



Ignition!
Exchange

Best Practices and Style Guide

Quick Reference

		Perspective	Vision
Visualization	View / Window Names*	View Name	Window Name
	Template Names*		Template Name
	Properties	propertyName	propertyName
	Component Names	ComponentName	ComponentName
	Page Names	page-name	
	Fonts	Use font defaults	Use font defaults
	Theming	Built-In Theme Colors	
	Style Class	style-class	

Resource Structure		exchange/<resource>/<item> - resources that start lowercase Exchange/<resource>/<item> - resources that start uppercase
Project	Project Names	project-name
	Project Titles	Project Title
Tags / UDTs	Folders and Tag Names	Exchange/ResourceName/TagFolder/TagName or Exchange/Resource Name/Tag Folder/Tag Name
Named Queries	Query Names*	Query Name
	Parameter Names	parameterName
Python	Tabs v.s. Spaces	Tabs
	Project Library	exchange.folder.script exchange.folderName.scriptName
	Variable Names	variableName
	Function Names	functionName
	Class Names	ClassName
	Single-Line Comments	# Single-Line Comment
	Multi-Line Comments	# Multi-Line # Comments
	Single Line Doc Strings	"""Single-Line Documentation String."""
	Multi-Line Doc Strings	Google's Python Style Guide
	Other code style choices	PEP-8
Database	Table Names	ex_<project name abbreviation>_table_name
	ID Column	id

	Foreign Key Column Name	foreign_table_name_id
	Column Names	column_name
Alarm Notification Pipelines	Names and Folders*	Exchange/Resource Name/Alarm Pipeline
	Custom Properties	propertyName
SFCs	Names and Folders*	Exchange/Resource Name/SFC Name
	Chart Parameters	parameterName
	Block Names	BlockName
SQL Bridge	Names and Folders*	Exchange/Resource Name/New Group
	Item Names*	Item Name
Reports	Names and Folders*	Exchange/Resource Name/New Report
	Parameter Names	ItemName
	Component Names	ComponentName
WebDev	Names and Folders	exchange/resource-name/example-resource
Image Management		exchange/resource-name/image-name.jpg
Translations		word (single-word translations)
		_translation_phrase (multi-word translations)

*Top level resources may use Title Case or PascalCase

Table of Contents

Table of Contents	4
Ignition Exchange Best Practices	5
General Guidance	5
Project Browser Resource Structure	5
Variable / Data Storage	6
Naming Conventions	7
Projects	7
Perspective	7
Tags	9
Vision	10
Code	11
Database	14
Named Queries	15
Alarm Notification Pipelines	16
Sequential Function Charts (SFCs)	16
SQL Bridge (Transaction Groups)	17
Reports	18
Web Development Module (WebDev)	19
Miscellaneous Designer Tools	20
Before You Upload to the Exchange	22
Things to check for:	22
Check the project export	22
Include the tags	22
Don't forget database backups	22
Take notes of the steps taken when checking your project export	22
Uploading to the Exchange	23
Overview Section	23
Package Section	26
Helpful Notes	29

Ignition Exchange Best Practices

When developing Exchange Resources, following the styles outlined below will help with consistency of naming conventions.

Having resources conform to these standards helps make resources easier to understand, explore, implement, and upgrade. These best practices should be adhered to unless it is not possible for the resource in question. We encourage you to follow the naming conventions outlined here, however these conventions are not required. **Consistent naming conventions are critical, regardless of what naming conventions you ultimately use.**

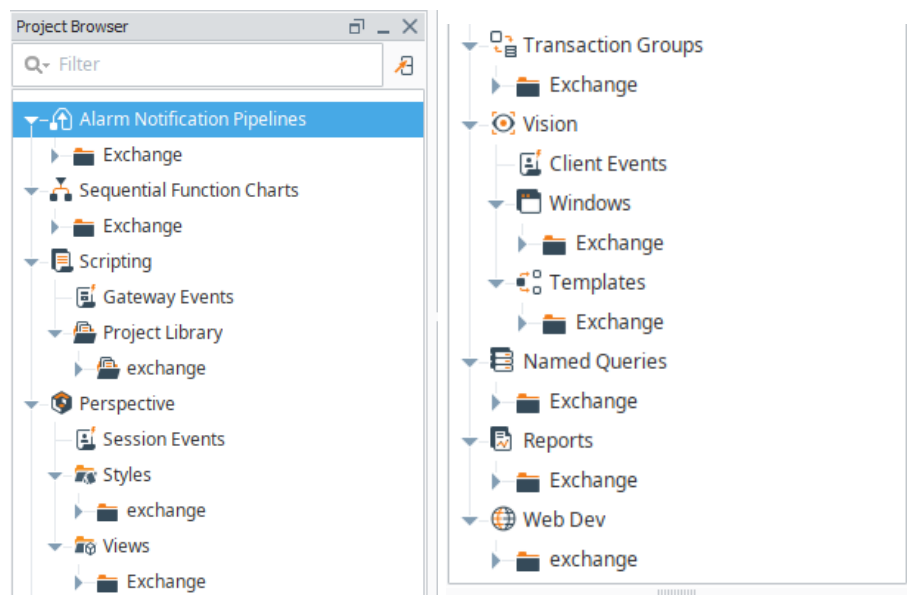
General Guidance

Project Browser Resource Structure

`exchange/<resource>/<item>` - resources that start lowercase

`Exchange/<resource>/<item>` - resources that start uppercase

Views, Styles, Scripts, etc. **must** all be contained within an appropriately named folder. This allows easy imports into existing projects, where those project resources won't overlap, interfere, or overwrite existing parts of a user's project. It also allows for easy updates to Exchange resources for users when a new version is released. Look below to see which resource categories should start lowercase and which should start uppercase.



