Trend Annotations Reference (10.1.5 draft)

Reference page for the 10.1.5 trend annotation API surface: per-chart binding, operator-safety opt-out, customer-script extension points, and the auto-created SQL table schema.

Draft preview — 10.1.5

This page documents behavior introduced in version 10.1.5. It is a draft preview, soft-launched as a child of the live *TrendChart Annotations Example* page. Promote to a top-level reference when 10.1.5 GA ships.

Overview

Annotations are timestamped operator-authored or AI-authored notes overlaid on a TrendChart pen. Use them to mark events, capture commentary on a value at a moment in time, or surface AI-generated explanations next to the data they refer to.

The annotation surface in 10.1.5 is intentionally scoped to the two time-series chart controls: the **TrendChart** and the **DrillingChart**. Other chart types (BarChart, PieChart, GanttChart, etc.) have no annotation surface by design.

Quick start (zero-config)

The simplest pattern uses the platform-default behavior: the auto-created `Annotations` SQL table inside the TagHistorian database, plus the chart's built-in operator UX.

1. On the TrendChart properties, set `AnnotationsSource = "Annotations"`.
2. Leave the four `*Method` tags (`AddNoteMethod`, `AddRangeEventMethod`, `GetAnnotationTableMethod`, `MouseClickAnnotationMethod`) empty.
3. At runtime, operators right-click the chart and choose *Add Tag Note* the platform-supplied dialog opens the row is inserted into the `Annotations` table the pin renders on the chart.

No customer-side `Script.Class` is required for this default path.

AnnotationsSource (per-chart)

The `AnnotationsSource` field on each TrendChart and DrillingChart names the Dataset.Table object that backs the annotation overlay for that chart.

```
// Per-chart Dataset.Table object name that backs the annotation overlay.
// Empty (default) preserves legacy behavior (the chart's built-in default
// falls through to nothing). Customer sets this to e.g. "Dataset.Table.Annotations"
// or just "Annotations" to wire the overlay to a SQL Dataset they manage.
// Naming is per-chart, not global.
public string AnnotationsSource = "";
```

Behavior:

- **Empty (default)** — no overlay; the chart performs no annotation work. Operator context-menu items are still visible (when `AnnotationsEnabled` is true) but yield a loud "no source configured" warning if invoked.
- **Set** — the chart resolves the named Dataset.Table at runtime, reads annotations through it for the visible time range, and writes new annotations to it through the platform-default add path (or through your `AddNoteMethod` / `AddRangeEventMethod` override, when set).

The value is per-chart on purpose. Different charts can be wired to different annotation tables, and a single solution can mix the platform-default `Annotations` table with one or more customer-managed tables on different charts.

AnnotationsEnabled (operator-safety opt-out)

Each chart has a runtime tag at `Client.TrendChart.AnnotationsEnabled` and `Client.DrillingChart.AnnotationsEnabled`. Both default to `true`.

```
// Master operator-safety opt-out for the TrendChart annotation surface.
// Defaults to true so the zero-config Historian-namespace annotation
// default works on upgrade. When set to false, all four context-menu
// items ("Add Tag Note", "Add Time Note", "Add XValue Note", "Add Event
// Range") are hidden -- both the customer-override path (when one of the
// four *Method tags is set) AND the built-in inline default fallthrough.
// Use this to opt out wholesale on kiosk / regulated / view-only
// deployments without scripting.
public bool AnnotationsEnabled = true;
```

Setting either tag to `false` hides the four annotation context-menu items and suppresses the read-back overlay on that chart kind. This is the recommended switch for kiosk, regulated, or view-only deployments where operators must not author annotations — no scripting required.

Customer-extension *Method tags

Each TrendChart and DrillingChart exposes four optional override hooks. When set, they redirect the corresponding chart action to a customer `Script.Class` method, replacing the platform default for that action only. When empty, the platform-default zero-config behavior takes over.

AddNoteMethod

```
// Full AddNoteMethod method name in "Script.Class" with Client domain.
// Ex: Script.Class.ClientMain.MyAddNoteMethod.
// Prototype:
//   public void AddNote(TTrendChart trend, string tagName, object xValue, double yValue);
// Note: tagName can be null.
```

AddRangeEventMethod

```
// Full AddRangeEventMethod method name in "Script.Class" with Client domain.
// Ex: Script.Class.ClientMain.MyAddRangeEventMethod.
// Prototype:
//   public void AddRangeEventMethod(TTrendChart trend, string[] tagNames,
//                                   DateTime start, TimeSpan duration);
```

GetAnnotationTableMethod

```
// Full GetAnnotationTableMethod method name in "Script.Class" with Client domain.
// Ex: Script.Class.ClientMain.MyGetAnnotationTable.
// Prototype:
//   public DataTable GetAnnotationTableMethod(TTrendChart trend, string[] tagNames,
//                                             object xMinValue, object xMaxValue,
//                                             bool showTagNotes, bool showTimeNotes,
//                                             bool showXValueNotes, bool showRangeEvents);
```

