

system.tag.editTag

This function is used in **Python Scripting**.

Description

Edits an existing Tag in Ignition. This will not work on Client Tags, because there is a Client Provider for each project.

This can now be used to edit members of a nested UDT instance.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when ran from the Gateway scope.

Syntax

system.tag.editTag(tagPath, attributes, parameters, accessRights, overrides, alarmList, alarmConfig)

- Parameters

String tagPath - The full path to the Tag you want to edit. For members of UDT instances, the tagPath will be the path to the UDT instance, with the overrides parameter listing out the member Tags to edit. Note: you can specify the Tag provider name in square brackets at the beginning of the parentPath string. Example: "[myTagProvider]MyTagsFolder". If the Tag provider name is left off then the project default provider will be used.

PyDictionary attributes - The Tag's configuration attributes.

PyDictionary parameters - The parameters for a UDT instance Tag.

String accessRights - The access rights for the Tags. Possible values are Read_Only, Read_Write, and Custom.

PyDictionary overrides - All of the overrides for a UDT instance Tag. The dictionary should be in the form of the names of member Tags as keys, with the values being a dictionary of properties/overrides ie. {'memberTagName':{dictionary of overrides}}.

String alarmList - List of legacy alarms for the Tag. The legacy alarm system was retired in 7.6.0. Newer systems should utilize the `system.tag.editAlarmConfig` function instead.

PyDictionary alarmConfig - The alarm configuration for the Tag. Note that this parameter cannot edit alarms on UDTs. Instead, the `system.tag.editAlarmConfig` function (which can also edit alarms on non-UDT Tags) should be used instead. See `editAlarmConfig` for details on how to use this parameter.

- Returns

Nothing

- Scope

All

If called in the Gateway scope, a Tag provider must be specified.

Associated attributes:

- Complete list of the acceptable `Tag Properties` and `alarmConfig`.

Code Examples

Code Snippet

Example 1: Edit OPC Tag.

```
system.tag.editTag(tagPath="Tag1",
attributes={"OPCServer":"Ignition OPC-UA Server",
"OPCItemPath":"[MLX]N7:2"})
```

Code Snippet

Example 2: Edit UDT instance parameters.

```
system.tag.editTag(tagPath="Tag5",
parameters={"DeviceName":"CLX", "MotorNumber":2})
```

Code Snippet

Example 3: Edit UDT instance called Tag8 and override certain properties of a member Tag called STATUS.

```
system.tag.editTag(tagPath="Tag8",
overrides={"STATUS":{"ScanClass":"Default"}})
```

Code Snippet

Example 4: Edit a nested UDT instance. Assuming the following:
- A UDT instanced named 'ParentInstance' contains a nested UDT named 'ChildInstance'
- ChildInstance contains a Tag (member) named 'ChildMember'

The following script would override multiple properties on the ChildMember tag.

```
path = "ParentInstance/ChildInstance"
myOverrides = {"ChildMember":{"ScanClass":"Default",
"Enabled":"false"}})
```

```
system.tag.editTag(tagPath=path, overrides=myOverride)
```

Code Snippet

Example 5: Edit UDT instance called Tag8 and remove certain overrides from a member Tag called STATUS.

```
system.tag.editTag(tagPath="Tag8",
parameters={"Param":"Something"},
overrides={"STATUS":{"ScanClass":None}})
```

Code Snippet

Example 6: Enable history on a Tag, set the historical scanclass to "Default Historical", and set the History Provider to the "Data" provider.

```
system.tag.editTag(tagPath="Folder/Tag",attributes={"HistoryEnabled":True, "HistoricalScanclass":"Default Historical",
"PrimaryHistoryProvider":"Data"})
```

Example 7: Edit a Tag with an Alarm using the Legacy alarmList parameter.

```
alert = "myAlert;Medium_High;0.0;50.0;0;;2.0;SEC$"
```

```
system.tag.editTag(tagPath = "Folder/Tag", alarmList = alert)
```