Variable / Data Storage

Temporary Client / Session Variables

[View / Window Properties](#)

```
system.util.getGlobals()['exchange']['resourceName']['clientId']
```

Temporary Gateway Variables (Persists Until Gateway Restart)

```
system.util.getGlobals()['exchange']['resourceName']
```

Long Term Storage

[Database Tables](#)

Avoid

[Perspective Session Properties](#)

[Vision Client Tags](#)

Variables that are only expected to be used by a single client / session can often be stored in properties on views or windows. For items that need to be stored even when the view or window isn't shown anymore, the globals dictionary at `system.util.getGlobals()` can be a great place to store these things. Data, objects, or variables intended to be used over multiple sessions can be stored in globals as well.

Although projects will often use Perspective session properties or Vision client tags for this type of storage, doing this for an Exchange resource limits the ability to import your resource into another project. Using just the variables suggested here will help ensure that importing your resource will be smooth when someone wants to use it in a separate project.

A scoping note on `system.util.getGlobals()`:

Vision: In a Vision client, this is a dictionary inside the client, and there's a separate dictionary for each client. For scripts running in the client scope, they'll access the client globals dictionary. For scripts running in the Gateway scope, it'll access the shared Gateway globals dictionary.

Perspective: the globals dictionary is always the shared Gateway globals dictionary. The same data is available from the Gateway scope and from sessions.

Naming Conventions

Consistent naming helps with projects feeling cohesive, and can also help to easily identify what type of object is being used just by looking at its name for certain resources.

Projects

Project Names

`project-name`

Starting in Ignition 8.1.11, these should be lowercase and use dashes for multi-word names to be URL-friendly. Project names should only include letters, numbers, hyphens, and underscores. Please do not append the version number to the end of the project name, the version number can be managed through the Exchange during the [upload process](#).

Project Titles

`Project Title`

These should be Human-friendly, with capitals and spaces as appropriate.

Perspective

View Names

`View Name` or `ViewName`

Views can either be Title Case or PascalCase.. This gives view names an easy to read format.

Properties

`propertyName`

These should be camelCase, starting with a lowercase letter. This matches the style of the built-in properties on Perspective components.

View Properties

`params.publicProperty`

`custom.privateProperty`

For embedded views, put any property that's intended to be used to configure the view as a property in the param category. Put properties that you create for internal use inside the view itself as a property in the custom category. This separation helps make it clearer

which properties will just be used internally in the view and which are intended to be settings for the view.

Component Names

`ComponentName`

These should be PascalCase starting with a capital letter and capitalizing the first letter of each word. This provides consistency with the default component names.

Page Names

`page-name`

These should be lowercase and use dashes for multi-word names. This puts them in a format that's URL-friendly.

Message Handlers

`exchange.resourceName.handlerName` or `exchange.resourceName.viewName.handlerName`

Unique message handler names prevent collision with existing handlers. Using `exchange.resourceName` as a naming convention will identify the resource in console logs as well as facilitate unique naming. This is a recommendation and not a requirement for publishing a resource on the Exchange.

Fonts

Font sizes should match the default font sizes of most components, or take advantage of styling as appropriate. Matching defaults makes the resource fit stylistically inside projects that are using Ignition's defaults.

Theming

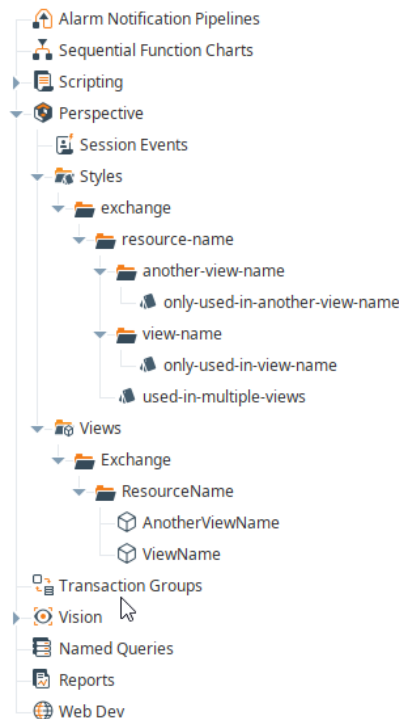
Supporting themes is optional, but is a nice feature for users of your resource. When supporting theming, set colors to appropriate theme colors so Ignition's theming can change the colors displayed as different themes are selected. Refer to the Ignition documentation for [Built-In Theme Colors](#).

Style Classes

`style-class`

This should be lowercase with dashes for multi-word names. Style class naming and hierarchy should match the hierarchy of the Perspective views they belong to. If style class is used across multiple views, place the style class at the topmost folder that

encompasses all views using it. If the style class is exclusively used in one view, create a folder named after the view name and place the style class within it. (See picture below for an example.)



Tags

For tag names and folders, you can use either PascalCase or Title Case. It's nice that Ignition supports spaces in tag names and folders, and you may use this to your advantage to make your tags and tag folders more readable. However, you may find PascalCase to be a neater option for tag names. This structure also applies to UDT definitions.

Note that spaces in tag names may lead to property names with spaces specifically when using UDT bindings in Perspective. The designer will notify of this as non-standard, but doesn't degrade performance or affect clients in any way.

Tag Folders and Names

`Exchange/ResourceName/TagFolder/TagName`

or

`Exchange/ResourceName/Tag Folder/Tag Name`

Vision

Window Names

[Window Name](#) or [WindowName](#)

Window names can be either Title Case or PascalCase.

Template Names

[Template Name](#) or [TemplateName](#)

As with window names, template names can be either Title Case or PascalCase.

Properties

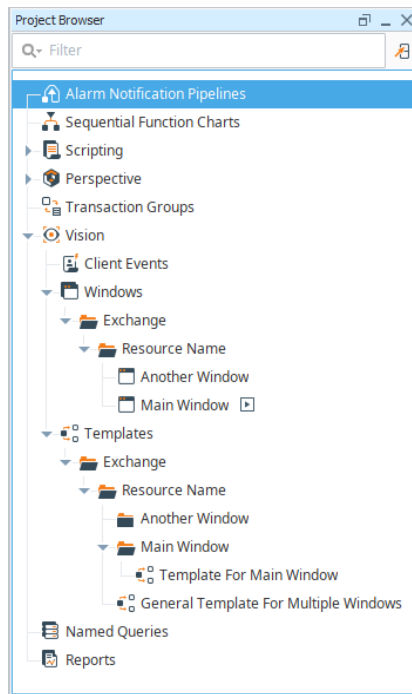
[propertyName](#)

These should be camelCase, starting with a lowercase letter. This matches the style of the built-in parameters on Perspective components.

