WIDESEAWCS_Communicator.DLL使用说明文档

一、简介

本文档旨在为开发者提供关于如何使用C#封装的`WCS`（Warehouse Control System，仓库控制系统）工业通讯模块的详细指导。该模块封装了与`WCS`系统通讯的底层细节，提供了一系列易于使用的接口，以便开发者在C#应用程序中高效地与`WCS`系统进行数据交换。

二、SDK版本及依赖

- **SDK版本：**.NET6及以上。
- **第三方类库**：`HslCommunication`。
- **DLL文件：**`WIDESEAWCS_Communicator.dll`

- **NuGet包地址：** http://115.159.85.185:8070/
- **当前最新版本：** `2.2.0`

三、DLL文件结构

`WIDESEAWCS_Communicator.dll`包含以下主要类和接口：

1. `BaseCommunicator`：基础抽象类，通讯基类，由子类去实现该类的属性和方法。
2. `CommunicationException`：工业通信异常类，封装了与`PLC`通信时可能发生的异常，继承`System.Exception`。
3. `SiemensDBDataType`：`西门子PLC`的数据类型类，封装了`西门子PLC`的数据类型常量。
4. `SiemensS7Communicator`：`西门子S7`通讯类，用于使用`S7`协议的`西门子PLC`，负责建立连接、读取数据、写入数据等。
5. `InovanceTcpCommunicator`：`汇川ModbusTcp`通讯类，用于使用`ModbusTcp`协议的`汇川PLC`，负责建立连接、读取数据、写入数据等。
6. `ModbusTcpCommunicator`：`ModbusTcp`通讯类，用于使用`ModbusTcp`协议的`PLC`，负责与`PLC`建立连接、读取数据、写入数据等。
7. `OmronEtherNetCommunicator`：`欧姆龙EtherNet/IP(CIP)`通讯类，用于使用`EtherNet/IP(CIP)`协议的欧姆龙`PLC`，负责与`PLC`建立连接、读取数据、写入数据等。
8. `AllenBrandlyEtherNetCommunicator(2.2.0)`：`罗克韦尔(AB)EtherNet/IP(CIP)`通讯类，用于使用`EtherNet/IP(CIP)`协议的罗克韦尔(AB)`PLC`，负责与`PLC`建立连接、读取数据、写入数据等。

四、类与接口详细描述

1、`BaseCommunicator`--通讯基类

> **Caution**
>
> 注意：该类属于抽象类，不可被实例化。

- **命名空间**：`WIDESEAWCS_Communicator`。

1.1、属性

- `LogNet`：`HslCommunication.LogNet.ILogNet`类型，日志记录实例对象，用于记录与`PLC`通讯时的异常。
- `Name`：`string`类型，`PLC`名称。
- `IsConnected`：`bool`类型，表示当前是否与`PLC`建立了连接。

1.2、方法

1.2.1、`BaseCommunicator.Connect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来创建与`PLC`的通讯连接。

1.2.2、`BaseCommunicator.Disconnect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来断开与`PLC`的通讯连接。

1.2.3、`BaseCommunicator.Read(string address, int length)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `length(int)`：要读取的数据长度。

- 返回类型：`byte[]`
- 描述：从`PLC`读取数据。

1.2.4、`BaseCommunicator.Read<T>(string address)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `T(泛型)`：读取数据的类型。

- 返回类型：`T`。
- 描述：从`PLC`读取数据。

1.2.5、`BaseCommunicator.ReadAsObj(string address, string dataType)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `dataType(string )`：要读取的数据类型。

- 返回类型：`object`
- 描述：从`PLC`读取数据。

1.2.6、`BaseCommunicator.Write(string address, byte[] data)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `data(byte[])`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据。

1.2.7、`BaseCommunicator.Write<T>(string address, T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据。

- 返回类型：`bool`

- 描述：从`PLC`读取数据。

1.2.8、`BaseCommunicator.WriteObj(string address, string dataType, object value)`

- 参数：
  - `address(string)`：要写入数据的`PLC`地址。
  - `dataType(string)`：要写入的数据类型。
  - `value(object)`：要写入的数据。
- 返回类型：`bool`
- 描述：从`PLC`读取数据。

1.2.9、`BaseCommunicator.Wait<T>`

- 参数：
  - `T(泛型)`：指定的值的类型泛型。
  - `address(string)`：要读取数据的`PLC`地址。
  - `readInterval(int)`：读取的间隔时间。
  - `waitTimeout(int)`：等待的超时时间，如果超时时间为-1的话，则是无期限等待。
  - `value(string)`：等待检测的值。
- 返回类型：`OperateResult<TimeSpan>`
- 描述：等待指定地址的泛型类型值为指定的值。

1.2.10、`BaseCommunicator.ReadCustomer<T>(string address)`

- 参数：
  - `T(泛型)`：读取数据的类型。
  - `address(string)`：要读取数据的`PLC`起始地址。
- 返回类型：`bool`
- 描述：读取自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，返回一个新的类型的实例对象。

1.2.11、`BaseCommunicator.WriteCustomer<T>(string address, [NotNull] T value)`

- 参数：
  - `T(泛型)`：写入数据的类型。
  - `address(string)`：要写入数据的`PLC`地址。
  - `value(T)`：要写入的数据。
- 返回类型：`bool`
- 描述：写入自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数。

1.2.12、`BaseCommunicator.Dispose()`

- 参数：无
- 返回类型：`void`
- 描述：释放对象资源的接口。

2、`CommunicationException`--自定义通讯异常类

> **Note**
>
> 继承`System.Exception`基类。

- **命名空间**：`WIDESEAWCS_Communicator`。

2.1、属性

- `ErrorCode`：`int?`类型，自定义通讯异常代码。
- `ErrorType`：`WIDESEAWCS_Communicator.CommunicationErrorType`类型，通讯异常的类型。
- `Message`：`string`类型，重写父类属性，自定义通讯异常信息。

