AT.Toolkit.WindowDataBind 二次开发教程

📖 概述

AT.Toolkit.WindowDataBind 是一个数据窗口绑定工具，用于在生产检测环境中实时显示工位数据、产品统计、OK/NG率等信息。本教程详细介绍如何在二次开发中调用该模块的接口。

目录

第一章：获取模块实例

1.1 功能说明

在使用数据窗口绑定工具前，需要先通过容器获取实例。

1.2 代码示例

```
using AT.Librarys.Container;
using AT.Librarys.Container.Toolkit.Data;

/// <summary>
/// 数据窗口绑定工具使用示例
/// </summary>
public class WindowDataBindExample
{
    // 数据窗口绑定实例
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 初始化数据窗口绑定工具
    /// </summary>
    /// <param name="instanceName">实例名称，如"检测数据窗口"</param>
    /// <returns>是否初始化成功</returns>
    public bool Initialize(string instanceName)
    {
        // 通过容器获取已创建的实例
        _windowDataBind = DS.Container.GetToolkit<IWindowDataBind>(instanceName);

        // 验证实例是否获取成功
        if (_windowDataBind == null)
        {
            DS.NotifyG.Error($"获取数据窗口绑定失败: {instanceName}");
            return false;
        }

        DS.NotifyG.Info($"数据窗口绑定初始化成功: {instanceName}");
        return true;
    }
}
```

第二章：更新测量数据

2.1 功能说明

将测量工位的检测结果更新到数据窗口，自动统计OK/NG。

2.2 代码示例

```
using AT.Librarys.Container.Station;
using AT.Librarys.Container.Toolkit.Data;
using System;
using System.Collections.Generic;

/// <summary>
/// 更新测量数据示例
/// </summary>
public class MeasureDataExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 更新单个测量结果
    /// </summary>
    /// <param name="sn">产品条码</param>
    /// <param name="thickness">厚度值</param>
    /// <param name="thicknessOK">厚度是否OK</param>
    public void UpdateMeasureResult(string sn, double thickness, bool thicknessOK)
    {
        // 创建测量结果对象
        MeasureResult result = new MeasureResult()
        {
            // 工位名称（对应配置中的工位）
            StationName = "检测工位1",

            // 产品条码
            SN = sn,

            // 检测时间
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),

            // 总体结果
            IsSucceed = thicknessOK,

            // 信息提示
            Message = thicknessOK ? "" : "厚度超差"
        };

        // 添加测量项
        result.Items = new List<MeasureResultItem>()
        {
            new MeasureResultItem()
            {
                // 测量项名称（对应配置中的工位项）
                ValueName = "厚度",

                // 测量值
                Value = thickness.ToString("F3"),
```

```
                // 是否OK
                IsOK = thicknessOK
            }
        };

        // 更新到数据窗口
        _windowDataBind.Udpate(result);
    }

    /// <summary>
    /// 更新多项测量结果
    /// </summary>
    /// <param name="sn">产品条码</param>
    /// <param name="measurements">测量数据字典：名称->值</param>
    /// <param name="okResults">OK结果字典：名称->是否OK</param>
    public void UpdateMultipleMeasureResults(string sn,
        Dictionary<string, double> measurements,
        Dictionary<string, bool> okResults)
    {
        // 创建测量结果对象
        MeasureResult result = new MeasureResult()
        {
            StationName = "检测工位1",
            SN = sn,
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
            Items = new List<MeasureResultItem>()
        };

        // 总体结果：所有项都OK才算OK
        bool allOK = true;

        // 遍历添加所有测量项
        foreach (var item in measurements)
        {
            bool isOK = okResults.ContainsKey(item.Key) ? okResults[item.Key] : true;
            if (!isOK) allOK = false;

            result.Items.Add(new MeasureResultItem()
            {
                ValueName = item.Key,              // 测量项名称
                Value = item.Value.ToString("F3"), // 测量值
                IsOK = isOK                        // 是否OK
            });
        }

        // 设置总体结果
        result.IsSucceed = allOK;
        result.Message = allOK ? "" : "检测NG";

        // 更新到数据窗口
        _windowDataBind.Udpate(result);
    }

    /// <summary>
    /// 带上下限判断的测量结果更新
    /// </summary>
    /// <param name="sn">产品条码</param>
```

```
    /// <param name="itemName">测量项名称</param>
    /// <param name="value">测量值</param>
    /// <param name="upperLimit">上限</param>
    /// <param name="lowerLimit">下限</param>
    public void UpdateMeasureWithLimit(string sn, string itemName,
        double value, double upperLimit, double lowerLimit)
    {
        // 判断是否在公差范围内
        bool isOK = (value >= lowerLimit) && (value <= upperLimit);

        // 创建结果
        MeasureResult result = new MeasureResult()
        {
            StationName = "检测工位1",
            SN = sn,
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
            IsSucceed = isOK,
            Message = isOK ? "" : $"{itemName}超差: {value:F3}",
            Items = new List<MeasureResultItem>()
            {
                new MeasureResultItem()
                {
                    ValueName = itemName,
                    Value = value.ToString("F3"),
                    IsOK = isOK
                }
            }
        };

        // 更新
        _windowDataBind.Udpate(result);
    }
}
```