Component Names

[ComponentName](#)

These should be PascalCase, with a capital letter and capitalizing the first letter of each word. This provides consistency with View Names and with Vision Component Names.



Code

Having good names for variables, functions, and consistency with white space is important for readability and maintainability of code.

Tabs v.s. Spaces

Ignition's editor uses tabs when the tab key is pressed. Therefore, tabs should be used for consistency, compatibility with code from other Ignition users, and maintainability.

Project Library

`exchange.folder.script`

`exchange.folderName.scriptName`

Scripts and folders in the project library should be all lowercase, and should be single words where possible.

Variable Names

`variableName`

These should be camelCase, starting with a lowercase letter. This is the style used in the Ignition user manual, to have consistency with property names. Note that this differs from PEP-8. Variable names that are being used to pass data to Ignition's API functions should match the names of the arguments in functions where it makes sense. For example, if you're defining a variable to pass to startDate in a function, call your variable startDate. Prefixing with another name may also make sense depending on your application, like processStartDate.

Function Names

`functionName`

These should be camelCase, starting with a lowercase letter. This is the style used in the Ignition user manual, to have consistency with Java method name conventions. Note that this differs from PEP-8.

Class Names

`ClassName`

CamelCase starting with an uppercase letter.

Single-Line Comments

```
# Single-Line Comment
```

Comments should start with a `#` followed by a space before adding the comment. Single-line comments should be placed one line above the code being explained.

Multi-Line Comments

```
# Multi-Line  
# Comments
```

Each line should start with a `#` followed by a space to line up the starting word of each line.

Single-Line Documentation Strings

```
"""Single-Line Documentation String."""
```

Documentation strings are used for the first statement in functions, classes, and methods. These should appear after the `def` line, have one additional indent from the `def` line, and be wrapped in triple double-quotes on the same line as the comment itself. Each sentence should be terminated by either a period, question mark, or exclamation point.

Multi-Line Documentation Strings

```
"""Multi-Line Documentation String.
```

```
    Some Description.
```

```
    Args, Returns, or Raises:
```

```
        A description of what is returned.
```

```
        Example: {foo: bar}
```

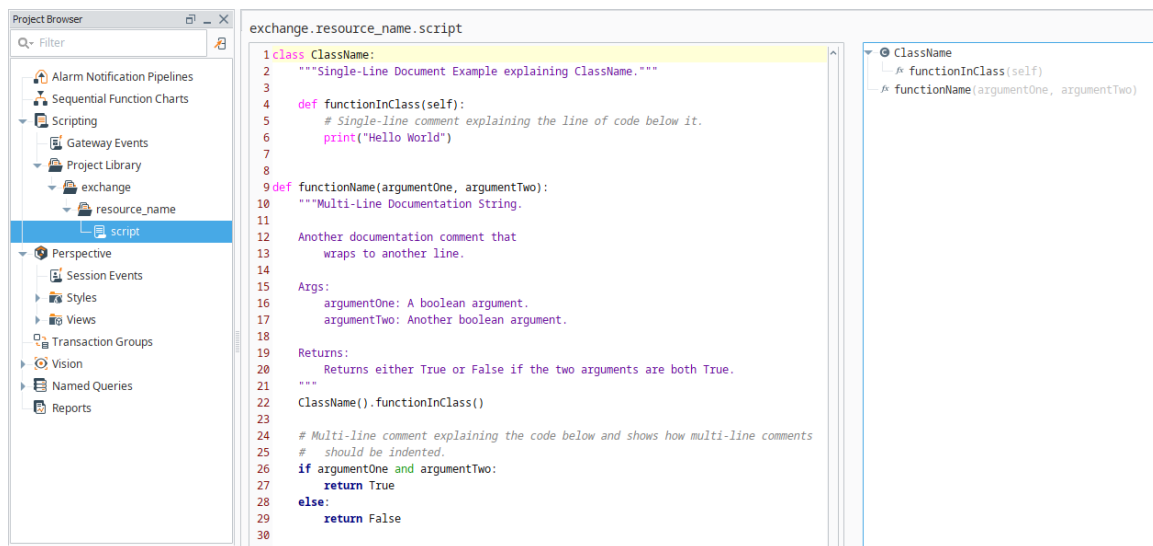
```
    """
```

Following the same guidelines as single line documentation strings, the main difference is the format (i.e., listing arguments, return format). Each description or summary line should be terminated by a period, question mark, or exclamation point. When writing more in the same comment block, there should be a blank line between the two descriptions. Each following line after the first, should start at the same indent spacing as the first quote in the first line. Each args, returns, or raises section title should end with a colon and the following description should be indented with tabs. Refer to [Google's Python Style Guide](#) for more examples.

Beginning with Ignition 8.1.32, custom project script functions that have docstrings defined in this manner are displayed via Ignition's autocomplete.

Line breaks and other code style choices

For everything other than the items listed above, refer to [PEP-8](#). A number of naming conventions in Ignition match PEP-8, and a number are different (like Variable Names and Function Names). When a choice isn't covered in this style guide, use PEP-8 as a fallback reference.



Database

Some database systems are case sensitive and some do not allow upper case in table or column names by default. The following standards will ensure consistent naming regardless of database platform. In general, use snake_case for naming all database objects. This means all lowercase with underscores between words for human readability. Hyphens or dashes are not accepted as names by all database platforms. Avoid numbers, sql keywords and special characters in names.

Schema Names

ex_resource_name

Schema names should start with ex followed by the Exchange Resource name to avoid sql import/create conflicts in pre-existing schemas.