2.2、方法

- `ToString()`：提供一个更友好的字符串表示形式，包含所有内部异常信息。

3、`SiemensS7Communicator`--西门子S7通讯类

> **Note**
>
> 西门子S7通讯类，继承`BaseCommunicator`基类，重写父类`BaseCommunicator`里面的抽象方法及属性，可被实例化，并可以用父类`BaseCommunicator`变量接收实例化的值。
>
> **构造函数参数：**
>
> 1. `ipAddress(string)`：IP地址。
> 2. `port(int)`：端口。
> 3. `name(string)`：`PLC`名称。

- **命名空间：**`WIDESEAWCS_Communicator`。

3.1属性

- `LogNet`：`HslCommunication.LogNet.ILogNet`类型，日志记录实例对象，用于记录与`PLC`通讯时的异常，继承重写父类`BaseCommunicator`属性。
- `Name`：`string`类型，`PLC`名称，继承重写父类`BaseCommunicator`属性。
- `IsConnected`：`bool`类型，表示当前是否与`PLC`建立了连接，继承重写父类`BaseCommunicator`属性。

3.2方法

3.2.1、`SiemensS7Communicator.Connect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来创建与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

3.2.2、`SiemensS7Communicator.Disconnect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来断开与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

3.2.3、`SiemensS7Communicator.Read(string address, int length)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `length(int)`：要读取的数据长度。

- 返回类型：`byte[]`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

3.2.4、`SiemensS7Communicator.Read<T>(string address)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `T(泛型)`：读取数据的类型。

- 返回类型：`T`。
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

3.2.5、`SiemensS7Communicator.ReadAsObj(string address, string dataType)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `dataType(string )`：要读取的数据类型。

- 返回类型：`object`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

3.2.6、`SiemensS7Communicator.Write(string address, byte[] data)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `data(byte[])`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

3.2.7、`SiemensS7Communicator.Write<T>(string address, T value)`

- 参数：
  - `T(泛型)`：写入数据的类型。
  - `address(string)`：要写入数据的`PLC`地址。
  - `value(T)`：要写入的数据。
- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

3.2.8、`SiemensS7Communicator.WriteObj(string address, string dataType, object value)`

- 参数：
  - `address(string)`：要写入数据的`PLC`地址。
  - `dataType(string)`：要写入的数据类型。
  - `value(object)`：要写入的数据。
- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

3.2.9、`SiemensS7Communicator.Wait<T>`

- 参数：
  - `T(泛型)`：指定的值的类型泛型。
  - `address(string)`：要读取数据的`PLC`地址。
  - `readInterval(int)`：读取的间隔时间。
  - `waitTimeout(int)`：等待的超时时间，如果超时时间为-1的话，则是无期限等待。
  - `value(string)`：等待检测的值。
- 返回类型：`OperateResult<TimeSpan>`
- 描述：等待指定地址的泛型类型值为指定的值，继承并重写父类`BaseCommunicator`方法。

3.2.10、`SiemensS7Communicator.ReadCustomer<T>(string address)`

- 参数：
  - `T(泛型)`：读取数据的类型。
  - `address(string)`：要读取数据的`PLC`起始地址。
- 返回类型：`bool`
- 描述：读取自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，返回一个新的类型的实例对象，继承并重写父类`BaseCommunicator`方法。

3.2.11、`SiemensS7Communicator.WriteCustomer<T>(string address, [NotNull] T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据，继承并重写父类`BaseCommunicator`方法。

- 返回类型：`bool`
- 描述：写入自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，继承并重写父类`BaseCommunicator`方法。

3.2.12、`SiemensS7Communicator.Dispose()`

- 参数：无
- 返回类型：`void`
- 描述：释放对象资源的接口，继承并重写父类`BaseCommunicator`方法。

4、`InovanceTcpCommunicator`--汇川`ModbusTcp`通讯类

> **Note**
>
> 汇川`ModbusTcp`通讯类，继承`BaseCommunicator`基类，重写父类`BaseCommunicator`里面的抽象方法及属性，可被实例化，并可以用父类`BaseCommunicator`变量接收实例化的值。
>
> **构造函数参数：**
>
> 1. `ipAddress(string)`：IP地址。
> 2. `port(int)`：端口。
> 3. `name(string)`：`PLC`名称。

- **命名空间：**`WIDESEAWCS_Communicator`。

4.1属性

- `LogNet`：`HslCommunication.LogNet.ILogNet`类型，日志记录实例对象，用于记录与`PLC`通讯时的异常，继承重写父类`BaseCommunicator`属性。
- `Name`：`string`类型，`PLC`名称，继承重写父类`BaseCommunicator`属性。
- `IsConnected`：`bool`类型，表示当前是否与`PLC`建立了连接，继承重写父类`BaseCommunicator`属性。

4.2方法

4.2.1、`InovanceTcpCommunicator.Connect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来创建与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

4.2.2、`InovanceTcpCommunicator.Disconnect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来断开与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

4.2.3、`InovanceTcpCommunicator.Read(string address, int length)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `length(int)`：要读取的数据长度。

- 返回类型：`byte[]`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

4.2.4、`InovanceTcpCommunicator.Read<T>(string address)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `T(泛型)`：读取数据的类型。

- 返回类型：`T`。
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

4.2.5、`InovanceTcpCommunicator.ReadAsObj(string address, string dataType)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `dataType(string )`：要读取的数据类型。

- 返回类型：`object`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

4.2.6、`InovanceTcpCommunicator.Write(string address, byte[] data)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `data(byte[])`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

4.2.7、`InovanceTcpCommunicator.Write<T>(string address, T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据。

- 返回类型：`bool`

- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

4.2.8、`InovanceTcpCommunicator.WriteObj(string address, string dataType, object value)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `dataType(string)`：要写入的数据类型。

