Historian Monitor Reference

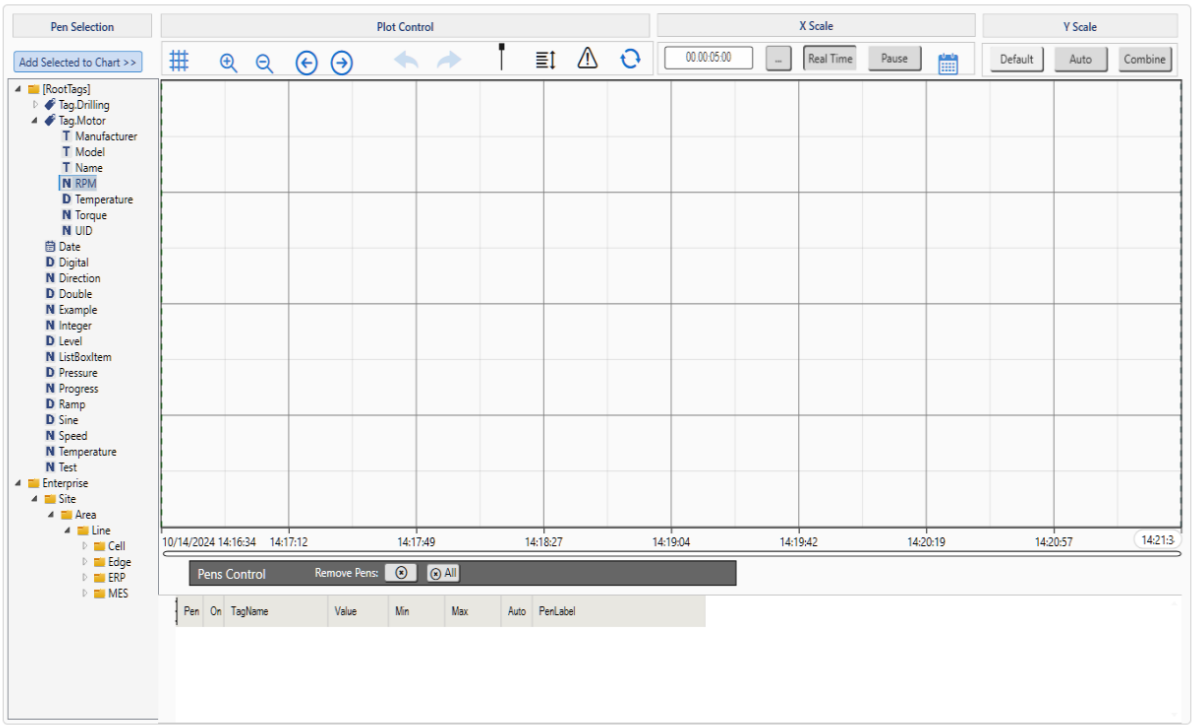
Monitor historian data collection.

Historian Monitor (Reference) provides real-time visibility into historian table operations and data collection status during runtime, it enables:

- Real-time status of historian tables
- Trend chart visualization of historical data
- Performance metrics tracking
- Error monitoring and diagnostics
- Remote access via web browser

Access the monitor at **Historian Historian Monitor** when connected to runtime.

Monitor Components
Remote Access
Monitor Table Properties
Runtime Property Access
Using Monitor Data
Troubleshooting with Monitor
Best Practices



Monitor Components

Trend Chart Page

Built-in user interface for visualizing historian data:

- Created using TrendChart Control
- Same tools available for custom displays
- Template available in Charts library
- Real-time and historical data viewing

Status Table

Runtime properties of all historian tables:

- Row counts and success metrics
- Error tracking and messages
- Last stored timestamps
- Performance indicators

Remote Access

Monitor historian status remotely without installation:

URL Format:

`http://[server]:3101/html5/?Monitor=Historian`

Parameters:

- `[server]`: IP address or hostname
- Port `3101`: Production profile (default)
- Port `3201`: Development profile

Example:

`http://192.168.1.100:3101/html5/?Monitor=Historian`

Monitor Table Properties

Column	Description	Runtime Access
Namespace	Item namespace location	-
Name	Historian table identifier	`@Historian.Table.<Name>`
RowCount	Total rows stored	`@Historian.Table.<Name>.RowCount`
SuccessCount	Successful storage operations	`@Historian.Table.<Name>.SuccessCount`
ErrorCount	Failed storage attempts	`@Historian.Table.<Name>.ErrorCount`
LastStoredTimeStamp	Most recent data timestamp	`@Historian.Table.<Name>.LastStoredTimeStamp`
LastStoredErrorMessage	Latest error description	`@Historian.Table.<Name>.LastStoredErrorMessage`

Runtime Property Access

Access monitor properties in displays or scripts:

csharp

```csharp
// Get table status
int rows = @Historian.Table.ProcessData.RowCount;
int errors = @Historian.Table.ProcessData.ErrorCount;
DateTime lastStore = @Historian.Table.ProcessData.LastStoredTimeStamp;

// Check for errors
if (@Historian.Table.ProcessData.ErrorCount > 0)
{
    string error = @Historian.Table.ProcessData.LastStoredErrorMessage;
    // Handle error condition
}
```

Using Monitor Data

Health Monitoring

csharp

```csharp
// Calculate success rate
double successRate = @Historian.Table.MyTable.SuccessCount /
                    (@Historian.Table.MyTable.SuccessCount +
                     @Historian.Table.MyTable.ErrorCount) * 100;
```

Data Validation

csharp

```
// Check data currency
TimeSpan dataAge = DateTime.Now - @Historian.Table.MyTable.LastStoredTimeStamp;
if (dataAge.TotalMinutes > 5)
{
    // Alert: Data may be stale
}
```

Troubleshooting with Monitor

Common Issues

High Error Count:

- Check storage location connectivity
- Verify database permissions
- Review error messages for patterns
- Check disk space

No Row Count Increase:

- Verify triggers are firing
- Check tag quality
- Review deadband settings
- Confirm table is enabled

Timestamp Not Updating:

- Check data collection is running
- Verify tag values are changing
- Review SaveOnChange settings
- Check time deadband

Best Practices

- [ ] **Monitor regularly** - Check error counts daily
- [ ] **Set up alerts** - Notify on error thresholds
- [ ] **Track trends** - Monitor row count growth
- [ ] **Document errors** - Keep log of issues
- [ ] **Use remote access** - Monitor from anywhere
- [ ] **Create dashboards** - Build custom monitoring displays

---

In this section...