Table Names

`ex_<project name abbreviation>_table_name` (ex_cm_contacts for example)

Table names should start with "ex_" followed by the Exchange Resource's abbreviated project name to avoid sql import/create conflicts in pre-existing schemas.

ID Column

`id`

First auto-increment primary key column in all tables should be "id". Being a primary key also allows Ignition's database query browser to support the table fully, allowing users to edit fields with Ignition's built-in tools. A consistent name for the primary key in all tables will aid in creating reusable components.

Foreign Key Column

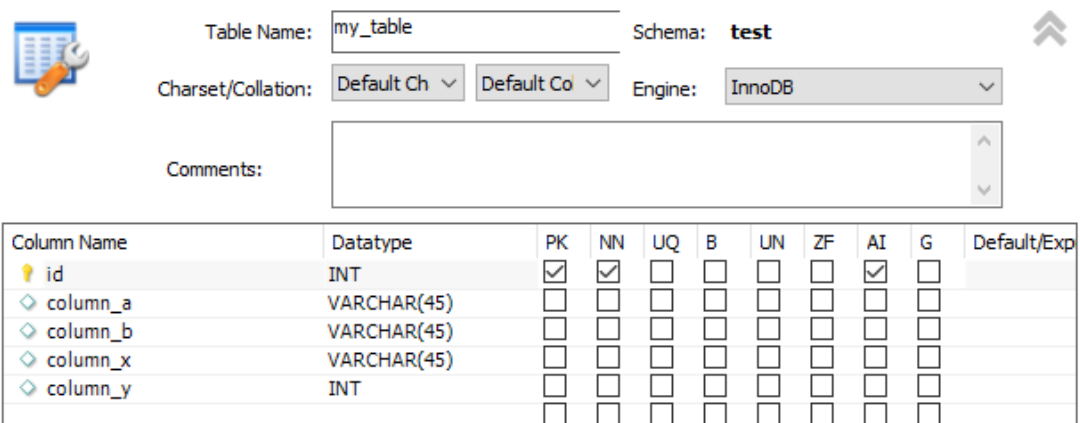
`foreign_table_name_id`

Foreign key column names should be prefixed with the table name that the foreign key is referencing followed by "_id". Thus making it easy to understand what table the foreign key ID is referencing.

Column Names

`column_name`

Use lowercase table names with no numbers or spaces. General Recommendations
Stay consistent with column and table names. For example, do not use "order_no" for a column in one table and "order_number" in another table or "cust_number" in one place and "customer_number" in another.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Exp
id	INT	☒	☒	☐	☐	☐	☐	☒	☐	
column_a	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
column_b	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
column_x	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	
column_y	INT	☐	☐	☐	☐	☐	☐	☐	☐	

Named Queries

Query Names and Folders

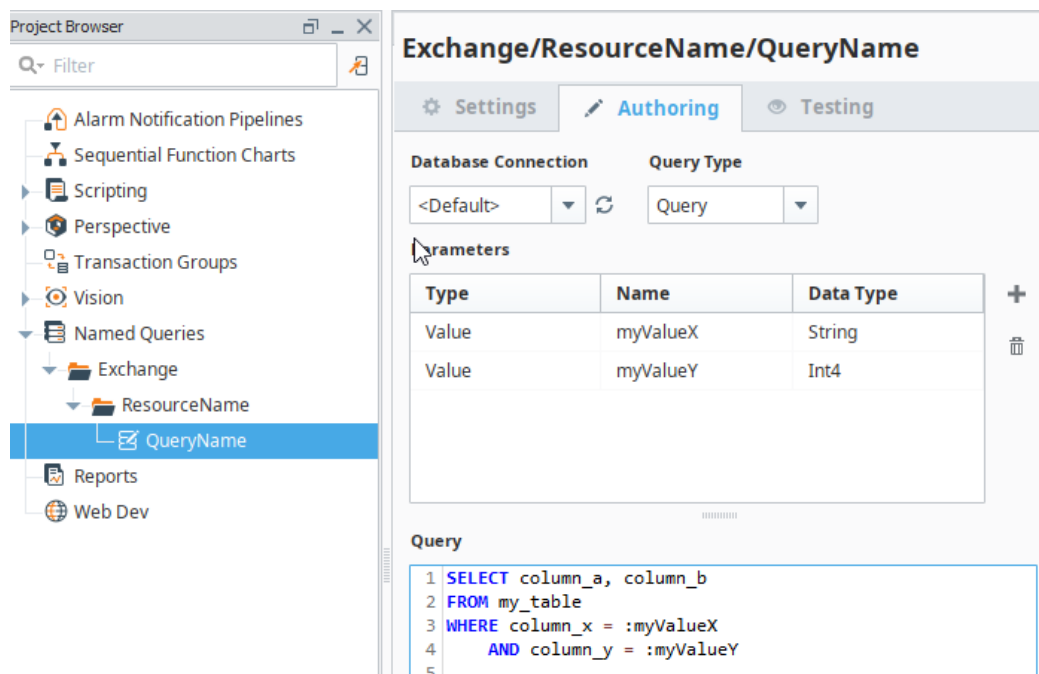
`Query Name` or `QueryName`

Named queries can be either Title Case or PascalCase.

Parameter Names

`parameterName`

Use camelCase, starting with a lowercase letter. These are often used in scripts and component properties, so matching Python variable names and component property name convention makes sense.