  `value(object)`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

4.2.9、`InovanceTcpCommunicator.Wait<T>`

- 参数：

  `T(泛型)`：指定的值的类型泛型。

  `address(string)`：要读取数据的`PLC`地址。

  `readInterval(int)`：读取的间隔时间。

  `waitTimeout(int)`：等待的超时时间，如果超时时间为-1的话，则是无期限等待。

  `value(string)`：等待检测的值。

- 返回类型：`OperateResult<TimeSpan>`
- 描述：等待指定地址的泛型类型值为指定的值，继承并重写父类`BaseCommunicator`方法。

4.2.10、`InovanceTcpCommunicator.ReadCustomer<T>(string address)`

- 参数：

  `T(泛型)`：读取数据的类型。

  `address(string)`：要读取数据的`PLC`起始地址。

- 返回类型：`bool`
- 描述：读取自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，返回一个新的类型的实例对象，继承并重写父类`BaseCommunicator`方法。

4.2.11、`InovanceTcpCommunicator.WriteCustomer<T>(string address, [NotNull] T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据，继承并重写父类`BaseCommunicator`方法。

- 返回类型：`bool`

- 描述：写入自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，继承并重写父类`BaseCommunicator`方法。

4.2.12、`InovanceTcpCommunicator.Dispose()`

- 参数：无
- 返回类型：`void`
- 描述：释放对象资源的接口，继承并重写父类`BaseCommunicator`方法。

5、`ModbusTcpCommunicator`--`ModbusTcp`通讯类

> **Note**
>
> `ModbusTcp`通讯类，继承`BaseCommunicator`基类，重写父类`BaseCommunicator`里面的抽象方法及属性，可被实例化，并可以用父类`BaseCommunicator`变量接收实例化的值。
>
> **构造函数参数：**
>
> 1. `ipAddress(string)`：IP地址。
> 2. `port(int)`：端口。
> 3. `name(string)`：`PLC`名称。

- **命名空间：**`WIDESEAWCS_Communicator`。

5.1属性

- `LogNet`：`HslCommunication.LogNet.ILogNet`类型，日志记录实例对象，用于记录与`PLC`通讯时的异常，继承重写父类`BaseCommunicator`属性。
- `Name`：`string`类型，`PLC`名称，继承重写父类`BaseCommunicator`属性。
- `IsConnected`：`bool`类型，表示当前是否与`PLC`建立了连接，继承重写父类`BaseCommunicator`属性。

5.2方法

5.2.1、`ModbusTcpCommunicator.Connect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来创建与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

5.2.2、`ModbusTcpCommunicator.Disconnect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来断开与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

5.2.3、`ModbusTcpCommunicator.Read(string address, int length)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `length(int)`：要读取的数据长度。

- 返回类型：`byte[]`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

5.2.4、`ModbusTcpCommunicator.Read<T>(string address)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `T(泛型)`：读取数据的类型。

- 返回类型：`T`。
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

5.2.5、`ModbusTcpCommunicator.ReadAsObj(string address, string dataType)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `dataType(string )`：要读取的数据类型。

- 返回类型：`object`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

5.2.6、`ModbusTcpCommunicator.Write(string address, byte[] data)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `data(byte[])`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

5.2.7、`ModbusTcpCommunicator.Write<T>(string address, T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

5.2.8、`ModbusTcpCommunicator.WriteObj(string address, string dataType, object value)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `dataType(string)`：要写入的数据类型。

  `value(object)`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

5.2.9、`ModbusTcpCommunicator.Wait<T>`

- 参数：

  `T(泛型)`：指定的值的类型泛型。

  `address(string)`：要读取数据的`PLC`地址。

  `readInterval(int)`：读取的间隔时间。

  `waitTimeout(int)`：等待的超时时间，如果超时时间为-1的话，则是无期限等待。

  `value(string)`：等待检测的值。

- 返回类型：`OperateResult<TimeSpan>`
- 描述：等待指定地址的泛型类型值为指定的值，继承并重写父类`BaseCommunicator`方法。

5.2.10、`ModbusTcpCommunicator.ReadCustomer<T>(string address)`

- 参数：

  `T(泛型)`：读取数据的类型。

  `address(string)`：要读取数据的`PLC`起始地址。

- 返回类型：`bool`
- 描述：读取自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，返回一个新的类型的实例对象，继承并重写父类`BaseCommunicator`方法。

5.2.11、`ModbusTcpCommunicator.WriteCustomer<T>(string address, [NotNull] T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据，继承并重写父类`BaseCommunicator`方法。

- 返回类型：`bool`
- 描述：写入自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，继承并重写父类`BaseCommunicator`方法。

5.2.12、`ModbusTcpCommunicator.Dispose()`

- 参数：无
- 返回类型：`void`
- 描述：释放对象资源的接口，继承并重写父类`BaseCommunicator`方法。

6、`OmronEtherNetCommunicator`--欧姆龙`EtherNet/IP(CIP)`通讯类

> **Note**
>
> 欧姆龙EtherNet/IP(CIP)通讯类，继承`BaseCommunicator`基类，重写父类`BaseCommunicator`里面的抽象方法及属性，可被实例化，并可以用父类`BaseCommunicator`变量接收实例化的值。
>
> **构造函数参数：**
>
> 1. `ipAddress(string)`：IP地址。
> 2. `port(int)`：端口。
> 3. `name(string)`：`PLC`名称。

- **命名空间：**`WIDESEAWCS_Communicator`。

6.1属性

- `LogNet`：`HslCommunication.LogNet.ILogNet`类型，日志记录实例对象，用于记录与`PLC`通讯时的异常，继承重写父类`BaseCommunicator`属性。
- `Name`：`string`类型，`PLC`名称，继承重写父类`BaseCommunicator`属性。
- `IsConnected`：`bool`类型，表示当前是否与`PLC`建立了连接，继承重写父类`BaseCommunicator`属性。

