Appendix

Object Model and Namespaces

More advanced than most systems, where you must create Tags or Variables for all internal properties and custom logic for your projects, FactoryStudio allows your application(s) to directly access all the business objects that were created in your project. This means that user-created temporary tags are not required to manage the status of PLC network nodes, the total number of alarms in a group, or the number of rows in a dataset. You can now access runtime objects, business objects (representing a network node), an alarm group or dataset, and display required information or take action directly through their built-in properties.

FactoryStudio has an underlying .NET object model, 100% managed code, specifically targeting the development of Real-Time data management applications. The hierarchical object model includes the following top-level objects, which correspond to the main modules in FactoryStudio:

Tags	Dataset
Historian	Script
Security	Server
Alarm	Client
Device	Info

That top-level hierarchy is implemented as .NET Namespaces. Each Namespace has the .NET classes and objects created when building a project configuration. Besides having the configuration settings, those objects also have runtime properties, methods and status.

For instance that Tag namespace has all the tags in the application and each tag has built-in properties field properties such as Quality, TimeStamp, Min, Max, Units and many others.

Examples

Tag.tagname1.bit0, tag.tagname2.timestamp
The same concept of the tag fields applies to all namespaces, for instance:
Alarm.TotalCount:, Alarm.Group.Warning.Disable:

When building the project configuration, filling input fields or creating scripts, the system always has the Intellisense auto-completion, which guides you to the existing properties that are allowed to use according to what you are editing. This feature allows you to easily "drill down" to a specific property.

When accessing a project object in the .NET Script Editor, it is necessary to prefix the namespace with "@" symbol in order to avoid conflict with the .NET local variables names.

Examples

In Script-Tasks and CodeBehind, use:
@Tag.Analog1
@Device.Node.Node1.Status

The @ symbol is not necessary on Grids and Dialogs. Some input fields may require object of only one type, such as Tag or Display, the Intellisense will automatically guide you to the allowed objects.

For some users that don't have previous experience in .NET or similar object-oriented systems, those concepts are abstract at the beginning, but when learning the engineering configuration tools and the FactoryStudio modules, the power of those concepts will be clear. What is completely sure is that when getting used with object models and Intellisense, there is a huge productivity increment and you no longer accept working with systems lacking those features.

On this section we will explain more about the Namespaces available on Factory Studio.

On this page