Alarm Notification Pipelines

Pipeline Names and Folders

`Exchange/Resource Name/Alarm Pipeline`

or

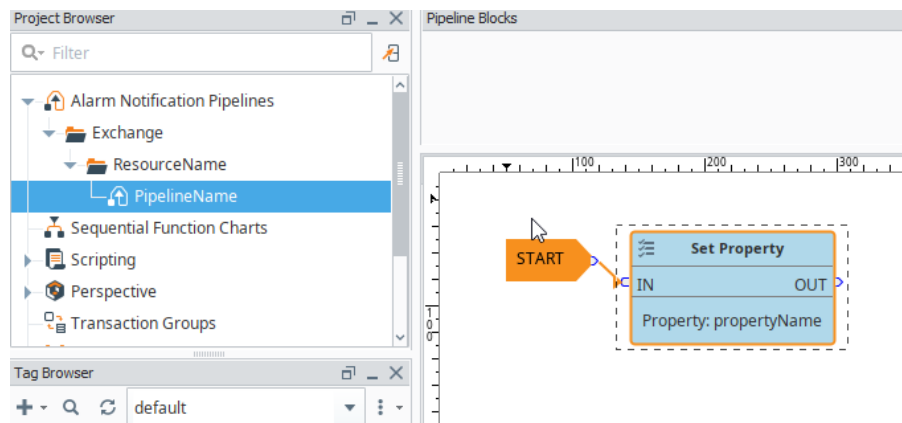
`Exchange/ResourceName/AlarmPipeline`

Alarm pipeline names can be either Title Case or PascalCase.

Custom Properties

`propertyName`

Use camelCase, starting with a lowercase letter. This matches the convention used in built-in properties when accessed through Alarm Pipelines in expression or script blocks.



Sequential Function Charts (SFCs)

SFC Names and Folders

`Exchange/Resource Name/SFC Name`

or

`Exchange/ResourceName/SFCName`

SFC names and folders can be either Title Case or PascalCase.

Chart Parameter Names

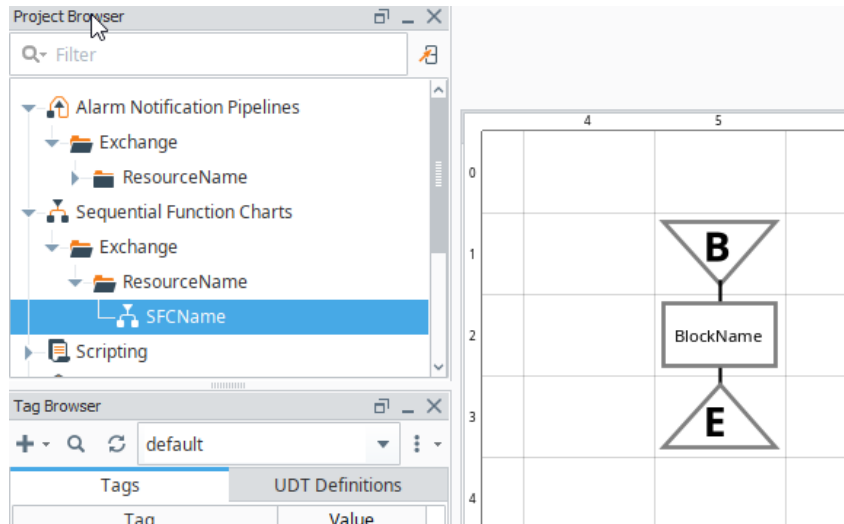
`parameterName`

Use camelCase, starting with a lowercase letter. These are often used in scripts and component properties, so matching Python variable names and component property name convention makes sense.

SFC Block Names

`BlockNames`

Use PascalCase, starting with a capital letter and capitalizing the first letter of each word. Keep block names short and concise and avoid using multiple word names when possible.



SQL Bridge (Transaction Groups)

Transaction Group Names and Folders

[Exchange/Resource Name/New Group](#)

or

[Exchange/ResourceName/NewGroup](#)

Transaction group names and folders can be either Title Case or PascalCase.

Item Names

[Item Name](#) or [ItemName](#)

Item names can be either Title Case or PascalCase.

Item Name	Source V...	Latched ...	Mode	Target Name	Data Type	Properties
Tag Item	N/A	N/A	Use group's mode	Read-only	String	
OPC Item	N/A	N/A	Use group's mode	OPC_Item	String	

Item Name	Source Value	Latched Value	Target Name	Data Type	Properties
Expression Item	N/A	N/A	Read-only	Int4	

Reports

Report Names and Folders

`Exchange/Resource Name/New Report`

or

`Exchange/ResourceName/NewReport`

Report names can be either Title Case or PascalCase.

Parameter Names

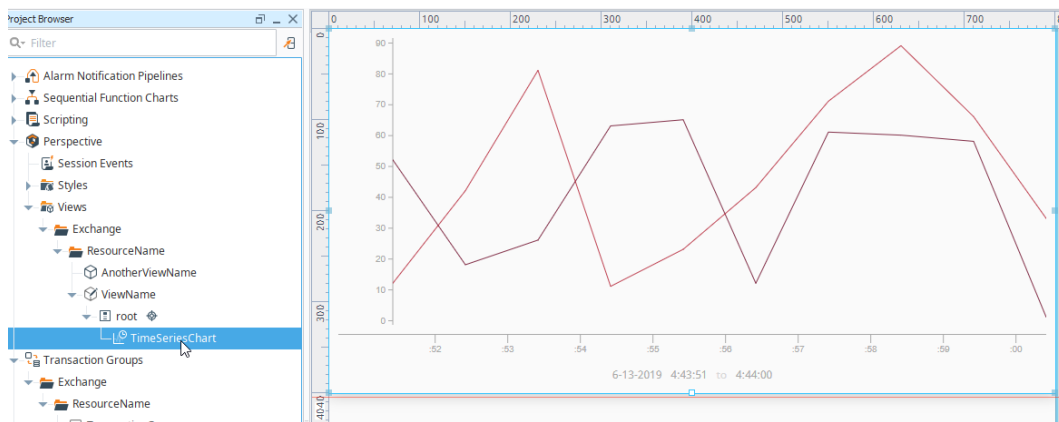
`ItemName`

Report parameters are uppercase with no spaces for multi-worded names. This format matches the default naming convention when adding new parameters.

Component Names

`ComponentName`

Component names should be PascalCase, starting with a capital letter and capitalizing the first letter of each word.

