TrendWorX64 Viewer – Performance Optimization

Application Note April 2018

Description: Guide to optimizing your TrendWorX64 Viewer.

Introduction

TrendWorX64 Viewer has the ability to replay large amounts of historical data in many pens. However, when replaying large periods from the history, the viewer could be overloaded by too many requested samples. In these cases the parameters of the TrendWorX Viewer must be correctly set to replay data smoothly and without any lag. We will focus namely on ICONICS Hyper Historian as the source of OPC historical data in this application note.

Graphics Optimization

There are a couple configuration tips you can use to optimize your TrendWorX Viewer on a GraphWorX display.

- **Viewer size** – You can improve the performance such as Viewer response period time by decreasing the viewer's size.
- **SampleDecluttering** – The Viewer will read all of the requested samples, however, it will only draw samples that can be visible depending on your screen resolution. It will maintain all peaks, density and quality of data. This feature is turned on for pens by default. In order to disable this optimization set SampleDecluttering to False in the Pen → Advanced menu, in the Data section.

Time Optimization

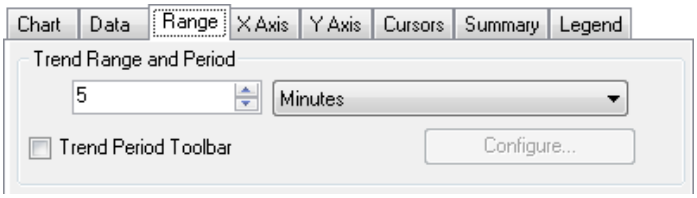

Figure 1 – Range Properties for Chart

- **Trend Period** – Shortening the Trend Period (the period of displayed history) will affect the overall performance of TrendWorX64 Viewer, including CPU usage, memory usage, chart data initialization time, and chart refresh time (under high CPU load), as well as chart manipulation response time.
- **Automatic Time Alignment –** If Automatic Time Alignment is checked, then the TrendWorX Viewer will automatically set tick marks based on the size of the control and the trend period.

Data Optimization

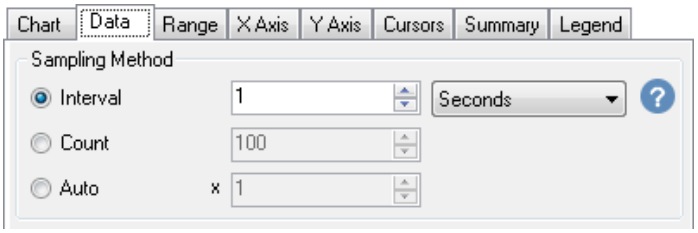

Figure 2 – Data Settings for Chart

- **Sampling Method** – The sampling method is only applicable for data sources that are both historical and aggregated. Real-time data sources use the sampling interval as the data collection rate. Raw historical data sources do not use sampling methods at all.
  - **Interval** – For historical pens with an aggregate such as Interpolative, the Interval setting determines how often a sample should be drawn. For example, with a five minute trend period and a sampling interval of 1 minute you will have five samples drawn. Slower rates make for faster updates between the server and client and fewer samples drawn, which leads to better performance. Choose Interval when the trend period is pre-configured and will not change in Runtime.
  - **Count** – This sampling method will draw a set number of samples, independent of the trend period. Choose Count if you expect the user to zoom or change the trend period to analyze analog data.
  - **Auto** – This sampling method automatically changes the interval based on the time axis settings. The number next to Auto refers to the sample count per major tick mark. Choose Auto for reliable performance.

Figure 3 – Realtime Mode Properties for Chart

- **Rates**
  - **Animation Speed** – Configuring this rate to be slower will affect the CPU usage when the Viewer is in Runtime and displaying data.
  - **History Read Rate** – Under some conditions this rate is modified automatically by the viewer to prevent overloading of the client-server communication. This

time defines how often the TrendWorX64 Viewer will retrieve the historical data.

- **Optimize History Read** – This setting automatically adjusts the frequency of the historical data read. It is enabled by default. If you need to turn this feature off you can find it on the Data tab of the Chart object as a checkbox, or in the Advanced tab of the Chart object as a True/False choice.

Samples and Buffer Optimization

- **Maximum buffer size** – This is the maximum sample count that an aggregate can use when drawing the particular pen after all values are calculated according to the currently applied aggregate type. Configuring this number will strongly affect the overall CPU usage when using aggregate pens.

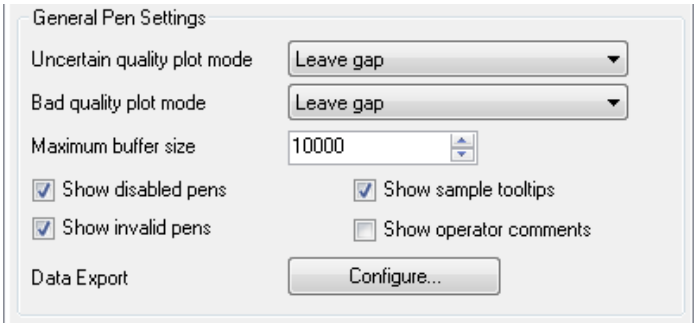

Figure 4 - Setting Buffer Sizes

Client Aggregates

There are many types of aggregation you can select and each of them has a different influence on data replaying performance. For example the most commonly used aggregates are: **Interpolative, Minimum, Maximum, Average, Start** and **End.**

Aggregates provide a way to perform calculations based on the logged data but they can also provide a way to optimize the network bandwidth because they are calculated server side. Optimization occurs only in cases where you have a large amount of data.

Client vs. Server Aggregates

It is important to understand the difference between client and server defined aggregates.

When aggregates are requested by client application on the fly, the historian server must process all samples in the interval and calculate the data explicitly for each request. Client aggregates are useful when searching for specific data or analyzing arbitrary intervals. They reduce the number of samples transferred over network and offloads the client application. On the other hand, it is the most expensive way to retrieve historical data from the server point of view.

Recommended solution is to define all common requests in historian configuration. Data will be aggregated continuously on the server, avoiding CPU and memory peaks, characteristic to client aggregates. In this case, the client can request "raw data" of the server aggregates, reducing both server load and amount of data to transfer.

This setup can be improved further by logging server aggregates to dedicated database.

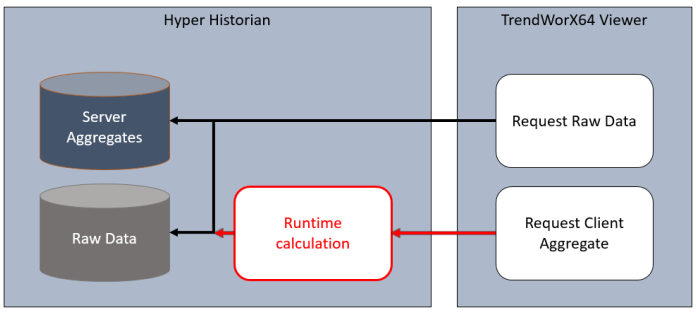

Figure 5 – Client and Server Aggregate