Tags Assets and Templates

What are Tags and Templates?

Tags

In FactoryStudio and throughout this guide, the term "tag" refers to any real-time variable or its associated historical data. Tags usually map to devices, such as PLC registers or other physical equipment in the production process. A tag can also be connected to entries in SQL databases, external data sources, or an internally calculated value.

All tags have a specific type, such as integer, text, or date and time. FactoryStudio provides over a dozen predefined types, and allows users to define new types as well. In addition, some tags may be defined as arrays, and tags may have optional parameters.

Tags are the process variables for your application. Use tags and their properties to set up the data model for your process.

Templates

Although FactoryStudio provides many predefined types, sometimes you may need to define your own type. This is done using a template. Templates may be defined with properties similar to those of a predefined type. each template represents a user-defined type that you can use in the same manner as predefined types.

Using templates, you can extend the types of tags available and create new types to fit your application needs, such as machine data, equipment status, vessels, or the representation of any asset attributes in your plant.

On this page:

Built-in Tag Types

FactoryStudio provides 13 built-in tag types, most of which are based directly on .NET datatypes. These are summarized in the following table.

Tag Type	.Net Type	Value Range
Digital	System.Int32	0 through 1
Integer	System.Int32	-2,147,483,648 through 2,147,483,647
Long	System.Int64	-2,147,483,648 through 2,147,483,647
Double	System.Double	-1.79769313486231570E+308 through -4.94065645841246544E-324 for negative values; 4.94065645841246544E-324 through 1.79769313486231570E+308 for positive values
Decimal	System.Decimal	0 through +/-79,228,162,514,264,337,593,543,950,335 with no decimal point; 0 through +/-7.92281625142643375935439503 35 with 28 places to the right of the decimal; smallest nonzero number is +/-0.000
Text	System.String	0 to approximately 2 billion Unicode characters
Timer	System.Int32	Same range as Integer, but with built-in parameters to produce a repetitive wave pattern (see notes below)
DateTime	System.DateTimeOffset	from 12:00:00 midnight, January 1, 0001 to 11:59:59 P.M., December 31, 9999
TimeSpan	TimeSpan	Data Interval, in Days, Hours, Minutes, Seconds and Milliseconds, where each of those properties can hold a Double value
Guid	Guid	Standard Microsoft Globally Unique Identifier (GUID)
DataTable	System.Data.DataTable	Holds an in-memory DataTable
Image	System.Byte[]	Can hold an Image file or any binary content. Maximum size of the content is the long value.

Decimal

The Decimal type allows calculation with higher precision than Double. However, math operations using Decimal can be 40 times slower than using Double. So the Decimal type should only be used when Double precision is not enough.

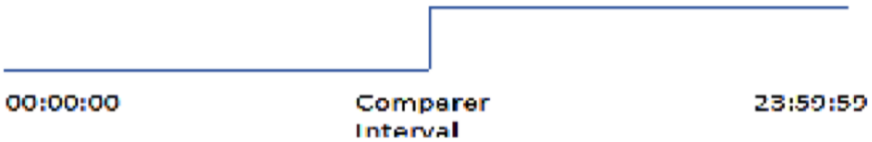
Timer

Timer is a built-in integer type that can be used to generate precise timing signals. Timers have the following varieties:

- SquareWave:  the value toggles between 0 and 1.
- Pulse: the the tag changes to 0, then changes immediately to 1.

- DelayOff: the tag behaves as a PLC Timer Off . During the runtime, if you set the tag with some value other than the StartValue, the tag will hold that value for the period specified in the Interval. The tag goes back to its StartValue after the period of time (Interval)

- Comparer: the tag is set to 1 after the specified comparer Interval, and then goes back to zero at midnight .



For the SquareWave, Pulse and Comparer, If you set the tag StartValue, the tag toggles between 0 and StartValue (instead of 0 and 1) .

Reference

Reference Tags allows dynamic addressing of the variables.

The concept is closer to the concept of References in .NET programming, more than the old C++ pointers. This is because Reference Tags, like .NET references, are **typed**; that means that when creating a Reference Tag, you need to define the type of the object the reference will be pointing to. Like .NET References and unlike C++ pointers, a Reference Tag cannot be defined to point to invalid memory areas that would cause errors in the application.

The target Type for Reference types is defined in the Parameters columns.

The use of typed references brings advantages both in Engineering and Runtime. In Engineering it allows Intellisense to directly browse the members of the template, when the reference is pointing to a template; In the runtime it allows string data validation.

All Reference Tags have an additional runtime attribute, the **link**, which specifies the tag that the reference will be linked to during execution.

Essentially the Link property is a string property that should receive the target tag name before using the reference tag. You can assign a string directly or by using a string expression, but the best way to set the Link property is to use the Method GetName(), which will create the string based on the current tag name. This way, you can rename the tag without having to search the strings. This also shows the tag names linked on the cross-reference utility.