MouseClickAnnotationMethod

```
// Full MouseClickAnnotationMethod method name in "Script.Class" with Client domain.
// Ex: Script.Class.ClientMain.MyMouseClickAnnotationMethod.
// Prototype:
//   public void MouseClickAnnotationMethod(MouseButtonEventArgs e, bool isEvent,
//                                          string identification, string title,
//                                          string contents, string color,
//                                          object xValue, double yValue,
//                                          double duration);
```

Built-in zero-config default behavior

When `AnnotationsSource` is non-empty AND the four `*Method` tags are empty, the chart uses the platform-default add and read paths:

- **Add path** — operator picks *Add Tag Note* from the context menu the platform-supplied annotation dialog opens the resulting row is INSERTed into the resolved Dataset.Table.
- **Read path** — for each chart re-pan / refresh, the chart SELECTs annotations from the resolved Dataset.Table for the visible time range and the visible pen tags, and renders them as pins overlaid on the appropriate pen.

You can override either path independently. Setting `AddNoteMethod` alone gives you a custom add UX while keeping the platform-default read. Setting `GetAnnotationTableMethod` alone gives you a custom read query while keeping the platform-default add. Mix and match as the deployment requires.

By-design scope: TrendChart and DrillingChart only

The 10.1.5 annotation API surface applies to time-series charts — specifically the **TrendChart** and **DrillingChart** controls. Other chart kinds (BarChart, PieChart, GanttChart, scatter plots, etc.) have no annotation overlay, no `AnnotationsSource` field, and no `AnnotationsEnabled` tag. This is by design: the time-axis semantics that make annotation pinning meaningful exist only on these two chart kinds.

If you need annotation-like commentary on a non-time-series chart, the Dataset.Table the annotations live in is plain SQL — you can author your own visualization on top of it.

Annotations SQL table schema

The platform auto-creates an `Annotations` table inside the TagHistorian SQLite database the first time the Historian module starts. The table has **26 columns** covering the full annotation feature set:

```
-- Auto-created Annotations table (26 columns)
CREATE TABLE Annotations (
    ID                BIGINT           PRIMARY KEY,   -- auto-assigned
    rowguid           UNIQUEIDENTIFIER,
    DateTimeUTCTicks  BIGINT,
    DateTime          DATETIME,
    SourceID          NVARCHAR(128),
    Sync              TINYINT,
    Replaced          TINYINT,
    Hide              TINYINT,
    IsEvent           TINYINT,
    Duration          FLOAT,
    Color             NVARCHAR(10),
    Fill              TINYINT,
    Font              NVARCHAR(128),
    PopupText         TINYINT,
    UserID            NVARCHAR(64),
    UserAlias         NVARCHAR(128),
    UserGroup         NVARCHAR(128),
    DataContext       NVARCHAR(128),
    TagName           NVARCHAR(1024),
    Attribute         NVARCHAR(128),
    GroupName         NVARCHAR(128),
    Value             FLOAT,
    Title             NVARCHAR(128),
    ChartStart        DATETIME,
    ChartDuration     FLOAT,
    Contents          NVARCHAR(256)
);
-- Indexes: DateTimeUTCTicks (ASC), DateTime (ASC), Sync (ASC)
```

The chart's INSERT path (and the `TK.AddAnnotation` helper described on the *TK Annotation Helpers Reference* page) writes a **17-column subset**:

```
rowguid, DateTimeUTCTicks, DateTime, Sync, Replaced, Hide, IsEvent, Duration, Color, Fill, UserID, UserAlias,
UserGroup, TagName, Value, Title, Contents
```

The remaining 9 columns (`ID`, `SourceID`, `Font`, `PopupText`, `DataContext`, `Attribute`, `GroupName`, `ChartStart`, `ChartDuration`) take their SQL defaults on insert; the server reconciles the INSERT shape against the target schema by name and silently drops any column the target table does not expose. Customers managing their own annotation table need only the 17-column subset for ingest compatibility, but should provision the full 26-column shape if they want full feature coverage (event ranges, custom fonts, popup customization, chart-zoom-derived timestamps, etc.).

Customer-managed annotation tables

To use your own annotation table instead of the auto-created one:

1. Create a Dataset.Table object in your solution pointing to a SQL table with at least the 17-column ingest shape above.
2. Set `AnnotationsSource` on each chart that should use it to the Dataset.Table object name (e.g., `"MyAnnotations"`).
3. The chart will read from and write to the named table; the platform does NOT auto-create customer-named tables. Provision the table yourself.

Code example

The minimum wiring — on a fresh solution, with no scripts:

```
Designer side:
  1. On the TrendChart's properties pane, set:
       AnnotationsSource = "Annotations"
  2. (Optional, but recommended for kiosk / view-only deployments)
       Drive Client.TrendChart.AnnotationsEnabled = false at runtime to
       hide all four context-menu items wholesale.

Runtime:
  - Operator right-clicks the chart -> Add Tag Note.
  - Annotation dialog opens; operator types content; OK.
  - Row INSERTed into the Annotations table.
  - Pin renders on the next chart refresh.

No customer Script.Class required.
```

For headless server-side annotation creation (AI-generated, batch import, report-time generation), see the *TK Annotation Helpers Reference* page.

In this section...