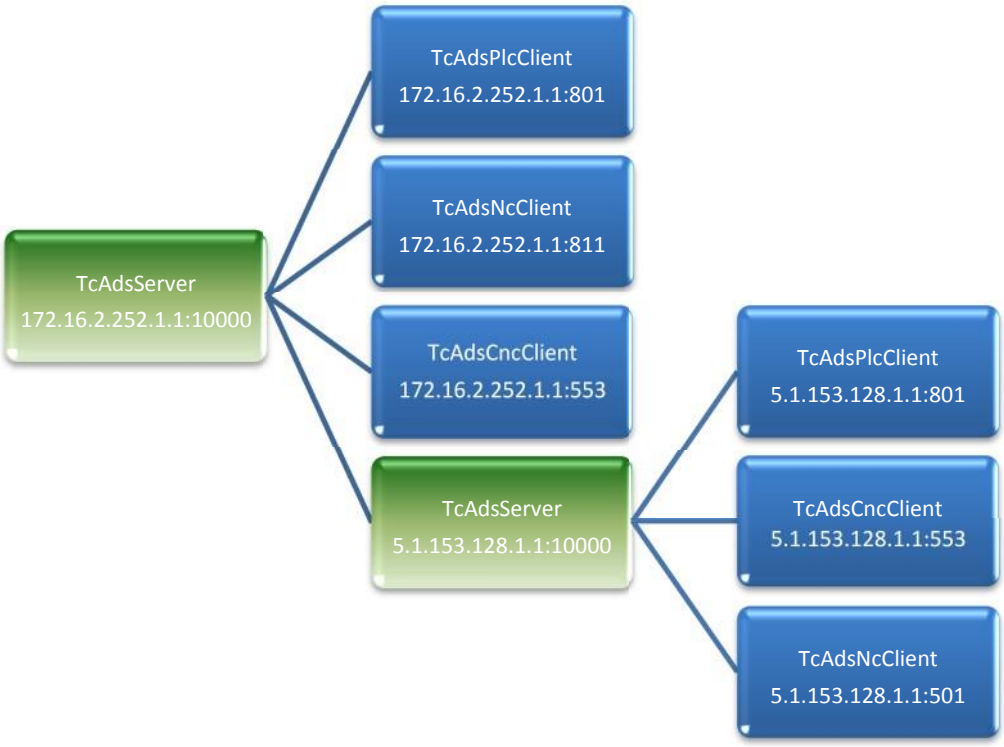
BECKHOFF

Beckhoff.App.Ads

1.9.2014 V 1.0

Structure of the ADS Wrappers (Example)



The internal structure of the real situation is shown in this image. Access to the instances of each is accomplished by means of interfaces. Entry point is by the interface **IBAAdsServer** which is implements the TcAsdServer. The interface can be specified in the constructor of a plugin Form (See Documentation "plugin for the HMI")

The method

```
TToBuild GetAdsClient<TToBuild>(string name);
```

returns a child client if it exists (otherwise NULL).

- To access the PLC use the following call:

  ```
  _plcClient = adsServer.GetAdsClient<IBAAdsPlcClient>("PLC");
  ```

  The name “PLC” is given in the setting for the main program and identifies the PLC client that is being used.

- To access the CNC use the following call:

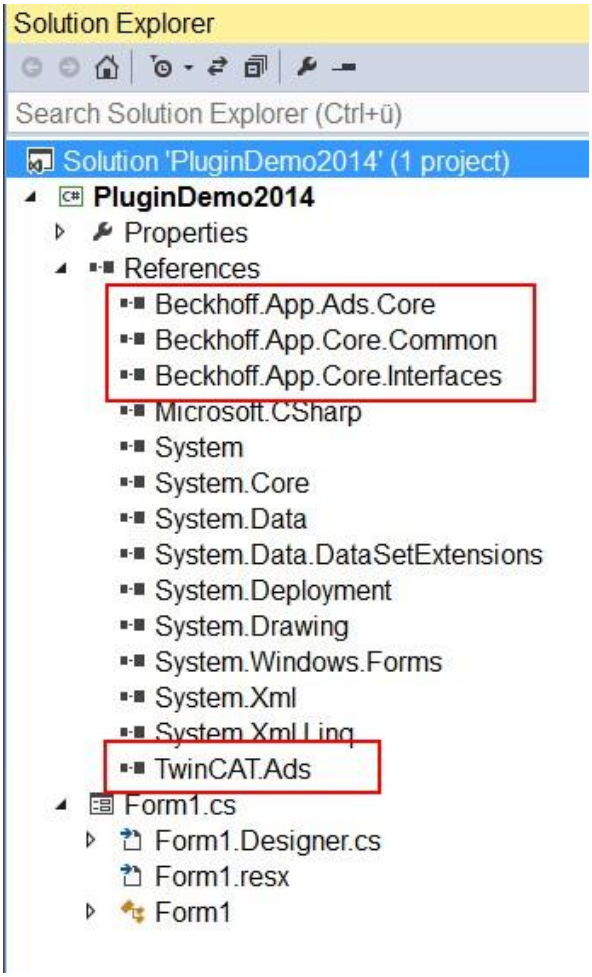
```
_cncClient = adsServer.GetAdsClient<IBAAdsCncClient>("CNC");
```