6.2方法

6.2.1、`OmronEtherNetCommunicator.Connect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来创建与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

6.2.2、`OmronEtherNetCommunicator.Disconnect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来断开与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

6.2.3、`OmronEtherNetCommunicator.Read(string address, int length)`

- 参数：
  - `address(string)`：要读取数据的`PLC`地址。
  - `length(int)`：要读取的数据长度。
- 返回类型：`byte[]`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.4、`OmronEtherNetCommunicator.Read<T>(string address)`

- 参数：
  - `address(string)`：要读取数据的`PLC`地址。
  - `T(泛型)`：读取数据的类型。
- 返回类型：`T`。
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.5、`OmronEtherNetCommunicator.ReadAsObj(string address, string dataType)`

- 参数：
  - `address(string)`：要读取数据的`PLC`地址。
  - `dataType(string )`：要读取的数据类型。
- 返回类型：`object`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.6、`OmronEtherNetCommunicator.Write(string address, byte[] data)`

- 参数：
  - `address(string)`：要写入数据的`PLC`地址。
  - `data(byte[])`：要写入的数据。
- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.7、`OmronEtherNetCommunicator.Write<T>(string address, T value)`

- 参数：
  - `T(泛型)`：写入数据的类型。
  - `address(string)`：要写入数据的`PLC`地址。
  - `value(T)`：要写入的数据。
- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.8、`OmronEtherNetCommunicator.WriteObj(string address, string dataType, object value)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `dataType(string)`：要写入的数据类型。

  `value(object)`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.9、`OmronEtherNetCommunicator.Wait<T>`

- 参数：

  `T(泛型)`：指定的值的类型泛型。

  `address(string)`：要读取数据的`PLC`地址。

  `readInterval(int)`：读取的间隔时间。

  `waitTimeout(int)`：等待的超时时间，如果超时时间为-1的话，则是无期限等待。

  `value(string)`：等待检测的值。

- 返回类型：`OperateResult<TimeSpan>`
- 描述：等待指定地址的泛型类型值为指定的值，继承并重写父类`BaseCommunicator`方法。

6.2.10、`OmronEtherNetCommunicator.ReadCustomer<T>(string address)`

- 参数：

  `T(泛型)`：读取数据的类型。

  `address(string)`：要读取数据的`PLC`起始地址。

- 返回类型：`bool`
- 描述：读取自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，返回一个新的类型的实例对象，继承并重写父类`BaseCommunicator`方法。

6.2.11、`OmronEtherNetCommunicator.WriteCustomer<T>(string address, [NotNull] T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据，继承并重写父类`BaseCommunicator`方法。

- 返回类型：`bool`
- 描述：写入自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，继承并重写父类`BaseCommunicator`方法。

6.2.12、`OmronEtherNetCommunicator.Dispose()`

- 参数：无
- 返回类型：`void`
- 描述：释放对象资源的接口，继承并重写父类`BaseCommunicator`方法。

7、`AllenBrandlyEtherNetCommunicator`--罗克韦尔(AB)CIP通讯类(`2.2.1`)

> **Note**
>
> `罗克韦尔(AB)EtherNet/IP(CIP)`通讯类，继承`BaseCommunicator`基类，重写父类`BaseCommunicator`里面的抽象方法及属性，可被实例化，并可以用父类`BaseCommunicator`变量接收实例化的值。
>
> **构造函数参数：**
>
> 1. `ipAddress(string)`：IP地址。
> 2. `port(int)`：端口。
> 3. `name(string)`：`PLC`名称。

- **命名空间：**`WIDESEAWCS_Communicator`。

7.1属性

- `LogNet`：`HslCommunication.LogNet.ILogNet`类型，日志记录实例对象，用于记录与`PLC`通讯时的异常，继承重写父类`BaseCommunicator`属性。
- `Name`：`string`类型，`PLC`名称，继承重写父类`BaseCommunicator`属性。
- `IsConnected`：`bool`类型，表示当前是否与`PLC`建立了连接，继承重写父类`BaseCommunicator`属性。

6.2方法

7.2.1、`AllenBrandlyEtherNetCommunicator.Connect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来创建与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

7.2.2、`AllenBrandlyEtherNetCommunicator.Disconnect()`

- 参数：无
- 返回类型：`bool`
- 描述：用来断开与`PLC`的通讯连接，继承并重写父类`BaseCommunicator`方法。

7.2.3、`AllenBrandlyEtherNetCommunicator.Read(string address, int length)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `length(int)`：要读取的数据长度。

- 返回类型：`byte[]`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

7.2.4、`AllenBrandlyEtherNetCommunicator.Read<T>(string address)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `T(泛型)`：读取数据的类型。

- 返回类型：`T`。
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

6.2.5、`AllenBrandlyEtherNetCommunicator.ReadAsObj(string address, string dataType)`

- 参数：

  `address(string)`：要读取数据的`PLC`地址。

  `dataType(string )`：要读取的数据类型。

- 返回类型：`object`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

7.2.6、`AllenBrandlyEtherNetCommunicator.Write(string address, byte[] data)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `data(byte[])`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

7.2.7、`AllenBrandlyEtherNetCommunicator.Write<T>(string address, T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

7.2.8、`AllenBrandlyEtherNetCommunicator.WriteObj(string address, string dataType, object value)`

- 参数：

  `address(string)`：要写入数据的`PLC`地址。

  `dataType(string)`：要写入的数据类型。

  `value(object)`：要写入的数据。

- 返回类型：`bool`
- 描述：从`PLC`读取数据，继承并重写父类`BaseCommunicator`方法。

7.2.9、`AllenBrandlyEtherNetCommunicator.Wait<T>`

- 参数：

  `T(泛型)`：指定的值的类型泛型。

  `address(string)`：要读取数据的`PLC`地址。

  `readInterval(int)`：读取的间隔时间。

  `waitTimeout(int)`：等待的超时时间，如果超时时间为-1的话，则是无期限等待。

  `value(string)`：等待检测的值。

- 返回类型：`OperateResult<TimeSpan>`
- 描述：等待指定地址的泛型类型值为指定的值，继承并重写父类`BaseCommunicator`方法。

