Devices Monitor Reference

View live device status and diagnostics.

Devices Monitor (Reference) provides real-time visibility into Device Node status, communication health, and error diagnostics during runtime operation.

The Device Monitor displays:

- Node connectivity status
- Communication errors
- Activity counters
- Invalid addresses
- Error timestamps
- Protocol-specific diagnostics

Access via **Devices** **Device Monitor** when solution is running.

Monitor Properties
Accessing Properties
Error Codes
Channel Status Monitoring
Invalid Addresses
Diagnostic Patterns
Best Practices
Troubleshooting Guide

Monitor Properties

Runtime Node Properties

Property	Description	Access
ActivityCounter	Increments on any node activity	`@Device.Node.NodeName.ActivityCounter`
Disable	Node disabled status	`@Device.Node.NodeName.Disable`
LastErrorCode	Error code (0 = success)	`@Device.Node.NodeName.LastErrorCode`
LastErrorDateTime	Timestamp of last error	`@Device.Node.NodeName.LastErrorDateTime`
InvalidAddresses	Unresolved device addresses	`@Device.Node.NodeName.InvalidAddresses`

Accessing Properties

In Scripts

csharp

```csharp
// Check node status
int errorCode = @Device.Node.PLC1.LastErrorCode;
if (errorCode != 0)
{
    DateTime errorTime = @Device.Node.PLC1.LastErrorDateTime;
    @Info.Trace($"Node error {errorCode} at {errorTime}");
}

// Monitor activity
int activity = @Device.Node.PLC1.ActivityCounter;
bool isActive = activity > 0;
```

In Displays

xml

```xml
<!-- Show node status -->
<TextBlock Text="{Device.Node.PLC1.LastErrorCode}" />
<TextBlock Text="{Device.Node.PLC1.LastErrorDateTime}" />

<!-- Activity indicator -->
<Ellipse Fill="{Binding Device.Node.PLC1.ActivityCounter,
    Converter=ActivityToColorConverter}" />
```

Error Codes

Success

- **0** - No errors, normal operation

Negative Codes (Connection Issues)

Code	Error	Description
-1	BuildCommandException	Command creation failed
-2	ParseCommandUnsolicitedException	Unexpected command received
-3	ParseReplyException	Response parsing failed
-4	BuildReplyUnsolicitedException	Unsolicited reply error
-5	ChannelException	Channel-level error
-6	NodeException	Node-level error

Network Errors (-100 to -115)

Code	Error	Description
-100	Base Send Error	Generic send failure
-101	Base SendAndWait Error	Send/wait operation failed
-102 to -104	TCP Create Error	Socket creation failed
-105	TCP Connect Callback	Connection callback error
-106	TCP Receive Error	TCP data receive failed
-107	UDP Create Error	UDP socket creation failed
-108	UDP Receive Error	UDP data receive failed
-109	Serial Create Error	Serial port initialization failed
-110	Serial Receive Error	Serial data receive failed
-111	TCP NotConnected	TCP connection lost
-112	Start Message Timeout	Message start not received
-113	Receiving Bytes Timeout	Data timeout during receive
-114	End Message Timeout	Message end not received
-115	Connect Timeout	Connection attempt timeout

Protocol Errors (-200 to -208)

Code	Error	Description
-200	ProtocolError	Generic protocol error
-201	InvalidProtocol	Protocol mismatch
-202	InvalidStation	Station address invalid
-203	InvalidCommand	Command not recognized
-204	InvalidMsgSequence	Message sequence error
-205	InvalidCheckSum	Checksum validation failed
-206	InvalidAddress	Device address invalid
-207	InvalidModifiers	Modifier configuration error
-208	ProtocolErrorCloseSocket	Protocol error requiring reconnection

Positive Codes

Protocol-specific errors - consult driver documentation

Windows socket errors are reported with their positive Windows code:

Code	Error	Description
10058	WSAESHUTDOWN	This error code suggests that a send or receive operation was attempted on a socket that has already been shut down. It can occur if the software on either the server or the device side tries to communicate over a socket that's been closed or is no longer valid.

Channel Status Monitoring

Access channel status for additional diagnostics:

csharp

```csharp
// Channel status
int channelStatus = @Device.Channel.ChannelName.Status;

// Interpret status using error code table
string statusMessage = GetStatusMessage(channelStatus);
```

Invalid Addresses

Monitor and handle unresolved addresses:

csharp

```csharp
// Get invalid addresses
string[] invalid = @Device.Node.PLC1.InvalidAddresses;

if (invalid.Length > 0)
{
    foreach(string address in invalid)
    {
        @Info.Trace($"Invalid address: {address}");
    }
}
```

Diagnostic Patterns

Connection Health Check

csharp

```csharp
public bool IsNodeHealthy(string nodeName)
{
    var node = @Device.Node[nodeName];

    return node.LastErrorCode == 0 &&
           node.ActivityCounter > 0 &&
           !node.Disable;
}
```

Error Recovery

csharp

```
public void HandleNodeError(string nodeName)
{
    var node = @Device.Node[nodeName];
    int error = node.LastErrorCode;

    switch(error)
    {
        case -111: // Not connected
            RestartChannel();
            break;
        case -115: // Timeout
            IncreaseTimeout();
            break;
        default:
            LogError(error);
            break;
    }
}
```

Best Practices

1. **Monitor Critical Nodes** - Set up alarms for errors
2. **Log Error Patterns** - Track recurring issues
3. **Handle Timeouts** - Implement retry logic
4. **Check Invalid Addresses** - Validate during startup
5. **Track Activity** - Detect stalled communications
6. **Document Error Codes** - Map to troubleshooting steps
7. **Set Up Notifications** - Alert on critical failures

Troubleshooting Guide

No Activity

- Check channel enabled
- Verify network connectivity
- Review node configuration
- Check polling settings

Frequent Errors

- Analyze error patterns
- Check network stability
- Review timeout settings
- Verify device status

Invalid Addresses

- Check address format
- Verify device mapping
- Review import results
- Test individual points

Connection Timeouts

- Increase timeout values
- Check network latency
- Verify device response
- Review firewall settings

In this section...