The name “CNC” is given in the setting for the main program and identifies the CNC client that is being used.

Required References

The following DLLs are required in order to incorporate the ADS wrapper:

Name of Dll	Location within the HMI
Beckhoff.App.Ads.Core.dll	References
Beckhoff.App.Core.Interfaces.dll	References
Beckhoff.App.Core.Common.dll	References
TwinCAT.Ads.dll	externalReferences



PLCClient synchronous read and write

The interface “IBAAdsPlcClient” can be used to synchronously read and write PLC variables.
Example:

```
 if
(_plcClient.SymbolExists("Global_HMI.bToggle"))
{
    var b =_plcClient.ReadSymbol<bool>("Global_HMI.bToggle");
    _plcClient.WriteSymbol("Global_HMI.bToggle", !b); }
```

Here we first check that the variable “Global_HMI.bToggle” exists in the PLC. If it does exists we then read it in as a BOOL and then write back out the inverse of its value.

PLCClient OnChangeNotification

Using the method “AddPlcDeviceNotification” it is possible to build a connection to a connection to a PLC variable that is connected to a callback method:

Example:

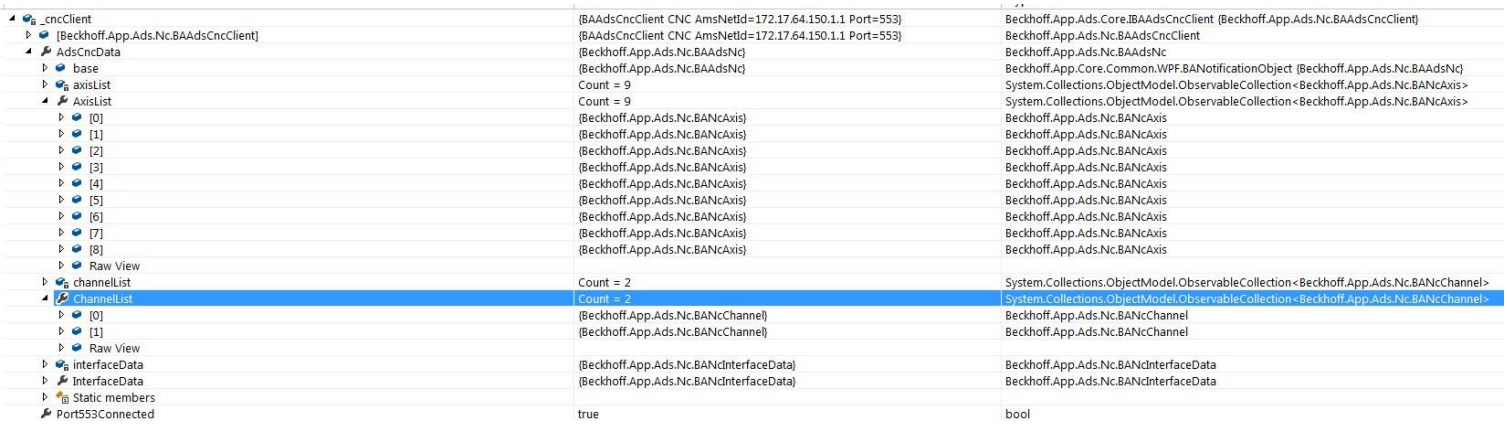
```
notify = new OnChangeDeviceNotification("Global_HMI.bToggle", PlcVarHandler);
_plcClient.AddPlcDeviceNotification(notify);
private void PlcVarHandler(object sender, BAAdsNotificationItem notification)
{
    if (notification.Name.Equals("Global_HMI.bToggle"))
    {
        if (notification.PlcObject is bool)
        {
            var b = (bool)(notification.PlcObject);
            this.label2.BackColor = b ? Color.Green : Color.Red;
        }
    }
}
```

