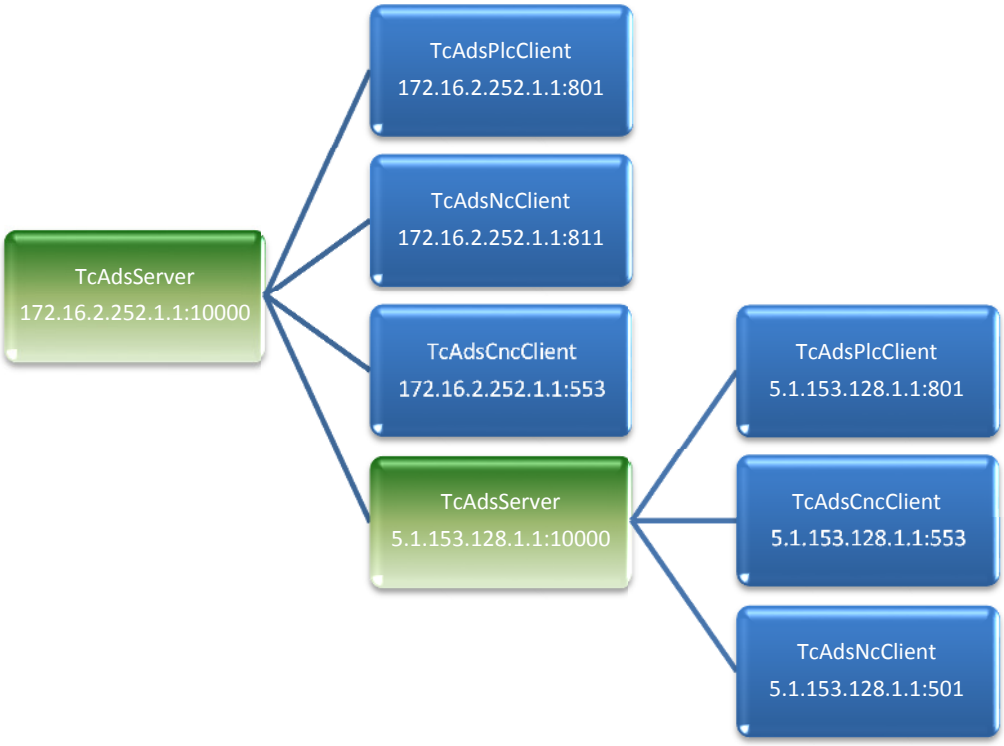
Struktur des ADS Wrappers (Beispiel)



Mit dem internen Aufbau wird ein Abbild der reellen Gegebenheiten gebildet.
Der Zugriff auf die jeweiligen Instanzen erfolgt mit Hilfe von Interfaces .
Einstiegspunkt ist das Interface **IBAAdsServer** welches von TcAsdServer implementiert wird. Das Interface kann im Konstruktor eines Plugin Forms angegeben werden (Vgl. Dokumentation „Plugin für die HMI“)

Die Methode

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

liefert einen untergeordneten Client, falls vorhanden (sonst NULL).

- Um Zugriff zur SPS zur erhalten heißt der Aufruf:

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

  Der Name „PLC“ ist im Hauptrahmen festgelegt und identifiziert den in den Settings aktuell eingestellten Client zum SPS Zugriff.

- Um Zugriff auf die CNC zu erhalten heißt der Aufruf:

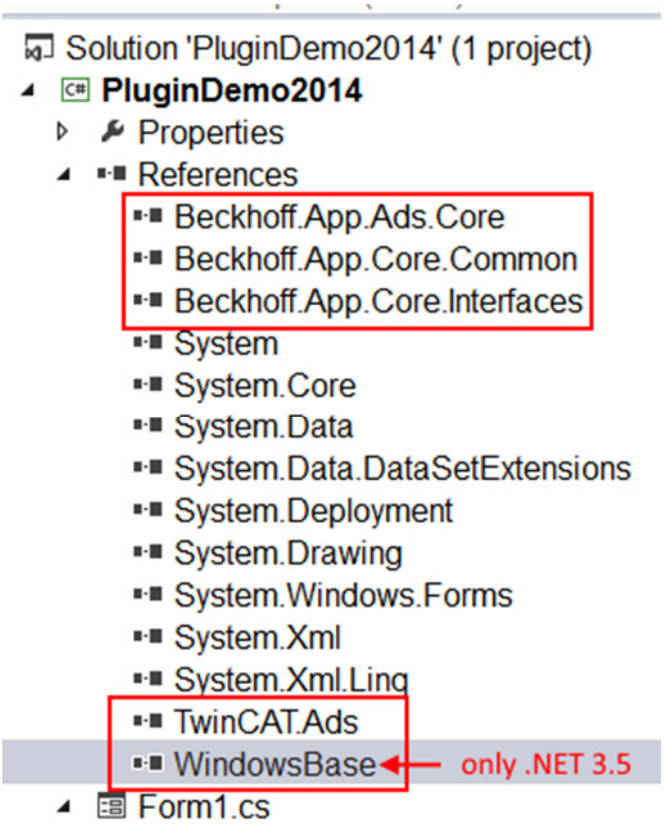
  ```
  _cncClient = adsServer.GetAdsClient<IBAAdsCncClient>("CNC");
  ```

Der Name „CNC“ ist im Hauptrahmen festgelegt und identifiziert den in den Settings aktuell eingestellten Client zum CNC Zugriff.

Benötigte Referenzen

Um den ADS Wrapper nutzen zu können, ist das Einbinden der folgenden Dlls nötig:

Name der Dll	Ablageort in der HMI
`Beckhoff.App.Ads.Core.dll`	`References`
`Beckhoff.App.Core.Interfaces.dll`	`References`
`Beckhoff.App.Core.Common.dll`	`References`
`TwinCAT.Ads.dll`	`externalReferences`
`WindowsBase.dll (nur bei .Net 3.5)`	



PLCClient synchron lesen und schreiben

Über das Interface „IBAAdsPlcClient“ kann lesend und schreibend synchron auf SPS Variablen zugegriffen werden.
Beispiel:

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

Hier wird zuerst geprüft, ob es die Variable „Global_HMI.bToggle“ in der SPS gibt und falls ja wird der aktuelle Wert als BOOL gelesen und dann invertiert geschrieben.

PLCClient OnChangeNotification

Mit Hilfe der Methode „AddPlcDeviceNotification“ ist es möglich eine Verbindung zu einer SPS Variablen mit einer Callback Methode aufzubauen.

Beispiel:

```
.
.
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.label12.BackColor = b ? Color.Green : Color.Red;
        }
    }
}
```

Hier wird eine „OnChangeDeviceNotification“ erzeugt. Bei jeder Änderung des Inhalts der Variablen, wird nun die Callback Methode aufgerufen.
Um Resourcen zu sparen empfiehlt es sich mit der Methode „RemovePlcDeviceNotification“ Notifications wieder aus dem System zu entfernen, wenn diese nicht mehr benötigt werden. (Zum Beispiel, wenn ein Formular Visible==false bekommt)

CNCClient

Im CNCClient liegen die Informationen zur CNC in der Klasse AdsCncData.
Um diesen Client nutzen zu können, muss in den Settings die Einstellung „General / newAdsClient“ „CNC“ auf „true“ gesetzt werden, damit er beim Start der HMI erzeugt wird.
Im Client im Property AdsCncData gibt es eine „AxisList“ und eine „ChannelList“.
In der AxisList sind alle Achsen des Systems in einer Collection zusammengefasst.
In der Channellist sieht man alle konfigurierten Kanäle, in der sich wiederum eine ChannelAxisList befindet, die die Achsen in dem betroffen Kanal enthalten.

Sämtliche Eigenschaften und Listen (Properties und Collections) implementieren das 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

Mit „INotifyProprtyChanged“ kann man eventbasierte Implementierungen realisieren.
Beispiel:

```
.
_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 Beispiel:

```
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);
                }
            }
        }

    }
}
```