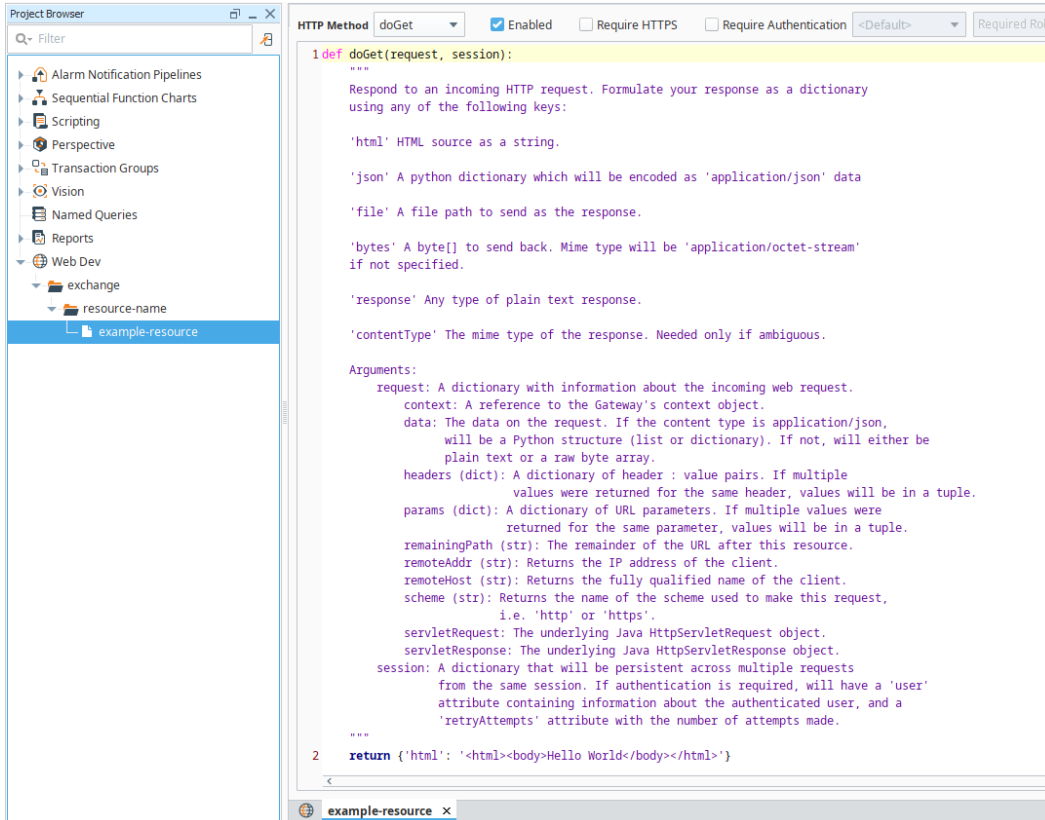


Web Development Module (WebDev)

Source Names and Folders

`exchange/resource-name/example-resource`

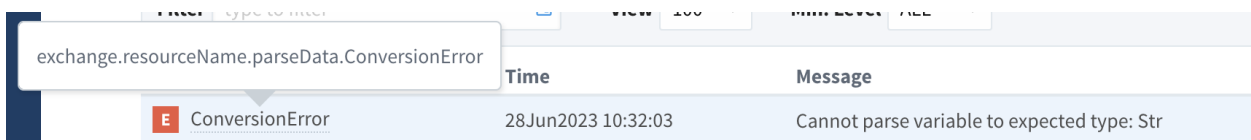
Web Dev should be lowercase with dashes for multiple words. This format ensures that referencing the resource is web friendly.



Loggers

Logger names should use dot separation to create a logging hierarchy, indicating where in the project the logger is used. To match the convention of other loggers used in the Ignition gateway, any logger name elements preceding the final period should be camelcase, and the logger name at the end should be PascalCase. Keep in mind that only the final logger name is shown on the logs, but a mouseover event will show the entire logger name.

It is recommended that loggers are called via scripts in the Project Library instead of directly on views, which can be difficult to find. In other words, scripts containing loggers should be called from views.



Logger naming convention

`exchange.resourceName.LoggerName`

`exchange.resourceName.script.scriptName.LoggerName`

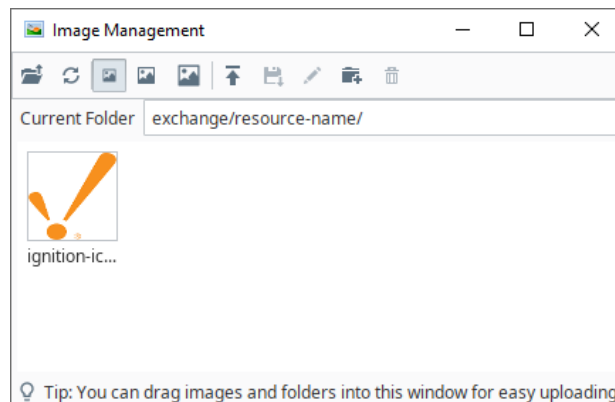
For projects with a large number of scripts in the project library, it may be appropriate to show an abbreviated path to the script. For instance, if the path to the script is “Exchange/MyResource/Utilities/ParseData”, a useful logger name would be “exchange.resourceName.script.utilities.parseData.ConversionError”

Miscellaneous Designer Tools

Image Management

`exchange/resource-name/image-name.jpg`

Images uploaded using the Image Management should be uploaded under a folder hierarchy that starts with a folder named "exchange" followed by another folder named after the resource name. Folders and image names should be lowercase and use dashes between words. Image names should be meaningful and accurately represent the image.