Here an “OnChangeDeviceNotification“ connection is created. When the contents of the variable changes the method is called.
In order to save resources it is recommended that you also use “RemovePlcDeviceNotification“. Typically this would be done at some notification from the system that the connection is no longer needed. (For example, when Form Visible==false)

CNCClient

Within CNCClient is the information for the CNC defined in the AdsCncData class.
To use this client, the setting "General / newAdsClient" in the settings of the CNC HMI must be set to "true", so that it is generated at the start of the HMI. In the client there is a property AdsCncData which includes an "Axis List" and a "Channel List". In the Axis List all axes of the system are summarized. In the Channel list you can see all the configured channels, which in turn contains a ChannelAxisList describing all the axes in the specific channel.

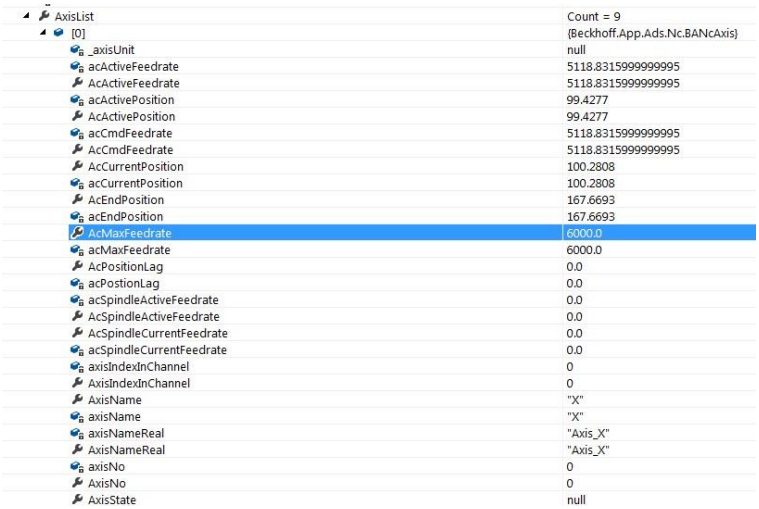
All properties and lists (Properties and Collections) implement the Microsoft interface "INotifyPropertyChanged".



_cncClient	{BAAdsCncClient CNC AmsNetId=172.17.64.150.1.1 Port=553}	Beckhoff.App.Ads.Core.IBAAdsCncClient {Beckhoff.App.Ads.Nc.BAAdsCncClient}
[Beckhoff.App.Ads.Nc.BAAdsCncClient]	{BAAdsCncClient CNC AmsNetId=172.17.64.150.1.1 Port=553}	Beckhoff.App.Ads.Nc.BAAdsCncClient
AdsCncData	{Beckhoff.App.Ads.Nc.BAAdsNc}	Beckhoff.App.Ads.Nc.BAAdsNc
base	{Beckhoff.App.Ads.Nc.BAAdsNc}	Beckhoff.App.Core.Common.WPF.BANotificationObject {Beckhoff.App.Ads.Nc.BAAdsNc}
axisList	Count = 9	System.Collections.ObjectModel.ObservableCollection<Beckhoff.App.Ads.Nc.BANcAxis>
AxisList	Count = 9	System.Collections.ObjectModel.ObservableCollection<Beckhoff.App.Ads.Nc.BANcAxis>
[0]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[1]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[2]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[3]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[4]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[5]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[6]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[7]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
[8]	{Beckhoff.App.Ads.Nc.BANcAxis}	Beckhoff.App.Ads.Nc.BANcAxis
Raw View		
channelList	Count = 2	System.Collections.ObjectModel.ObservableCollection<Beckhoff.App.Ads.Nc.BANcChannel>
ChannelList	Count = 2	System.Collections.ObjectModel.ObservableCollection<Beckhoff.App.Ads.Nc.BANcChannel>
[0]	{Beckhoff.App.Ads.Nc.BANcChannel}	Beckhoff.App.Ads.Nc.BANcChannel
[1]	{Beckhoff.App.Ads.Nc.BANcChannel}	Beckhoff.App.Ads.Nc.BANcChannel
Raw View		
interfaceData	{Beckhoff.App.Ads.Nc.BANcInterfaceData}	Beckhoff.App.Ads.Nc.BANcInterfaceData
InterfaceData	{Beckhoff.App.Ads.Nc.BANcInterfaceData}	Beckhoff.App.Ads.Nc.BANcInterfaceData
Static members		
Port553Connected	true	bool