7.2.10、`AllenBrandlyEtherNetCommunicator.ReadCustomer<T>(string address)`

- 参数：

  `T(泛型)`：读取数据的类型。

  `address(string)`：要读取数据的`PLC`起始地址。

- 返回类型：`bool`
- 描述：读取自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，返回一个新的类型的实例对象，继承并重写父类`BaseCommunicator`方法。

7.2.11、`AllenBrandlyEtherNetCommunicator.WriteCustomer<T>(string address, [NotNull] T value)`

- 参数：

  `T(泛型)`：写入数据的类型。

  `address(string)`：要写入数据的`PLC`地址。

  `value(T)`：要写入的数据，继承并重写父类`BaseCommunicator`方法。

- 返回类型：`bool`
- 描述：写入自定义的数据类型，需要继承自`HslCommunication.IDataTransfer`接口，并且有无参数构造函数，继承并重写父类`BaseCommunicator`方法。

7.2.12、`AllenBrandlyEtherNetCommunicator.Dispose()`

- 参数：无
- 返回类型：`void`
- 描述：释放对象资源的接口，继承并重写父类`BaseCommunicator`方法。

五、枚举类型

`CommunicationErrorType`：通信错误类型枚举。

> **Note**
>
> **命名空间**：`WIDESEAWCS_Communicator`。

- `IpAddressError`：IP地址转换错误。
- `ConnectionFailed`：连接PLC错误。
- `Unknown`：未知类型错误。
- `ReadFailed`：读取PLC数据失败。
- `WriteFailed`：写入PLC数据失败。
- `TypeError`：数据类型转换错误。
- `ReadException`：读取PLC数据异常。

六、常量

`SiemensDBDataType`：西门子PLC的数据类型。

> **Note**
>
> **命名空间**：`WIDESEAWCS_Communicator`。

- `DataType_DInt(string)`：`西门子PLC`中`dint`类型，对应C#中32位有符号整型(`int`)，常量值为`dint`。
- `DataType_Bool(string)`：`西门子PLC`中`bool`类型，对应C#中布尔类型(`bool`)，常量值为`bool`。
- `DataType_String(string)`：`西门子PLC`中`string`类型，对应C#中字符串类型(`string`)，常量值为`string`。
- `DataType_Int(string)`：`西门子PLC`中`int`类型，对应C#中16位有符号整型(`short`)，常量值为`int`。
- `DataType_Byte(string)`：`西门子PLC`中`byte`类型，对应C#中字节类型(`byte`)，常量值为`byte`。
- `DataType_DW(string)`：`西门子PLC`中`dw`类型，对应C#中32位无符号整型(`uint`)，常量值为`dw`。
- `DataType_W(string)`：`西门子PLC`中`w`类型，对应C#中16位无符号整型(`ushort`)，常量值为`w`。
- `DataType_Float(string)`：`西门子PLC`中`float`类型，对应C#中单精度浮点型(`float`)，常量值为`float`。
- `DataType_Char(string)`：`西门子PLC`中`char`类型，对应C#中32位有符号整型(`char`)，常量值为`char`。

七、使用示例

1、`BaseCommunicator`及其子类

1.1实例化示例

> **Caution**
>
> 因`BaseCommunicator`为虚拟类，不可被实例化，所以需实例化子类，用将子类实例化的对象赋值给父类`BaseCommunicator`(这里的示例子类使用`西门子S7`通讯类[`SiemensS7Communicator`]，其他继承重写的子类与该示例一致)。

```
//实例化对象，通过实例化赋值ip与端口。
WIDESEAWCS_Communicator.BaseCommunicator communicator = new
WIDESEAWCS_Communicator.SiemensS7Communicator("127.0.0.1", 102, "测试PLC");
```

1.2方法示例

> **Note**
>
> **注意：方法示例使用上面实例化的对象举例，其他继承重写的子类使用与该示例一致。**
>
> **建议读写PLC数据使用`Read<T>(string address)`和`Write<T>(string address, T value)`方法。**

```
//实例化后ip与端口已赋值，所以在使用的时候可以直接调用连接PLC的方法
communicator.Connect();//连接PLC

bool resutl = communicator.Disconnect();//断开与工业设备的连接。

byte[] result = communicator.Read("DB1.0", 10);//从PLC读取数据。返回读取的字节数组

int result = communicator.Read<int>("DB1.0");//从PLC读取数据。返回读取到的泛型结果

object result = communicator.ReadAsObj("DB1.0",
WIDESEAWCS_Communicator.SiemensDBDataType.DataType_DInt);//从PLC读取数据。返回object类型结果

bool result = communicator.Write("DB1.0", new byte[] { 1, 1 });//向PLC写入字节数组数据。

bool result = communicator.Write<int>("DB1.0", 1);//向PLC写入指定类型数据。

bool result = communicator.WriteObj("DB1.0",
WIDESEAWCS_Communicator.SiemensDBDataType.DataType_DInt, 1);//向PLC写入指定的PLC数据类型数据。

OperateResult<TimeSpan> result = communicator.Wait("DB1.0", 500, 10 * 6000, 1);//等待指定地址的泛型类型值为指定的值，一旦通信失败，或是等待超时就返回失败。否则返回成功，并告知调用方等待时长

ConveyorLineTaskCommand result =
communicator.ReadCustomer<ConveyorLineTaskCommand>("DB1.0");//读取自定义的数据类型，返回一个新的类型的实例对象。
```

```
bool result = communicator.WriteCustomer<ConveyorLineTaskCommand>("DB1.0", new ConveyorLineTaskCommand());//读取自定义的数据类型，写入自定义类型的数据。
```

