IP-S7-Link .Net Advanced COM Code Snippets

TRAEGER.DE INDUSTRY COMPONENTS

The following code snippets imply that you are using the Helper.bas file contained in the samples included within the installation of the library.

Device Provider

Use one of the different PLC device providers by instantiating an instance of the appropriate PlcDevice class derivates.

```
Dim device As PlcDevice
Set device = New_SiemensDevice(New_IPDeviceEndPoint("192.168.0.80"))
```

Device Configuration

After an instance of (for e.g.) the SiemensDevice class has been created just use the properties provided by the class to setup the appropriate device metadata.

```
Dim device As SiemensDevice
Set device = New_SiemensDevice(New_IPDeviceEndPoint("192.168.0.80"))

device.Type = SiemensDeviceType_S71200
device.ChannelType = SiemensChannelType_OperationPanel
```

In the most scenarios its already enough to create a device using the appropriate device type in the constructor of the device, because the default channel does mostly match the common needs.

```
Dim device As SiemensDevice
Set device = New_SiemensDevice(New_IPDeviceEndPoint("192.168.0.80"),
SiemensDeviceType_S71200)
```

Device End Point

The framework does provide the ability to define different types of end points through the PlcDeviceEndPoint class. The most common end point implementation to use is the IPDeviceEndPoint class. Using this class it is possible to specify the IP address and optionally the rack and slot number of the PLC.

```
Dim endPoint As PlcDeviceEndPoint
Set endPoint = New_IPDeviceEndPoint("192.168.0.80")

' Alternatively also specify the rack.
Dim endPointWithRack As PlcDeviceEndPoint
Set endPointWithRack = New_IPDeviceEndPoint("192.168.0.80", 0)

' Alternatively also specify the rack and slot.
Dim endPointWithRackSlot As PlcDeviceEndPoint
Set endPointWithRackSlot = New_IPDeviceEndPoint("192.168.0.80", 0, 2)
```

Device Connection

After creating an instance of one of the PlcDevice class derivates just create a new connection associated with the device represented. Creating an instance using this factory method will then return the device provider dependent implementation of the PlcDeviceConnection class.

```
Dim device As PlcDevice
Set device = New_SiemensDevice(New_IPDeviceEndPoint("192.168.0.80"))

Dim connection As PlcDeviceConnection
Set connection = device.CreateConnection()
```

Handle Connection State

To retrieve the current connectivity state of the PlcDeviceConnection class instance use the State property to get the according PlcDeviceConnectionState enumeration member.

```
If (connection.State = PlcDeviceConnectionState_Connected) Then
    ' ...
End If
```

To handle the state transitions use one or more of the state specific events of the PlcDeviceConnection class instance.

```
Private WithEvents m_connection As PlcDeviceConnection
```

To handle the event declare the matching event handler as follows:

```
Private Sub m_connection_Connected(ByRef sender As Object, ByVal e As IComEventArgs)
    Debug.Print "Connection established!"
End Sub
```

Read Values

To read a single value use one of the Read methods of the PlcDeviceConnection class.

```
Dim value As Long
value = connection.ReadInt32("DB1.DBD 1")
```

To read an array of values use the additional read overloads to specify the number of items to read as an array as follows:

```
Dim values() As Long
values = connection.ReadInt32Array("DB1.DBD 1", 3)
```

Write Values

To write a single value use one of the Write methods of the PlcDeviceConnection class.

```
Call connection.WriteInt32("DB1.DBD 1", 123)
```

To write an array of values use the additional write overloads to specify an array of items to write as follows:

```
Dim values(3) As Long
values(0) = 123
values(1) = 456
values(2) = 789

Call connection.WriteInt32Array("DB1.DBD 1", values)
```

Table of Contents

Device Provider 1
Device Configuration 1
Device End Point 1
Device Connection 2
Handle Connection State 2
Read Values 3
Write Values 3