VisionKit.VisionKitHelper.ConnectChengeEven += VisionKitHelper\_ConnectChengeEven

这段 C# 代码的功能如下：

事件订阅：代码行中的 += 操作符用于订阅事件。

事件来源：事件 ConnectChengeEven 来自 VisionKit.VisionKitHelper 类。

事件处理程序：VisionKitHelper\_ConnectChengeEven 是一个方法，当 ConnectChengeEven 事件被触发时，该方法会被调用执行。

功能描述：通常此类事件用于监听连接状态的变化，当连接状态发生改变时，会触发相应的处理逻辑。

简而言之，这行代码的作用是监听 VisionKit.VisionKitHelper 的连接状态变化，并在状态改变时执行 VisionKitHelper\_ConnectChengeEven 方法。

泛型

MyGenericArray<int> intArray = new MyGenericArray<int>(5);

该代码实例化了一个名为 intArray 的泛型类 MyGenericArray<int> 对象，指定其元素类型为 int，并设置初始容量为 5。这表示 intArray 将用于存储整数，并且初始可以容纳 5 个元素。

T表示泛型类型，意味着该函数可以用于任何类型的数据。

元素类型可以使用自定义的类类型。只要确保在使用自定义类类型作为泛型参数时，该类已经定义并且适用于泛型类所要求的操作，例如：

C:\Users\106407\AppData\Local\Temp\企业微信截图_17260326619506.png

**C#中的Action**

C#中的Action是一种委托类型，‌用于引用不返回值的方法。‌Action可以接受0到16个参数，‌并且不返回任何值。‌它是一种通用的委托类型，‌非常方便用于处理不同参数和不同函数签名的情况。‌Action的用法包括声明Action委托类型、‌创建Action实例并赋值给委托变量、‌以及调用Action。‌

声明Action委托类型：‌可以通过Action关键字声明一个Action类型的变量，‌例如Action action;。‌如果需要指定参数类型，‌可以使用Action<T>，‌其中T是参数的类型，‌例如Action<int> actionWithInt;或Action<string, int> actionWithStringAndInt;。‌

创建Action实例并赋值给委托变量：‌可以使用Lambda表达式创建Action实例并赋值给委托变量。‌例如，‌创建一个不接受参数的Action可以通过action = () => { Console.WriteLine("Action without parameters"); };来实现。‌同样地，‌带有参数的Action可以通过类似的方式创建，‌例如actionWithInt = (x) => { Console.WriteLine($"Action with int parameter: {x}"); };。‌

调用Action：‌创建的Action可以通过直接调用其引用的方法名来执行，‌例如action();会执行Lambda表达式中定义的代码块，‌输出"Action without parameters"。‌对于带有参数的Action，‌可以通过传递参数来调用，‌例如actionWithInt(10);会输出"Action with int parameter: 10"。‌

Action提供了一种灵活且类型安全的方式来封装和传递无返回值的方法，‌并且可以通过Lambda表达式方便地创建匿名方法作为参数传递。‌这使得Action在C#编程中非常有用，‌可以用于执行不返回值的任何操作，‌如打印消息、‌更新状态等

**Action**

* **Action 表示一个无返回值的方法**，并且可以接受任意数量和类型的参数（从0到16个）。
* **使用场景：只需要执行一段代码而不需要关心其返回值时**，例如用于事件处理器、回调方法、一次性任务等。
* 示例：

public delegate void Action();

public delegate void Action<T1>(T1 arg1);

public delegate void Action<T1, T2>(T1 arg1, T2 arg2);

// 使用Action的例子

Action sayHello = () => Console.WriteLine("Hello, 鸡哥!");

sayHello(); // 输出 "Hello, 鸡哥!"

//两种写法都彳亍

Action<int> printNumber = new Action<int>((number) =>

Console.WriteLine($"输出数字为: {number}"));

printNumber(114514); // 输出 "输出数字为: 114514"