1.3WCS重构代码中截图示例

- 实例化对象

```
// 加载程序集
Assembly assembly = Assembly.Load($"WIDESEAWCS_Communicator");
// 获取类型
Type? type = assembly.GetType($"WIDESEAWCS_Communicator.{x.DevicePlcType}");
// 创建实例
object? obj = Activator.CreateInstance(type, new object[] { x.DeviceIp, x.DevicePort, x.DeviceName });
// 调用连接方法
bool? connectResult = (bool)type.InvokeMember("Connect", BindingFlags.Default | BindingFlags.InvokeMethod, null, obj, new object[] { });
// 判断连接结果
if (connectResult ?? false) ConsoleHelper.WriteSuccessLine(x.DeviceCode + "连接成功"); else ConsoleHelper.WriteErrorLine(x.DeviceCode + "连接失败");
```

> **Note**
>
> 注意：代码中使用是通过反射`Activator.CreateInstance()`去实例化对象，开发使用时可以通过上面`实例化示例`去实例化对象，也可以通过反射方式实例化对象。

- 读取数据

```
public TRsult GetValue<TEnum, TRsult>(TEnum value) where TEnum : Enum
{
    if (!IsConnected) throw new Exception($"通讯连接错误，请检查网络");
    DeviceProDTO? devicePro = _deviceProDTOs.FirstOrDefault(x => x.DeviceProParamName == value.ToString()) ?? throw new Exception($"读取数据错误,未在协议信息里面找到参数{value.ToString()}");
    return (TRsult)Communicator.ReadAsObj(devicePro.DeviceProAddress, devicePro.DeviceDataType);
}
```

> **Note**

- 写入数据

```
public bool SetValue<TEnum, TValue>(TEnum @enum, TValue value)
    where TEnum : Enum
    where TValue : notnull
{
    if (!IsConnected) throw new Exception($"通讯连接错误，请检查网络");
    DeviceProDTO? devicePro = _deviceProDTOs.FirstOrDefault(x => x.DeviceProParamName == @enum.ToString()) ?? throw new Exception($"写入数据错误,未在协议信息里面找到参数{value.ToString()}");
    return Communicator.WriteObj(devicePro.DeviceProAddress, devicePro.DeviceDataType, value);
}
```

八、异常处理与日志记录

1、异常处理

1.1自定义异常

1.1.1、`Ip地址错误`：连接PLC时检查IP抛出的异常。

> **Note**
>
> **实例化时会抛出此异常**
>
> 1、检查数据库IP是否配置正确。
>
> 2、实例化时IP传值是否正确。

1.1.2、`数据读取失败`：使用`HslCommunication`读取数据时返回失败抛出的异常

> **Note**
>
> **调用`ReadAsObj(string address, string dataType)`、`Read<T>(string address)`、`Read(string address, int length)`读取数据错误时会抛出此异常，具体原因需根据错误信息定位。**

1、有可能由于网络问题导致读取失败。

2、读取的`PLC`地址或数据类型错误引起的失败。

1.1.3、`数据写入失败`：使用`HslCommunication`写入数据时返回失败抛出的异常

> **Note**
>
> **调用`Write(string address, byte[] data)`、`Write<T>(string address, T value)`、`WriteObj(string address, string dataType, [NotNull] object value)`写入数据错误时会抛出此异常，具体原因需根据错误信息定位。**
>
> 1、有可能由于网络问题导致读取失败。
>
> 2、读取的`PLC`地址或数据类型错误引起的失败。

1.1.4、`数据类型转换错误`：数据类型转换错误

> **Note**
>
> **调用`Write<T>(string address, T value)`、`WriteObj(string address, string dataType, [NotNull] object value)`写入数据错误时会抛出此异常。**
>
> 1、**数据类型转换失败。**如写入“a”(类型为字符串)的时候调用`WriteObj(string address, string dataType, [NotNull] object value)`方法，参数`dataType`给的是数值型(如：`int16`、`int32`、`float`等)时会出现数据类型转换错误异常。

1.1.5、`数据写入后读取校验失败`：将数据写入到PLC后再读取的数据与写入的数据不一致时抛出的异常

> **Note**
>
> **调用`Write(string address, byte[] data)`、`Write<T>(string address, T value)`、`WriteObj(string address, string dataType, [NotNull] object value)`写入数据校验错误时会抛出此异常。**
>
> 1、该地址写入数据后是否被其他地方覆盖。
>
> 2、PLC程序没有清除或者覆盖该数据。

1.1.6、`数据读取错误`：读取数据代码执行时的异常

> **Note**
>
> **调用`ReadAsObj(string address, string dataType)`、`Read<T>(string address)`、`Read(string address, int length)`读取数据错误时会抛出此异常。**

1.1.7、`PLC连接失败`

> **Note**
>
> **调用`Connect()`连接PLC时会抛出此异常。**
>
> 1、通讯异常，网络不通导致连接不上。
>
> 2、数据配置错误。

