

Ignition Tag Importer

Explanation on how to import tags, alarms and history settings from an Ignition tag export.

[Reference](#) [Solution](#) [Designer](#) [Settings and Tools](#) [Import Tags](#) Ignition

Available from FrameworkX 10.1.6.

The Ignition importer brings the tags of an Ignition tag export into a FrameworkX solution, with their data types and descriptions, and creates alarm items and historian tags from the alarm and history settings each tag carries. Use it to start a migrated project from the existing tag list instead of an empty one. The importer ships with FrameworkX and needs no extra installation.

- Instructions
 - [Wizard Startup](#)
 - [Select the Export File](#)
 - [Import Log](#)
- What Is Imported
 - [Tags](#)
 - [Alarms](#)
 - [History](#)
- [What Is Not Imported](#)

Instructions

Wizard Startup

Go to **Solution Import Tags** on the Designer workspace and select **Ignition Project (Tags)**. Double-click the wizard or press the **Execute** button to open the configuration dialog.

Select the Export File

In **Ignition Tag Export (JSON) File**, click the ... button and select the JSON file exported from the Ignition tag browser. Press **OK** to start the import.

Import Log

The importer writes a log of each run to the product's TraceLogs folder, in the file <SolutionName>LastImportIgnition.log. The log lists every tag imported and every item left out, with the reason, so nothing is dropped silently.

What Is Imported

Tags

Each Ignition tag becomes a FrameworkX tag with its data type and description. The importer walks folders and UDT instances and imports the tags inside them. A tag inside a folder gets a name built from the folder path, with the segments joined by an underscore. For example, the tag Level in folder Line1/Tank becomes Line1_Tank_Level. A tag of a UDT instance becomes an individual member tag in the same way.

Ignition data type	FrameworkX tag type
Boolean	Digital
Int1, Int2, Int4, Byte, Short, Integer	Integer
Int8, Long	Long
Float4, Float8, Float, Double	Double
String	Text
DateTime	DateTime
Any other type	Text, with a line in the import log

Running the import again on the same solution updates the existing tags in place and creates no duplicates.

Alarms

Alarm conditions, limits and priorities become alarm items. Ignition alarm modes map to FrameworkX conditions as follows:

Ignition alarm mode	FrameworkX condition
AboveValue	Hi

BelowValue	Lo
Equal	Equal
Inequality	NotEqual
AnyChange	Changed
RateOfChange	RateOfChange

The alarm setpoint becomes the alarm item limit, with its decimals. Ignition priorities map to FrameworkX priorities as follows: Diagnostic 1, Low 3, Medium 5, High 8, Critical 10. A numeric priority is kept and limited to the range 1 to 10.

A FrameworkX alarm item is identified by tag plus condition. When an Ignition tag carries two alarms of the same mode, only the first becomes an alarm item, and the import log lists the second one.

History

A tag with history enabled in Ignition becomes a historian tag. The historian table takes the name of the Ignition history provider and is created when it does not exist. When history is enabled but no history provider is named, the tag goes to a default historian table named `IgnitionHistory`. The history deadband value is carried to the historian tag.

A tag with no alarms and no history is imported as a tag only, with nothing extra added.

What Is Not Imported

The following items are not created in FrameworkX 10.1.6. The import log lists each tag-level item left out.

- Ignition UDT type definitions. They are not recreated as FrameworkX user types. The tags of UDT instances are imported as individual member tags.
- Display pages.
- Alarm modes with no FrameworkX condition: `BetweenValues`, `OutsideValues`, `BadQuality` and `OnCondition`.
- A second alarm of the same mode on the same tag.
- History sampling modes and deadband modes. The deadband value is imported.
- Device connections from the Ignition OPC item path. No channel, node or point is created.
- Placement of the tags in the asset tree.

In this section...