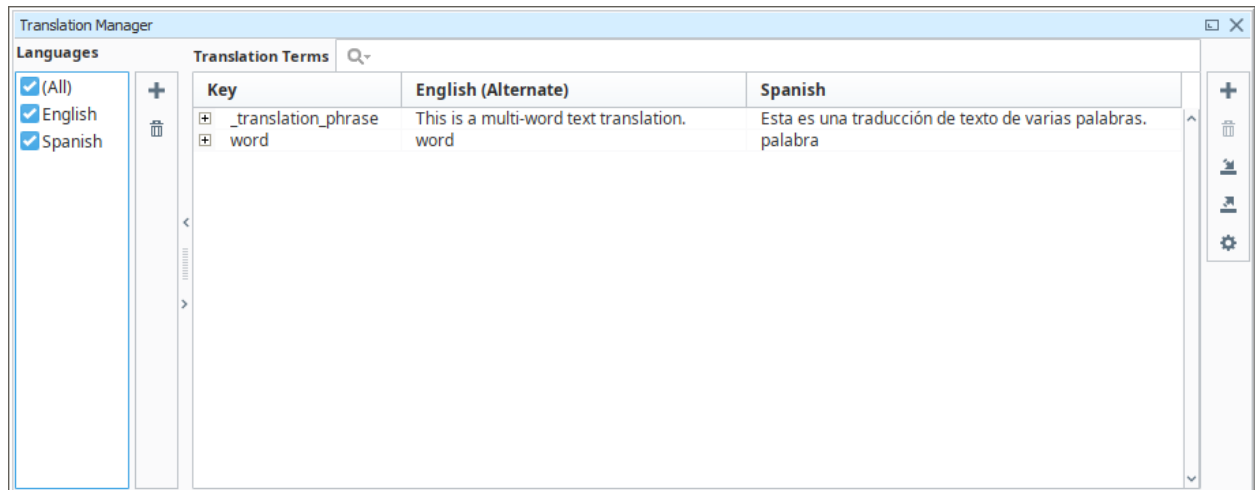


Translation Manager

`word` (single-word translations)

`_translation_phrase` (multi-word translations)

If the translation key is a single word, the key should match the word being translated. If translating a group of text, start the key with an underscore and replace spaces in the key with underscores.



Before You Upload to the Exchange

When your resource is finalized, there are a few steps that should be taken before uploading the resource to the Exchange. The purpose of checking your own resource is to make the process as smooth as possible and get your resource published on the Exchange in a timely manner. Following the steps below will insure this and help mitigate uploading incomplete or broken resources which will result in the resource being rejected until the issues are resolved.

Things to check for:

Check the project export

If the resource to be uploaded is a project export, check that everything that needs to be included in the project was exported correctly. This can be done by importing the exported project into a brand new project and verifying that the resources works as expected.

Include the tags

Does the resource require tags and/or user defined tags (UDTs)? If the resource requires tags, make sure to export the necessary tags and/or UDTs. Then follow the same procedure as [Check the project export](#) and make sure all the tags needed in the project are included in the tag export.

Don't forget database backups

Does the resource reference a predefined table structure or include example data? Include the necessary database dumps when uploading to the Exchange and make sure to state the database vendor (i.e., MySQL, MSSQL, PostgreSQL, etc.) being used in resource description and custom installation instruction step when uploading to the Exchange.

Take notes of the steps taken when checking your project export

When uploading your resource to the Exchange, there will be a section for Custom Installation Instructions. This is where you will give step-by-step instructions on how to set-up your resource to ensure that users get started correctly. It is important to understand that the Ignition experience level of users downloading resources from the Exchange vary from beginners to intermediate so it is important to be as clear and concise as possible.

Uploading to the Exchange

The following steps go over best practices for uploading a project to the Exchange. For each section of the Exchange upload process, please fill out the following to the best of your ability.

Overview Section

Visibility

Unless the project is a private project and not intended for public use, it is recommended that the visibility of the project is set to Public (Visible to everyone).

Visibility

Choose how your resource should appear on the Exchange.
Note: Public resources will be reviewed by Inductive Automation before being published.

Title

The title should be short, concise, and identify the functionality and/or purpose of the project being uploaded. In most cases, if this is a project being uploaded, the title can match the project title given for the project. The resource title should be formatted in Title Case.

Title

Create an easily identifiable title for your resource. Organization resources must have a unique title.

Tagline

The tagline is a short description that accompanies the title to give users a better general understanding of the functionality of the project. This should be a one sentence, short description of the project and should not match the title.

Tagline

Create a short but descriptive tagline to help identify your resource.

Description

The description expands upon the tagline to give a more detailed explanation of the project, how it is meant to be used, what types of people would benefit from this resource, etc.

Description

B *I* ☰ ☷

This Perspective User Management tool allows users to add, edit, and remove users and roles from the Gateway.

Create a detailed description of your resource. Include information like how you envision it being used, what types of people might benefit from this package, any relevant industries, etc.

Resource Type

Choose the best resource type from the drop down that best describes the type of resource that is being uploaded. Then choose a skill level that best fits the complexity of installation and use.

Resource Type

Perspective View ▼

Choose a primary resource type that best describes your resource.

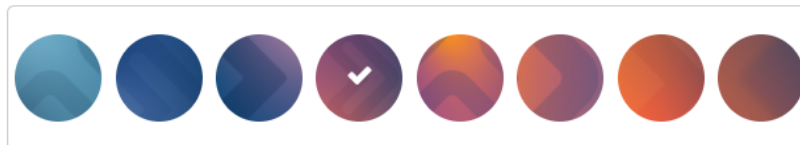
Beginner ▼

Choose a skill level that best describes the complexity of installation and use

Background Image

The background image is an optional step for adding a color and design to the background of the Exchange resource card and page.

Background Image *Optional*



Customize the background of your resource with a colorful design.

Category

The category section is used for filtering Exchange resources. In order to ensure that the resource being uploaded gets proper exposure on the Exchange, select up to 3 categories that best match the resource provided.

Category	Alarming	Analytics	Connectivity	Dashboard
	Diagnostics	Edge Computing	Enterprise	HMI
	I/O/TAGS	IIOT/MQTT	MES	Maintenance
	Mapping/GIS	Mobile	Monitoring	Other
	Reporting	SCADA	Scripting	Security
	Simulation	Trending/Charting	Utility	

Resources are displayed by category. Select up to 3 categories.