第三章：更新定位数据

3.1 功能说明

将定位工位的视觉定位结果更新到数据窗口。

3.2 代码示例

```
using AT.Librarys.Container.Station;
using AT.Librarys.Container.Toolkit.Data;
```

```
using System;
using System.Collections.Generic;

/// <summary>
/// 更新定位数据示例
/// </summary>
public class PositionDataExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 更新定位结果
    /// </summary>
    /// <param name="sn">产品条码</param>
    /// <param name="x">X坐标</param>
    /// <param name="y">Y坐标</param>
    /// <param name="angle">角度</param>
    /// <param name="isOK">是否成功</param>
    public void UpdatePositionResult(string sn, double x, double y, double angle, bool isOK)
    {
        // 创建定位结果对象
        PositionResult result = new PositionResult()
        {
            // 工位名称
            StationName = "定位工位1",

            // 产品条码
            SN = sn,

            // 定位时间
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),

            // 定位是否成功
            IsSucceed = isOK,

            // 信息提示
            Message = isOK ? "" : "定位失败"
        };

        // 添加定位项
        result.Items = new List<PositionResultItem>()
        {
            new PositionResultItem()
            {
                // 定位项名称
                ValueName = "产品定位",

                // 是否成功
                IsSucceed = isOK,

                // 相机坐标
                Camera = new PositionPoint()
                {
                    Point = new CPoint(y, x),  // Row=Y, Column=X
                    Angle = angle
                },
```

```
                // 机械坐标（如果已标定）
                Machine = new PositionPoint()
                {
                    Point = new CPoint(y, x),
                    Angle = angle
                },

                // 偏移量
                Offset = new PositionPoint()
                {
                    Point = new CPoint(0, 0),
                    Angle = 0
                }
            }
        };

        // 更新到数据窗口
        _windowDataBind.Udpate(result);
    }

    /// <summary>
    /// 更新多个定位点结果
    /// </summary>
    public void UpdateMultiplePositions(string sn, List<(string name, double x, double y, double angle,
    {
        PositionResult result = new PositionResult()
        {
            StationName = "定位工位1",
            SN = sn,
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
            Items = new List<PositionResultItem>()
        };

        bool allOK = true;

        foreach (var pos in positions)
        {
            if (!pos.ok) allOK = false;

            result.Items.Add(new PositionResultItem()
            {
                ValueName = pos.name,
                IsSucceed = pos.ok,
                Camera = new PositionPoint()
                {
                    Point = new CPoint(pos.y, pos.x),
                    Angle = pos.angle
                }
            });
        }

        result.IsSucceed = allOK;
        result.Message = allOK ? "" : "定位失败";

        _windowDataBind.Udpate(result);
    }
}
```