An example project (ReferenceTags) ships with FactoryStudio.

Examples:
@Tag.Reference1.Link = @Tag.TagName.GetName() (VB)
@Tag.Reference1.Link = @Tag.TagName.GetName(); (C#)
@Tag.Reference1.Link = "Tag.TagName"; (C#)
@Tag.Reference1.Link = "Tag.TagNa" + "me"; (C#)

The reason for using the GetName() method instead of using strings directly, is that with GetName() you retain the benefits of Cross-Reference and Refactoring.

Creating and Editing Tags

To create and edit tags:

- Go to **Edit > Tags > Objects**.
- To create a new tag, enter its name and other applicable properties in the blank top row (Insertion Row). To edit an existing row, select any item in the row and make the desired changes. Here is a complete table of available tag properties. Note that many of these properties are optional and only apply to specific kinds of tags.

Column	Description
Name	Enter a name for the tag. The system lets you know if the name is not valid. • If you edit the name of an existing tag, the system automatically updates the name throughout the project.
Type	Select the tag type, which may be a built-in type or a user-defined template.
Parameters	Configure the parameters, if any. The parameters vary based on the tag type. Once the type is entered, you can double click on the parameters field to see a dropdown menu with the applicable parameters.
Array	When this field is blank, the Tag is not an Array. When the field contains an integer value N, an Array is created from position 0 to N. For example, if the field contains the value 5, the Array is created from Tag[0] to Tag[5]. This means that 6 elements are created. Two programming styles are accommodated by this method; one that counts elements from 0 to less than five, and one that counts from 1 to 5.

The columns above are visible by default. To add or remove a column, right-click the column heading area and check or uncheck the columns that will be visible.

Column	Description
Units	Enter the engineering units of measure that you want to use as a label for this tag.
StartValue	Enter a starting value for this tag. This is the value the tag will be initialized with when FactoryStudio starts.
Format	Enter a default format for displaying the data. For example: • N0—Number with no decimal places. • N3—Number with three decimal places. • X—Hexadecimal (supported only for integral types). • C—Currency. • When configuring output on the Displays, it is possible to define the format for each output field individually, but in most scenarios it is easier to attach the default formatting to the tag itself.
Retentive	Select an option to save the value of the tag and its internal properties to the database every time the value changes. This retains the value when the application shuts down and makes the value available when the application next starts. • None—Does not retain the value or properties. • ValueOnly—Retains only the value. • Properties—Retains all properties, including the value. • PropertiesOnly—Retains all properties, except the value.
Min	Enter the minimum value that is valid for the object.
Max	Enter the maximum value that is valid for the object.
Visibility	Select the value visibility on the OPC server for remote projects: • Private—Tag is visible only to the local project and redundant pair. • Protected—Read-only tag that is visible on the OPC server to remote projects and OPC clients. • Public—Tag is visible on the OPC server to remote projects and OPC clients.

Domain	Tag value for the entire project or value specific to each client display. • Server—Tag value is consistent across the entire project and all remote client displays. • Most tags in a project should be Server tags. • Client—Tag value is local to each remote computer running a client display (web or visualizer displays). • Use Local tags to denote temporary data specific to individual client computers. The most common use of Local tags is when temporary data is needed to manage the user interface on the displays. • Local tags allow different values on each client computer.
Comment	Enter any comments about this tag.
ReadSecurity	Select which groups have the right to read the Tag. Tag Security protection can be configured in Display > Client Settings
ScaleMin	Enter the scale min value for communication.
ScaleMax	Enter the scale max value for communication.
DevicePoint	Read-only. Show which communication point is related to the tag (if related)
WriteSecurity	Select which groups have the right to write in the Tag. Tag Security protection can be configured in Display > Client Settings
[Other columns]	For definitions of other columns that are available in this table, see "Common Column Definitions".

Continue adding as many tags as you need.

Notes

You can also create a tag from anywhere in FactoryStudio by clicking **New Tag** in the toolbar.

Like any other configuration table, you can also import CSV files or copy/paste contents directly from an Excel spreadsheet or other applications.

With FactoryStudio you can replace names at any time, so an easy way to create a tag is to click with the mouse on the Name column of the insertion row, then press space and enter. Each time, the system will create a Tag of the same type as the last one created. You can use the mouse on the insertion row to select the Type, then click with the mouse in the header or any other part of that grid. That will also create a tag with a default name. Also, you can configure more than one row at the same time, just select the rows with Shift button then right-click and select "Edit Combined Rows". A new popup will open with the rows' information. All settings changed in this window will change all rows selected. If a column has blocked URL, it shows that this column has more than one configuration.

