Custom Taxonomies

- Using Custom Taxonomies with WPGraphQL
  - Registering a new Custom Taxonomy
  - Filtering an Existing Custom Taxonomy
- Querying Custom Taxonomies

Using Custom Taxonomies with WPGraphQL

In order to use Custom Taxonomies with WPGraphQL, you must configure the Taxonomy to `show_in_graphql` using the following fields:

Field	Type	Required	Description
`show_in_graphql`	boolean	Yes	If true, show the taxonomy in the GraphQL Schema
`graphql_single_name`	string	Yes	Camel case string with no punctuation or spaces. Needs to start with a letter (not a number)
`graphql_plural_name`	string	No	Camel case string with no punctuation or spaces. Needs to start with a letter (not a number)
`graphql_description`	string	No	Custom description for the type in the GraphQL schema. If not set, will fallback to taxonomy's description, then a default description

Field	Type	Required	Description
`graphql_kind`	string	No	Allows the type representing the taxonomy to be added to the graph as an object type, interface type or union type. Possible values are 'object', 'interface' or 'union'. Default is 'object'
`graphql_resolve_type`	callable	No	The callback used to resolve the type. Only used if "graphql_kind" is set to "union" or "interface"
`graphql_interfaces`	array<string>	No	List of Interface names the type should implement. These will be applied in addition to default interfaces such as "Node"
`graphql_exclude_interfaces`	array<string>	No	List of Interface names the type *should not* implement. This is applied after default and custom interfaces are added
`graphql_fields`	array<$config>	No	Array of fields to add to the Type. Applies if "graphql_kind" is "interface" or "object"
`graphql_exclude_fields`	array<string>	No	Array of fields names to exclude from the type. Applies if "graphql_kind" is "interface" or "object"
`graphql_connections`	array<$config>	No	Array of connection configs to register to the type. Only applies if the "graphql_kind" is "object" or "interface"
`graphql_exclude_connections`	array<string>	No	Array of connection names to exclude from the type
`graphql_union_types`	array<string>	No	Array of possible types the union can resolve to. Only used if "graphql_kind" is set to "union"
`graphql_register_root_field`	boolean	No	Whether to register a field to the RootQuery to query a single node of this type. Default true

Field	Type	Required	Description
`graphql_register_root_connection`	boolean	No	Whether to register a connection to the RootQuery to query multiple nodes of this type. Default true
`graphql_exclude_mutations`	array<string>	No	Array of mutations to prevent from being registered. Possible values are "create", "update", "delete"

Registering a new Custom Taxonomy

This is an example of registering a new "document_tag" Taxonomy to be connected to the "docs" Custom Post Type and enabling GraphQL support.

```
add_action('init', function() {
  register_taxonomy( 'doc_tag', 'docs', [
    'labels'  => [
      'menu_name' => __( 'Document Tags', 'your-textdomain' ), //@see
https://developer.wordpress.org/themes/functionality/internationalization/
    ],
    'show_in_graphql' => true,
    'graphql_single_name' => 'documentTag',
    'graphql_plural_name' => 'documentTags',
  ]);
});
```

Filtering an Existing Custom Taxonomy

If you want to expose a Taxonomy that you don't control the registration for, such as a taxonomy registered by a third-party plugin, you can filter the Taxonomy registration like so, adding any of the arguments documented above:

```
add_filter( 'register_taxonomy_args', function( $args, $taxonomy ) {

  if ( 'doc_tag' === $taxonomy ) {
    $args['show_in_graphql'] = true;
    $args['graphql_single_name'] = 'documentTag';
```

```
    $args['graphql_plural_name'] = 'documentTags';
  }

  return $args;

}, 10, 2 );
```

Querying Custom Taxonomies

Querying terms of Custom Taxonomies is nearly identical to querying Categories and Tags. The difference being the name assigned by `graphql_single_name` and `graphql_plural_name`.

Assuming the taxonomy was registered as shown above, with `graphql_plural_name` set to `documentTags`, you would be able to query like so:

```
{
  documentTags {
    nodes {
      id
      name
    }
  }
}
```

And because the taxonomy was registered in relation to the `docs` Post Type, you'd be able to query the connected nodes like so:

```
{
  documentTags {
    nodes {
      id
      name
      docs {
        nodes {
          id
          title
        }
      }
    }
```

```
  }
}
```

And you'd be able to query a single `documentTag` like so:

```
{
  documentTag(id: "validIdGoesHere") {
    id
    name
  }
}
```

□□□ #1
□ □□□ □□□ 2026-08-29 23:36:26 UTC
□ □□□ □□□ 2026-08-29 23:36:26 UTC