第四章：更新扫码数据

4.1 功能说明

将扫码工位的扫码结果更新到数据窗口。

4.2 代码示例

```
using AT.Librarys.Container.Station;
using AT.Librarys.Container.Toolkit.Data;
using System;
using System.Collections.Generic;

/// <summary>
/// 更新扫码数据示例
/// </summary>
public class ScanCodeDataExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 更新扫码结果
    /// </summary>
    /// <param name="sn">扫描到的条码</param>
    /// <param name="isOK">扫码是否成功</param>
    public void UpdateScanCodeResult(string sn, bool isOK)
    {
        // 创建扫码结果对象
        ScanCodeResult result = new ScanCodeResult()
        {
            // 工位名称
            StationName = "扫码工位",

            // 扫描到的条码
            SN = sn,

            // 扫码时间
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),

            // 扫码是否成功
            IsSucceed = isOK,
```

```
            // 信息提示
            Message = isOK ? "" : "扫码失败"
        };

        // 添加扫码项
        result.Items = new List<ScanCodeResultItem>()
        {
            new ScanCodeResultItem()
            {
                // 扫码项名称
                ValueName = "产品条码",

                // 扫码值
                Value = sn,

                // 是否成功
                IsSucceed = isOK,

                // 信息
                Message = isOK ? "" : "无法识别"
            }
        };

        // 更新到数据窗口
        _windowDataBind.Udpate(result);
    }

    /// <summary>
    /// 更新多个条码扫描结果
    /// </summary>
    /// <param name="mainSN">主条码</param>
    /// <param name="codes">其他条码列表</param>
    public void UpdateMultipleCodes(string mainSN, Dictionary<string, string> codes)
    {
        ScanCodeResult result = new ScanCodeResult()
        {
            StationName = "扫码工位",
            SN = mainSN,
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
            Items = new List<ScanCodeResultItem>()
        };

        bool allOK = true;

        foreach (var code in codes)
        {
            bool isOK = !string.IsNullOrEmpty(code.Value);
            if (!isOK) allOK = false;

            result.Items.Add(new ScanCodeResultItem()
            {
                ValueName = code.Key,    // 条码名称
                Value = code.Value,      // 条码值
                IsSucceed = isOK
            });
        }
```

```
        result.IsSucceed = allOK;
        result.Message = allOK ? "" : "部分条码扫描失败";

        _windowDataBind.Udpate(result);
    }
}
```

第五章：更新仪表数据

5.1 功能说明

将仪表检测工位的测量结果更新到数据窗口。

5.2 代码示例

```
using AT.Librarys.Container.Station;
using AT.Librarys.Container.Toolkit.Data;
using System;
using System.Collections.Generic;

/// <summary>
/// 更新仪表数据示例
/// </summary>
public class InstrumentDataExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 更新仪表测量结果
    /// </summary>
    /// <param name="sn">产品条码</param>
    /// <param name="itemName">测量项名称</param>
    /// <param name="value">测量值</param>
    /// <param name="isOK">是否OK</param>
    public void UpdateInstrumentResult(string sn, string itemName, double value, bool isOK)
    {
        // 创建仪表结果对象
        InstrumentResult result = new InstrumentResult()
        {
            // 工位名称
            StationName = "仪表检测工位",

            // 产品条码
            SN = sn,
```

```
            // 检测时间
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),

            // 检测是否成功
            IsSucceed = isOK,

            // 信息提示
            Message = isOK ? "" : $"{itemName}超差"
        };

        // 添加测量项
        result.Items = new List<InstrumentResultItem>()
        {
            new InstrumentResultItem()
            {
                // 测量项名称
                ValueName = itemName,

                // 测量值
                Value = value.ToString("F3"),

                // 是否OK
                IsOK = isOK,

                // 信息
                Message = isOK ? "" : "超差"
            }
        };

        // 更新到数据窗口
        _windowDataBind.Udpate(result);
    }

    /// <summary>
    /// 更新电阻测量结果
    /// </summary>
    public void UpdateResistanceResult(string sn, double resistance, double limit)
    {
        bool isOK = resistance <= limit;

        InstrumentResult result = new InstrumentResult()
        {
            StationName = "电阻测试工位",
            SN = sn,
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
            IsSucceed = isOK,
            Message = isOK ? "" : $"电阻超标: {resistance:F3}Ω > {limit:F3}Ω",
            Items = new List<InstrumentResultItem>()
            {
                new InstrumentResultItem()
                {
                    ValueName = "绝缘电阻",
                    Value = resistance.ToString("F3"),
                    IsOK = isOK,
                    Message = isOK ? "PASS" : "FAIL"
                }
```

```
            }
        };

        _windowDataBind.Udpate(result);
    }
}
```