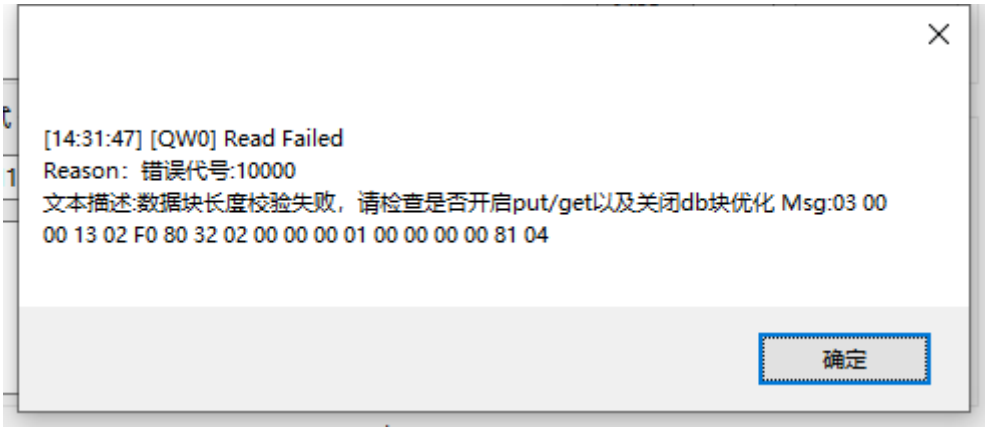
1.1.8、数据类型错误：PLC未定义该类型数据

> **Note**
>
> PLC未定义该类型数据

1.2 PLC配置异常

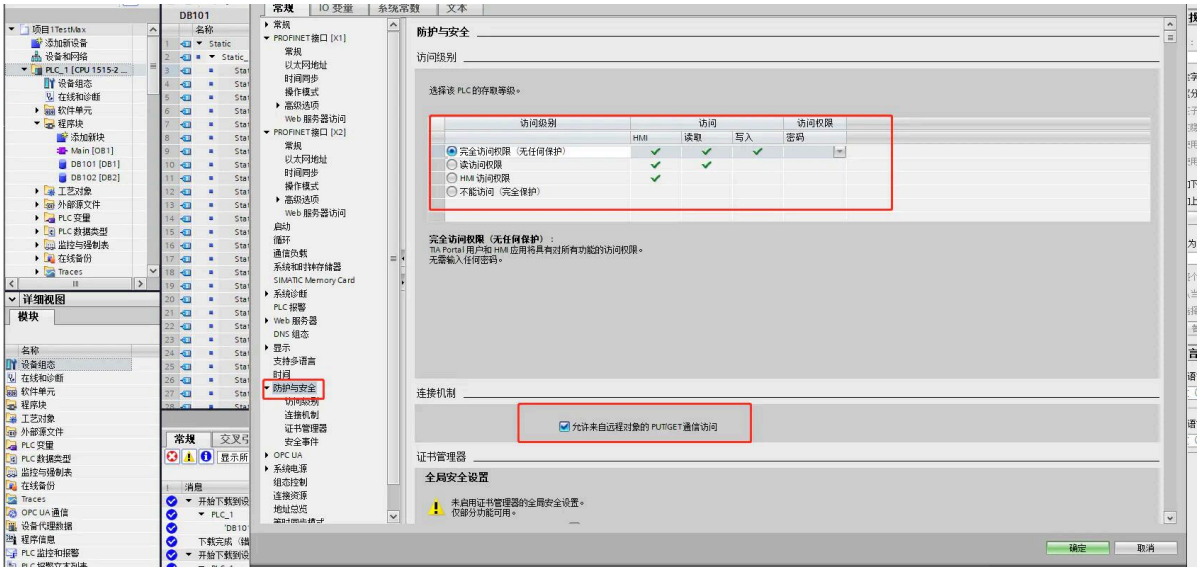
1.2.1西门子连接初始化配置

Siemens 的 102 通常是默认就开放的，如果你在使用demo测试时出现了如下的错误问题。



那么大概率是如下的问题，只要配置后把参数下载到PLC，断电重启下，基本上可以解决问题。

> **Caution**
>
> **注意：该配置是需要在PLC中配置。**



连接参数设置

通常情况下来说，对于S1200，S1500，200smart直接默认连接就可以了，不需要额外的设置参数信息。

对于 S300 而言，通常的情况是： Rack 为 0， Slot 为 2

对于 S400 而言， Rack 和 Slot 就要看PLC实际的情况来确认了。

对于 S200 而言，需要关注 LocalTSAP 和 DestTSAP 就要看PLC实际的情况来确认了。

> **Caution**

> 注意：`S1200`，`S1500`，`200smart`指的是通常情况下不用额外设置参数，如果`PLC`设置了，需要跟`PLC`保持一致

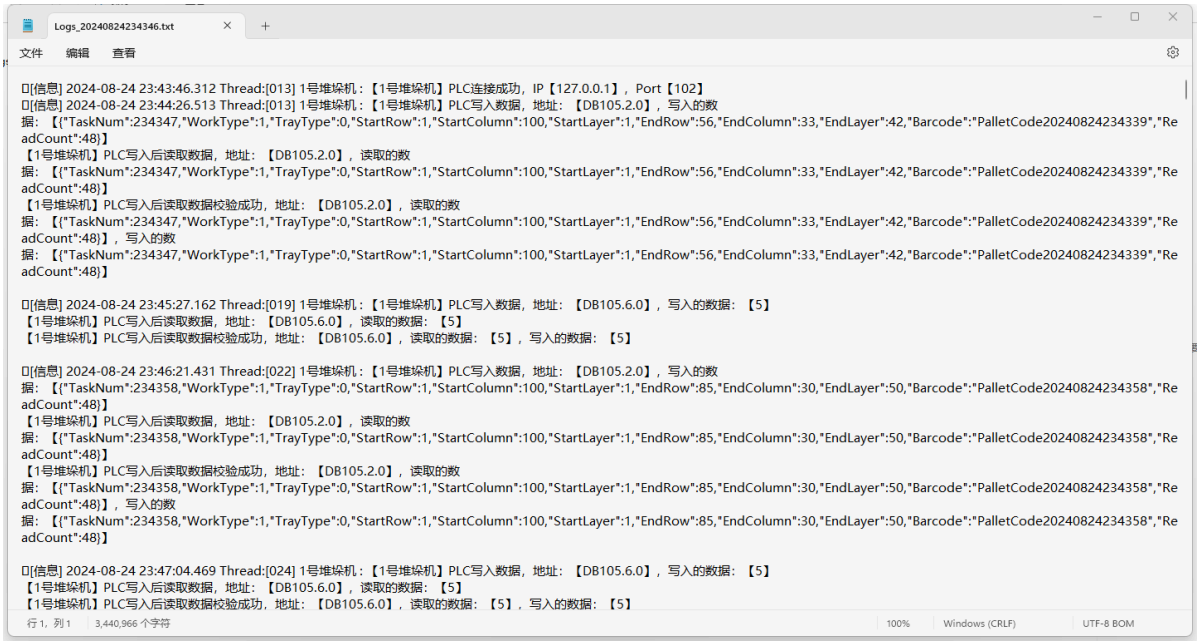
1.3其他异常

> **Caution**
>
> **遇到其他异常需联系开发或者维护人员。**

2、日志记录

> **Important**
>
> 通讯日志记录在文本文档文件中，**路径在程序根目录下`Log_PLCReadWrite`文件夹的对应`PLC`名称文件夹中。**