With “INotifyProprtyChanged“ event-based implementation can be realized. Example:

```
_cncClient.AdsCncData.AxisList[0].PropertyChanged += PropertyChangedCallback;
void PropertyChangedCallback(object sender, PropertyChangedEventArgs e)
{
    if (e.PropertyName.Equals("AcCurrentPosition"))
    {
        label1.Text = "1.Axis Position: " +
        _cncClient.AdsCncData.AxisList[0].AcCurrentPosition.ToString("0.00");
    }
}
```



AxisList	Count = 9
[0]	{Beckhoff.App.Ads.Nc.BANcAxis}
_axisUnit	null
acActiveFeedrate	5118.8315999999995
AcActiveFeedrate	5118.8315999999995
acActivePosition	99.4277
AcActivePosition	99.4277
acCmdFeedrate	5118.8315999999995
AcCmdFeedrate	5118.8315999999995
AcCurrentPosition	100.2808
acCurrentPosition	100.2808
AcEndPosition	167.6693
acEndPosition	167.6693
AcMaxFeedrate	6000.0
acMaxFeedrate	6000.0
AcPositionLag	0.0
acPostionLag	0.0
acSpindleActiveFeedrate	0.0
AcSpindleActiveFeedrate	0.0
AcSpindleCurrentFeedrate	0.0
acSpindleCurrentFeedrate	0.0
axisIndexInChannel	0
AxisIndexInChannel	0
AxisName	"X"
axisName	"X"
axisNameReal	"Axis_X"
AxisNameReal	"Axis_X"
axisNo	0
AxisNo	0
AxisState	null

Sourcecode Example:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data; using
System.Drawing; using
System.Linq; using System.Text;
using System.Windows.Forms;

namespace PluginDemo2014
{
using Beckhoff.App.Ads.Core;
using Beckhoff.App.Ads.Core.Plc;

    public partial class Form1 : Form
    {
        private IBAAdsCncClient _cncClient;
```

```
        private IBAAdsPlcClient _plcClient;

        private List<IBAAdsNotification> _notifications;

        public Form1()
        {
            InitializeComponent();
        }
        public Form1(IBAAdsServer adsServer) : this()
        {
                _notifications = new List<IBAAdsNotification>();
                _cncClient = adsServer.GetAdsClient<IBAAdsCncClient>("CNC");
                _plcClient = adsServer.GetAdsClient<IBAAdsPlcClient>("PLC");
                var notify = new OnChangeDeviceNotification("Global_HMI.bToggle", PlcVarHandler);
                _notifications.Add(notify);
                _cncClient.AdsCncData.AxisList[0].PropertyChanged += PropertyChangedCallback;
        }
        private void PlcVarHandler(object sender, BAAdsNotificationItem notification)
        {
            if (notification.Name.Equals("Global_HMI.bToggle"))
            {
                if (notification.PlcObject is bool)
                {
                    var b = (bool)(notification.PlcObject);
                    this.label2.BackColor = b ? Color.Green : Color.Red;
}
            }
        }
        void PropertyChangedCallback(object sender, PropertyChangedEventArgs e)
        {
            if (e.PropertyName.Equals("AcCurrentPosition"))
            {
                label1.Text = "1.Axis Position: "
                     + _cncClient.AdsCncData.AxisList[0].AcCurrentPosition.ToString("0.00");
            }
        }
        public void Test()
        {
            if (_plcClient.SymbolExists("Global_HMI.bToggle"))
            {
                var b =_plcClient.ReadSymbol<bool>("Global_HMI.bToggle");
                _plcClient.WriteSymbol("Global_HMI.bToggle", !b);
            }
        }
        private void Form1_VisibleChanged(object sender, EventArgs e)
        {
            if (Visible)
           {
                foreach (var notify in _notifications)
                {
                    _plcClient.AddPlcDeviceNotification(notify);
                }
            }
            else
            {
                foreach (var notify in _notifications)
                {
                    _plcClient.RemovePlcDeviceNotification(notify);
                }
            }
        }

    }
}
```