第六章：直接写入数据

6.1 功能说明

直接向指定工位和工位项写入数据值，适用于简单的单值写入场景。

6.2 代码示例

```
using AT.Librarys.Container.Toolkit.Data;

/// <summary>
/// 直接写入数据示例
/// </summary>
public class WriteValueExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 直接写入单个值
    /// </summary>
    /// <param name="stationName">工位名称（行）</param>
    /// <param name="itemName">工位项名称（列）</param>
    /// <param name="sn">产品条码</param>
    /// <param name="value">值</param>
    /// <param name="isOK">是否OK</param>
    /// <param name="message">信息</param>
    public void WriteValue(string stationName, string itemName,
        string sn, string value, bool isOK, string message)
    {
        // 直接写入值到数据窗口
        _windowDataBind.WriteValue(
            stationName,    // 工位名称，如 "检测工位1"
            itemName,       // 工位项名称，如 "厚度"
            sn,             // 产品条码
            value,          // 值，如 "10.235"
            isOK,           // 是否OK
            message         // 信息，如 "" 或 "超差"
```

```
        );
    }

    /// <summary>
    /// 写入测量值示例
    /// </summary>
    public void WriteMeasureValue(string sn, double thickness, double width, double height)
    {
        // 写入厚度
        bool thicknessOK = thickness >= 9.5 && thickness <= 10.5;
        _windowDataBind.WriteValue("检测工位1", "厚度", sn,
            thickness.ToString("F3"), thicknessOK, thicknessOK ? "" : "厚度超差");

        // 写入宽度
        bool widthOK = width >= 4.8 && width <= 5.2;
        _windowDataBind.WriteValue("检测工位1", "宽度", sn,
            width.ToString("F3"), widthOK, widthOK ? "" : "宽度超差");

        // 写入高度
        bool heightOK = height >= 2.9 && height <= 3.1;
        _windowDataBind.WriteValue("检测工位1", "高度", sn,
            height.ToString("F3"), heightOK, heightOK ? "" : "高度超差");
    }

    /// <summary>
    /// 写入参数（上下限）
    /// </summary>
    public void WriteParameter()
    {
        // 写入上限
        _windowDataBind.WritePara("上限", "厚度", "10.5");

        // 写入下限
        _windowDataBind.WritePara("下限", "厚度", "9.5");
    }
}
```

第七章：清除统计数据

7.1 功能说明

清除OK/NG统计计数，重新开始统计。

7.2 代码示例

```
using AT.Librarys.Container.Toolkit.Data;

/// <summary>
/// 清除统计数据示例
/// </summary>
public class ClearCountExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 清除所有工位的统计
    /// </summary>
    public void ClearAllCount()
    {
        // 清除所有统计数据
        _windowDataBind.ClearCount();

        DS.NotifyG.Info("已清除所有统计");
    }

    /// <summary>
    /// 清除指定工位的统计
    /// </summary>
    /// <param name="stationName">工位名称</param>
    public void ClearStationCount(string stationName)
    {
        // 清除指定工位的统计
        _windowDataBind.ClearCount(stationName);

        DS.NotifyG.Info($"已清除工位 [{stationName}] 的统计");
    }

    /// <summary>
    /// 换型时清除统计
    /// </summary>
    public void OnProductChange()
    {
        // 换型时清除所有统计
        _windowDataBind.ClearCount();

        // 刷新上下限显示
        _windowDataBind.GetStationLimit();
        _windowDataBind.ShowLimitValue();

        DS.NotifyG.Info("产品切换完成，统计已重置");
    }
}
```

