Keeping Tags Organized

It is important to give Tags a meaningful structure and arrange them in hierarchical Tag folders so that they are easy to understand, identify, and locate for all developers. You can also rename Tags independent of their Item Path.

Another important concept to consider when naming and organizing your Tags, is to do this early in your project. If you rename or move any of your Tags to another folder, and your Tag is being used in other places, chances are you are going to break the reference to the Tag on your screen. So keeping your Tags organized and defining your Tag structure early on in your project is critical.

On this page ...

INDUCTIVE UNIVERSITY

Keeping Tags Organized

Watch the Video

Tag Name Rules

The best way to keep your Tags organized is to rename them with something meaningful. By default, Ignition Tags are named after their OPC Server address when a Tag is dragged into the Tag Browser. You can change this name to just about anything that you want. We recommend using names that mean something to your process like 'Motor 3 Amps,' or creating folders in your Tag Browser like 'Motor 3/Amps'. When renaming Tags and folders, there is really only one question to ask: "does this structure make sense?"

When you choose a new name for your Tags and Folders, there are some rules that must be followed. The first character of the Tag name must be one of the following:

- any alphanumeric
- any valid unicode letter
- an underscore

The second character, and every character after that can then be one of the following:

- any alphanumeric
- any valid unicode letter
- an underscore
- a space
- and any of the following special characters:

'	-	:	(	)

All other special characters can not be used in a Tag name.

Tag Paths

Tags and their properties can be referenced by a string-based path. Each Tag has a unique absolute path and often has many equivalent relative paths when referenced from other Tags. You most often generate these paths by browsing or through dragging. However, it's a good idea to understand how Tag paths work, particularly if you get into indirect Tag binding or scripting.

A Tag path looks something like this: `[Tag Provider]folder/path/tag.property`

The `folder/path/tag.property` portion of the path may contain the following:

- A Tag
- Any number of nested folders followed by a Tag, separated by forward slashes (/).
- A period (.) followed by a property name after the Tag. Omitting this is equivalent to using the .Value property.

The `[Source]` portion surrounded by square braces can have the following options:

Source Option	Meaning	Applicability
[Tag Provider Name]	The name of the Tag provider that hosts the Tag.	OPC, Expression Tags
[] or not specified	The default Tag provider for the current project.	OPC, Expression Tags

[.]	Relative to the folder of the Tag that is being bound.	Expression, Client Tags
[~]	Relative to the Tag provider of the Tag that is being bound (root node).	Expression, Client Tags
[Client]	Refers to a Client Tag.	Client
[System]	Refers to a System Tag.	System

Using Relative Paths

Paths that begin with [.] or [~] are known as *relative paths*. They are used inside Tags that bind to other Tags, and are relative to the host Tag's path. Using the relative path syntax helps to avoid problems caused by moving Tags and renaming providers.

[.] refers to the Tag's current folder. By using [.], Tags can be moved from folder to folder without problem (provided that all of the applicable Tags are moved together). Additionally, you can use "**..**" (two periods) to go back one folder from the current relative position, for example [.]../../tag allows you to reference a Tag two folders up.

[~] refers to the Tag's provider root. It can replace the explicit provider name, and thus protect against provider renames and importing/exporting/moving Tags between different providers.

Tag Path Manipulation

Ignition provides a great deal of flexibility for Tag addressing since Tag paths and Tag properties are string-based. The underlying strings that compose a valid Tag path can be assembled from many different parts in with the eventual construction resulting in a valid Tag path.

The following scripting demonstrates this concept. Suppose there was a Tag path to a level indicator in a tank. In this case it is the default Tag provider, Tanks folder, Tank 1 Folder, and the Level Tag.

```
tagPath = "[default]Tanks/Tank 1/Level"
```

But suppose that there was more than just Tank 1 and instead there was Tank 2, Tank 3, Tank 4, etc,. Dynamically changing the Tag paths is simple with Ignition's Tag paths being nothing more than string representations. The following takes the tank number and inserts it into a new Tag path. The tankNumber variable changes the eventual creation of the tagPath.

```
tankNumber = 2
tagPath = "[default]Tanks/Tank %i/Level" % tankNumber
```

The result of the tagPath variable will be [default]Tanks/Tank 2/Level which is a valid Tag path to the the level sensor for Tank 2.

Related Topics ...

- Tag Quality and Overlays