Tag Formats

The format property defines the display format for tag values. These formats follow the specifications provided in Microsoft .NET. For valid numeric formats, refer to Standard Numeric Format Strings (http://msdn.microsoft.com/en-us/library/dwhawy9k%28v=VS.90%29.aspx). For example: N1 (number with 1 decimal place).

For valid date and time formats, refer to Standard Date and Time Format Strings (http://msdn.microsoft.com/en-us/library/az4se3k1%28v=VS.90%29.aspx). For example: d (short date).

For a more in-depth discussion of format strings, refer to Formatting Types (http://msdn.microsoft.com/en-us/library/fbxft59x%28v=VS.90%29.aspx).

Numeric format examples	
Specifier	**Description**
N0	Number with no decimal places
N3	Number with 3 decimal places
X	Hexadecimal (supported only for integral types)
C	Currency

Date/time format examples	
Specifier	**Description**
T (only)	Long time pattern (equivalent to HH:mm:ss).
d (only)	Short date pattern (equivalent to M/d/yyyy (month/day/year) for en-us).
dd	Show the day of the month as a number from 01 through 31.
ddd	Show the abbreviated name of the day of the week.

dddd	Show the full name of the day of the week.
MM	Show the month as a number from 01 through 12.
MMM	Show the abbreviated name of the month.
yy	Show the year as a two-digit number.
yyyy	Show the year as a four-digit number.
hh	Show the hour as a number from 01 through 12.
HH	Show the hour as a number from 00 through 23.
mm	Show the minute as a number from 00 through 59.
ss	Show the seconds as a number from 00 through 59.
fff	Show the milliseconds as a number from 000 through 999.
tt	Show the A.M./P.M. designator.

Creating Templates

Templates let you create new tag types based on existing built-in types.

To create a tag template:

- Go to **Edit > Tags > Templates**.
- Click **New**.
- The Create New Tag Template dialog shows.
- In the New Type Name field, enter a name for the tag type. In the Description, enter a description of the tag. Click **OK**. The Templates tab displays with the name of the new template at the top of the tab.
- Click the insertion row to create a new attribute for this tag template.
- Enter or select information, as needed, all the properties are the same as in Tags (see Creating and Editing Tags).
- To delete a template, select it from the User Custom Type drop-down list, then click **Del**.
- On the **Objects** tab, for new tags or existing ones, you can now use this new template in the Type column.

Assets and Categories

Assets

If you have the Enterprise version of FactoryStudio, assets let you configure additional metadata for your project. For example, you can organize objects in your project, such as tags, devices, and alarms, into a hierarchy. This lets you group tags that are related to each other. The hierarchy may reflect such things as areas of your manufacturing floor or the location of your machinery.

To create assets:

- Go to **Edit > Tags > Assets**.
- Right-click "Elements" and select **New Level**.
- Enter a name for the level.
- Right-click the new level and select **Insert Asset**.
- The Select Object window displays, with all objects, by type, on the left side.
- Select the object type from the left side and the object you want from the right side.
- Click **OK**.
- The object becomes a child of the selected level.
- Continue adding child or sibling levels and inserting assets, as needed.
- If needed, right-click a level to rename or delete, or right-click an asset to delete it.
- On the **Objects** tab, for new or existing tags, select the new level in the Level column.

Categories

If you have the Enterprise version of FactoryStudio, you can create user-defined categories of data that you can use as metadata for tags. Categories are useful for filtering, both when creating the project and during runtime.

To create categories:

- Go to **Run > Dictionaries > Categories**.
- Enter or edit the name and description for the category.
- Continue adding as many categories as you need.

- On the **Tag -> Objects** tab, for new or existing tags, select the new category in the Category column. Other Project elements can also use categories for project organization.

The Tag Namespace

All project tags are available to the runtime modules as .NET objects at the Tag Namespace.

All Built-in Tag Types share a common set of properties and methods defined in the base class **TagObj**. The Tags created from user defined templates are implemented by the base class **UserType**.

Class Type	Description.
TagObj	Base classes to all Tag objects.
Digital	Runtime properties for tags of Type Digital.
Analog	Runtime properties for all Analog Tag Types.
AnalogInt	Runtime properties for tags of Type Integer.
AnalogLong	Runtime properties for tags of Type Long.
AnalogDecimal	Runtime properties for tags of Type Decimal.
AnalogDouble	Runtime properties for tags of Type Double.
Text	Runtime properties for tags of Type Text.
TDateTime	Runtime properties for tags of Type DateTime.
Timer	Runtime properties for tags of Type Timer.
TTimeSpan	Runtime properties for tags of Type TimeSpan.
Reference	Runtime properties for tags of Type Reference.
TDataTable	Runtime properties for tags of Type DataTable.
UserType	Runtime Properties for tags from Templates.

See "Namespaces" for the complete programming reference on runtime objects.