第八章：订阅事件

8.1 功能说明

订阅数据窗口的事件，实现数据联动和扩展功能。

8.2 代码示例

```
using AT.Librarys.Container.Toolkit.Data;
using System;
using System.Collections.Generic;

/// <summary>
/// 订阅事件示例
/// </summary>
public class EventSubscribeExample
{
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 订阅所有事件
    /// </summary>
    public void SubscribeEvents()
    {
        // 订阅数据缓存变化事件
        // 当有新数据更新时触发
        _windowDataBind.DataBufferChange += OnDataBufferChange;

        // 订阅配置变化事件
        // 当配置发生变化时触发
        _windowDataBind.EventConfigChange += OnConfigChange;

        // 订阅数据输出事件
        // 当需要输出数据时触发
        _windowDataBind.DataOutputEvent += OnDataOutput;
    }

    /// <summary>
    /// 数据缓存变化事件处理
    /// </summary>
    private void OnDataBufferChange(object sender, EventArgs e)
    {
        // 当有新数据时触发
        // 可以在这里进行数据联动处理

        DS.NotifyG.Info("数据已更新");

        // 例如：刷新其他界面
        // RefreshOtherUI();
    }

    /// <summary>
    /// 配置变化事件处理
    /// </summary>
```

```
    private void OnConfigChange(object sender, EventArgs e)
    {
        // 当配置变化时触发
        // 可以在这里重新初始化相关组件

        DS.NotifyG.Info("配置已变更");
    }

    /// <summary>
    /// 数据输出事件处理
    /// </summary>
    private void OnDataOutputEvent(object sender, DataOutputArgs e)
    {
        // 获取输出数据
        string name = e.Name;
        Dictionary<string, object> data = e.Data;

        // 处理输出数据
        DS.NotifyG.Info($"数据输出: {name}");

        foreach (var item in data)
        {
            DS.NotifyG.Info($"  {item.Key} = {item.Value}");
        }

        // 例如：发送到MES系统
        // SendToMES(data);
    }

    /// <summary>
    /// 取消订阅事件
    /// </summary>
    public void UnsubscribeEvents()
    {
        _windowDataBind.DataBufferChange -= OnDataBufferChange;
        _windowDataBind.EventConfigChange -= OnConfigChange;
        _windowDataBind.DataOutputEvent -= OnDataOutput;
    }
}
```

第九章：完整应用示例

9.1 功能说明

综合使用数据窗口绑定工具的各项功能，实现完整的检测流程数据显示。

9.2 代码示例

```
using AT.Librarys.Container;
using AT.Librarys.Container.Station;
using AT.Librarys.Container.Toolkit.Data;
using System;
using System.Collections.Generic;

/// <summary>
/// 完整应用示例：产品检测流程
/// </summary>
public class CompleteExample
{
    // 数据窗口绑定实例
    private IWindowDataBind _windowDataBind;

    /// <summary>
    /// 初始化
    /// </summary>
    /// <param name="instanceName">实例名称</param>
    /// <returns>是否成功</returns>
    public bool Initialize(string instanceName)
    {
        // 获取数据窗口绑定实例
        _windowDataBind = DS.Container.GetToolkit<IWindowDataBind>(instanceName);

        if (_windowDataBind == null)
        {
            DS.NotifyG.Error($"初始化失败: {instanceName}");
            return false;
        }

        // 订阅事件
        _windowDataBind.DataBufferChange += OnDataChange;

        // 获取工位上下限
        _windowDataBind.GetStationLimit();

        return true;
    }

    /// <summary>
    /// 产品检测完成后更新数据
    /// </summary>
    /// <param name="sn">产品条码</param>
    /// <param name="measurements">测量结果</param>
    /// <param name="limits">上下限</param>
    public void OnInspectionComplete(string sn,
        Dictionary<string, double> measurements,
        Dictionary<string, (double lower, double upper)> limits)
    {
        // 创建测量结果
        MeasureResult result = new MeasureResult()
        {
            StationName = "检测工位1",
```

```
            SN = sn,
            DateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
            Items = new List<MeasureResultItem>()
        };

        bool allOK = true;
        string ngMessage = "";

        // 遍历所有测量项
        foreach (var item in measurements)
        {
            string itemName = item.Key;
            double value = item.Value;

            // 判断是否OK
            bool isOK = true;
            if (limits.ContainsKey(itemName))
            {
                var limit = limits[itemName];
                isOK = value >= limit.lower && value <= limit.upper;
            }

            if (!isOK)
            {
                allOK = false;
                ngMessage += $"{itemName}超差; ";
            }

            // 添加测量项
            result.Items.Add(new MeasureResultItem()
            {
                ValueName = itemName,
                Value = value.ToString("F3"),
                IsOK = isOK
            });
        }

        // 设置总体结果
        result.IsSucceed = allOK;
        result.Message = allOK ? "" : ngMessage.TrimEnd(' ', ';');

        // 更新到数据窗口
        _windowDataBind.Udpate(result);

        // 记录日志
        if (allOK)
            DS.NotifyG.Info($"[{sn}] 检测OK");
        else
            DS.NotifyG.Alarm($"[{sn}] 检测NG: {ngMessage}");
    }

    /// <summary>
    /// 换型处理
    /// </summary>
    /// <param name="productType">新产品型号</param>
    public void OnProductTypeChange(string productType)
    {
```

```
        // 清除统计
        _windowDataBind.ClearCount();

        // 重新获取上下限
        _windowDataBind.GetStationLimit();
        _windowDataBind.ShowLimitValue();

        // 触发配置变更
        _windowDataBind.OnEventConfigChange();

        DS.NotifyG.Info($"切换产品型号: {productType}");
    }

    /// <summary>
    /// 数据变化事件处理
    /// </summary>
    private void OnDataChange(object sender, EventArgs e)
    {
        // 数据更新后的处理
        // 例如：更新看板、发送到上位系统等
    }

    /// <summary>
    /// 清理资源
    /// </summary>
    public void Dispose()
    {
        if (_windowDataBind != null)
        {
            _windowDataBind.DataBufferChange -= OnDataChange;
        }
    }
}
```