>
> **文件格式为`.txt`，根据文件大小自动分文件存储，文件最大为10M，最多存储100个文件。**

- **日志记录示例**



九、维护与更新

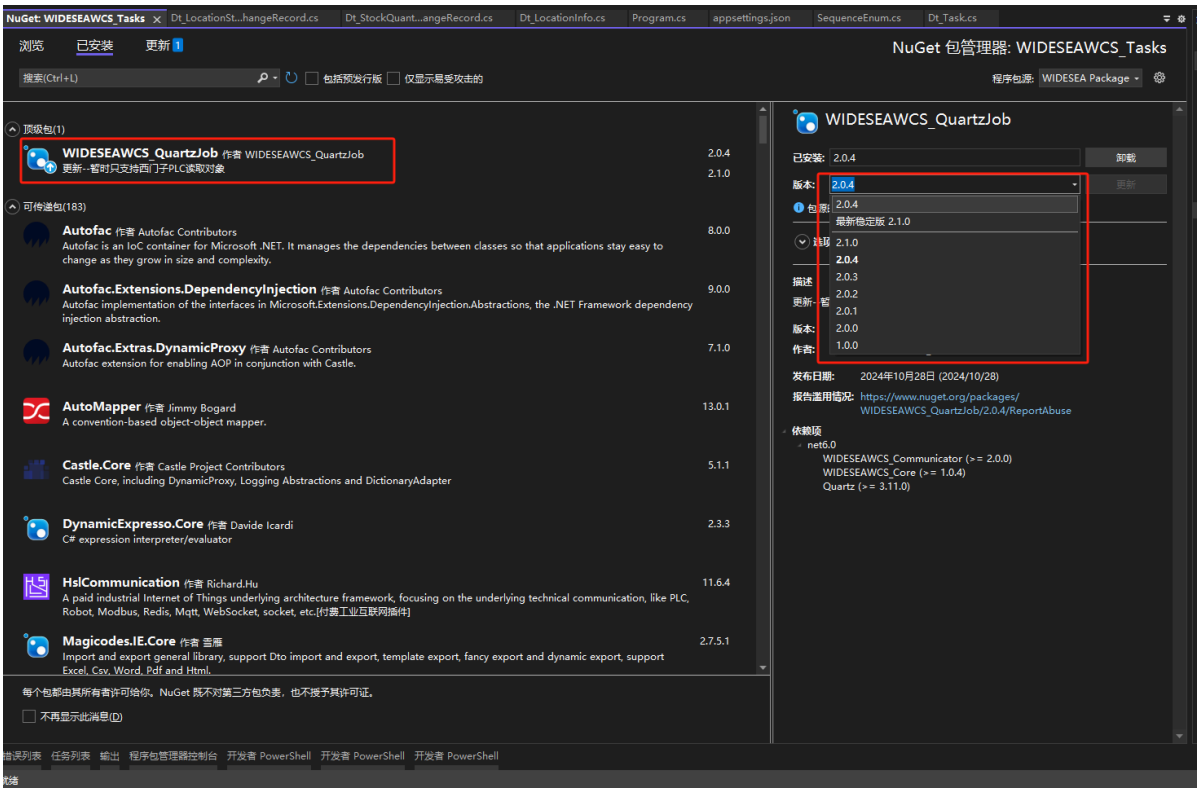
> **Note**
>
> **本文档将随着WIDESEAWCS_Communicator.dll的更新而不断更新。如有任何疑问、建议或发现错误，请联系DLL的维护人员。**

1、更新通知

为了保持WIDESEAWCS_Communicator.dll的稳定性和功能性，会不定期发布更新。这些更新可能包含新功能、性能改进、安全修复或兼容性调整。更新通知会发在微信群。

2、更新方法

在使用该DLL的项目代码中，去NuGet包管理器中更新版本，选择指定的版本，点击更新按钮。



3、兼容性

在更新WIDESEAWCS_Communicator.dll时，会尽力保持与旧版本的兼容性。然而，由于某些更新可能包含重大更改或新功能，因此在新版本中可能会引入与旧版本不兼容的变化。在更新之前，请仔细阅读更新日志，了解新版本中的更改和潜在的不兼容性问题。

> **Caution**
>
> **注意：如果遇到不兼容的情况，请及时回退版本或者联系开发和维护人员。**

4、版本记录

版本号	发布日期	主要更改	新增功能	修复的问题	兼容性
2.1.0	2024/11/1	新增欧姆龙EtherNet/IP(CIP)通讯	新增欧姆龙EtherNet/IP(CIP)通讯	/	.NET6及以上版本
2.2.1	2024/11/8	新增罗克韦尔(AB)EtherNet/IP(CIP)通讯	新增罗克韦尔(AB)EtherNet/IP(CIP)通讯	/	.NET6及以上版本，兼容以前版本

十、附录