Contact the Developer

Enabling the Contact the Developer functionality allows users to directly email the resource developer with any questions or concerns about the resource. There are two options to choose from. The first option being “Let contributors contact me about all my exchange resources”, will allow all visible resources uploaded in your account to be available for contact. The second option “Let contributors contact me about this resource”, will only enable this feature for the resource that is currently being uploaded.

- **Note:** Your email address will not be shown to users that are attempting to contact you. If someone chooses to contact the developer, you will receive an email with the question and the user’s email address. You will then have the ability to contact the user directly.

Contact the Developer
Optional

Let contributors contact me about all my exchange resources ☒

Let contributors contact me about this resource ☒

This feature allows contributors to contact you via email directly from the resource page.

Tags

Tags are another way for the resource to gain exposure on the Exchange. With the ability to create up to 10 tags, these tags will help the resource show up in the results when those tags are used as keywords in the search bar.

Tags

Optional

+

Tags make your resource more searchable. Create up to **10** tags.

User Management

X

Users

X

Images & Screenshots

Images and Screenshots, though optional, are highly encouraged to be included with the resource. Images, for example project views, give users a better understanding of the project that cannot be explained in the title and description alone.

Images & Screenshots

Optional

Drop files here to upload
(up to 10mb)

Or choose a file...

Browse

Package Section

The package section is where the resource files will be uploaded and information about Ignition and module requirements are defined.

Version

Unless uploading a subsequent package version, this package version should be Version 1.0.0 and any subsequent updates will be auto-incremented unless specified by the uploader.

Version

1

0

0

Version will auto-increment by patch unless otherwise specified.

Ignition Platform

The Ignition version should be the minimum version needed in order for the package being uploaded to be installed and run correctly. If unsure, select the Ignition platform version in which the package was developed with.

Ignition Platform

8.1

.3

Choose the minimum version of Ignition required.

Release Tagline

The release tagline summarizes the changes that were made to the package version being uploaded. A title will be needed for the initial release version as well.

Release Tagline

Initial release



Create a short but descriptive tagline to help identify what changes were made in this version.

Release Notes

The release notes give a more detailed explanation of the changes in the package version. Use this area to expand on the release tagline, give more information about the version, and explain how this version will impact the resource. The release version is required even for the initial release version, but may not include as much detail as subsequent version releases of the package.

Release Notes

B *I*

Initial release

List detailed release notes explaining what changed in this version. Think about what improvements were made and how they will impact the resource.

Required Modules

If the resource requires any modules to be installed in order for the package to work correctly, select the required modules by expanding the accordion for the Inductive Automation Modules, Cirrus Link Solutions MQTT Modules, and/or Sepasoft, Inc. MES Modules.

Required
Modules
Optional

Inductive Automation Modules



Cirrus Link Solutions MQTT Modules for Ignition



Sepasoft, Inc. MES Modules for Ignition



Choose any modules that are required when using this resource.



800.266.7798
www.inductiveautomation.com



Maker Edition

Maker Edition is a non-commercial, personal use only version of Ignition which is intended for hobbyists, students, and individuals. If the package being uploaded is intended for Maker Edition or will run in Maker Edition, enabling this option will let users know the package can be installed for their Maker Edition version of Ignition.

Maker Edition
Optional

☒ Let people know this resource is Maker Edition compatible.

This option is unavailable for resources requiring unsupported Maker Edition modules.
[Learn more about Ignition Maker Edition™](#)

Other Requirements

If there are other required items that need to be installed for your resource to work they need to be listed in the Other Requirements section. If there are no additional requirements for your resource this field can be left blank.

Other Requirements
Optional

List any other external or custom requirements as separate line items.

Package Files

Upload all files here that will be needed to ensure your project works as expected. This includes your project backup, tag files, etc.

Package Files


Drop files here to upload
(up to 10mb)

Or choose a file... **Browse**

Uploaded files

UserManagement.zip	108.97 KB	X
--------------------	-----------	---

Identifying common file types will help to reduce the complexity of installation.

Custom Installation Instructions

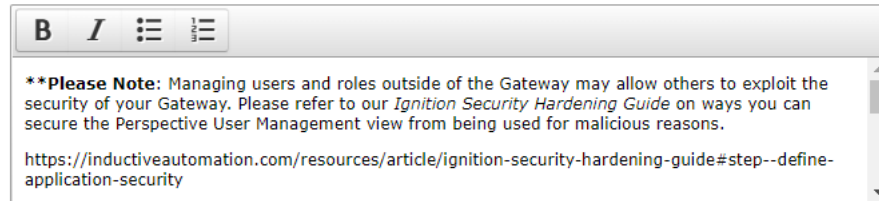
Detailed instructions are highly encouraged for resources as they assist the users that download your resource in understanding how to utilize your resource. These instructions

should include any necessary steps around uploading your resource and any required modules as well as instructions on how to utilize your resource once it is installed.

- **Note:** If you already have detailed documentation for your resource please note it here and include the documentation file in the Package File upload.

Custom Installation Instructions

Optional



****Please Note:** Managing users and roles outside of the Gateway may allow others to exploit the security of your Gateway. Please refer to our *Ignition Security Hardening Guide* on ways you can secure the Perspective User Management view from being used for malicious reasons.

<https://inductiveautomation.com/resources/article/ignition-security-hardening-guide#step--define-application-security>

Provide clear and concise documentation to help other people understand how to install your resource. This is only necessary if you have selected "other" as your filetype for any uploaded files. Installation instructions for known file types will be automatically generated.

Helpful Notes

- The Exchange will automatically generate a readme file from the information you have entered in the Resource Description and Custom Installation Instructions during the upload process
- If publishing your resource in a non-English language we encourage you to include a dropdown that allows your project to be displayed in your native language as well as English. This will increase community access to your project.
- As shown above, capitalized top level resources can be Title Case or PascalCase. Title Case increases readability, matches the default format from the Ignition Designer when creating new resources, can differentiate the resource name from the sub-resources like component names, and creates files on disk with spaces. PascalCase avoids spaces in filenames on disk, is still fairly readable, matches resource names from sub resources like component names, and is preferred by some Ignition users. Both are acceptable choices.