附录A：接口定义

IWindowDataBind 接口

```
/// <summary>
/// 数据窗口绑定接口
/// </summary>
public interface IWindowDataBind : IToolkitBase
{
    // ===== 更新数据方法 =====

    /// <summary>更新扫码结果</summary>
    void Udpate(ScanCodeResult data);
```

```
    /// <summary>更新定位结果</summary>
    void Udpate(PositionResult data);

    /// <summary>更新测量结果</summary>
    void Udpate(MeasureResult data);

    /// <summary>更新仪表结果</summary>
    void Udpate(InstrumentResult data);

    /// <summary>直接写入值</summary>
    void WriteValue(string stationName, string stationItemName,
        string sn, string value, bool isOk, string message);

    // ===== 参数方法 =====

    /// <summary>写入参数</summary>
    bool WritePara(string stationName, string stationItemName, string value);

    /// <summary>获取工位上下限</summary>
    void GetStationLimit();

    /// <summary>显示上下限值</summary>
    bool ShowLimitValue();

    // ===== 清除方法 =====

    /// <summary>清除指定工位统计</summary>
    void ClearCount(string station);

    /// <summary>清除所有统计</summary>
    void ClearCount();

    // ===== 事件 =====

    /// <summary>配置变化事件</summary>
    event EventHandler EventConfigChange;

    /// <summary>数据缓存变化事件</summary>
    event EventHandler<EventArgs> DataBufferChange;

    /// <summary>数据输出事件</summary>
    event EventHandler<DataOutputArgs> DataOutputEvent;

    /// <summary>触发配置变更</summary>
    void OnEventConfigChange();
}
```

附录B：数据结构说明

测量结果 MeasureResult

属性	类型	说明
StationName	string	工位名称
SN	string	产品条码
DateTime	string	检测时间
IsSucceed	bool	是否成功
Message	string	信息提示
Items	List	测量项列表

测量项 MeasureResultItem

属性	类型	说明
ValueName	string	测量项名称
Value	string	测量值
IsOK	bool	是否OK

定位结果 PositionResult

属性	类型	说明
StationName	string	工位名称
SN	string	产品条码
DateTime	string	定位时间
IsSucceed	bool	是否成功
Message	string	信息提示
Items	List	定位项列表

扫码结果 ScanCodeResult

属性	类型	说明
StationName	string	工位名称
SN	string	扫描条码
DateTime	string	扫码时间
IsSucceed	bool	是否成功
Message	string	信息提示
Items	List	扫码项列表

仪表结果 InstrumentResult

属性	类型	说明
StationName	string	工位名称
SN	string	产品条码
DateTime	string	检测时间
IsSucceed	bool	是否成功
Message	string	信息提示
Items	List	测量项列表

技术支持

如有问题，请联系开发团队。