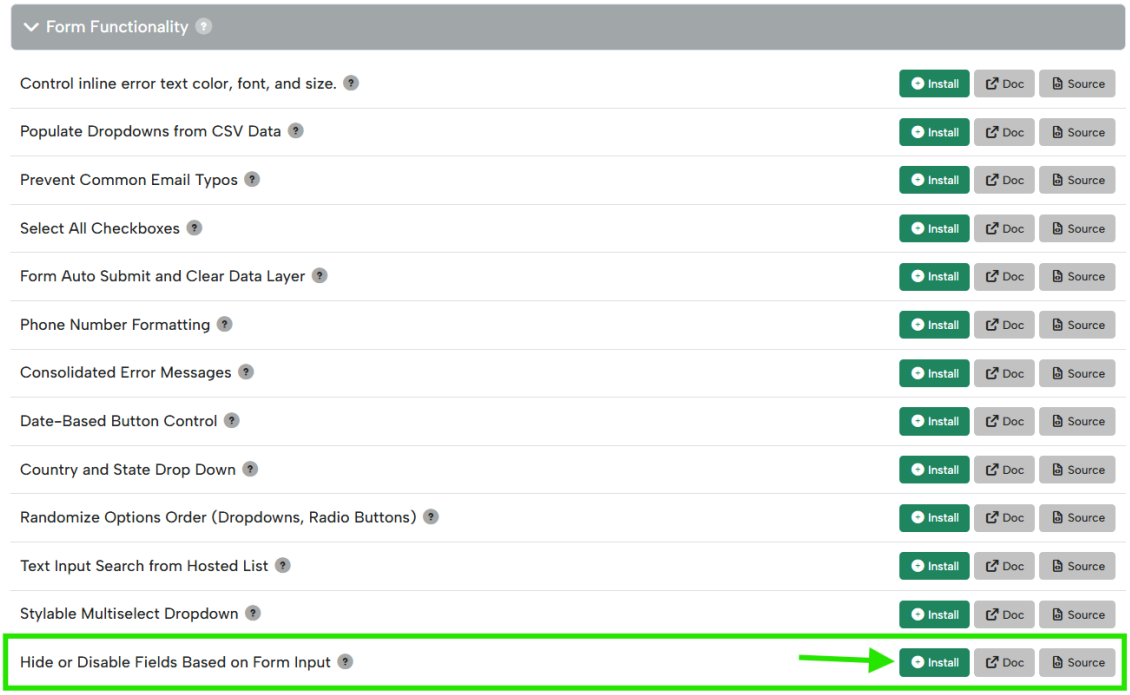
Hide or Disable Fields Based on Other Inputs

Last Modified on 10/29/2026 4:48 pm EDT

Use this extension to show, hide, enable, or disable form fields and campaign elements (images, text, HTML blocks) based on what a user selects or enters in another field. A common use case: show a free-text "Other" field only when a user picks "Other" from a dropdown.

Installation

1. Open your profile menu in the top-right corner of your Digioh Account.
2. Select **Extensions**.
3. Find **Hide or Disable Fields Based on Form Input** under the **Form Functionality** section and install it.

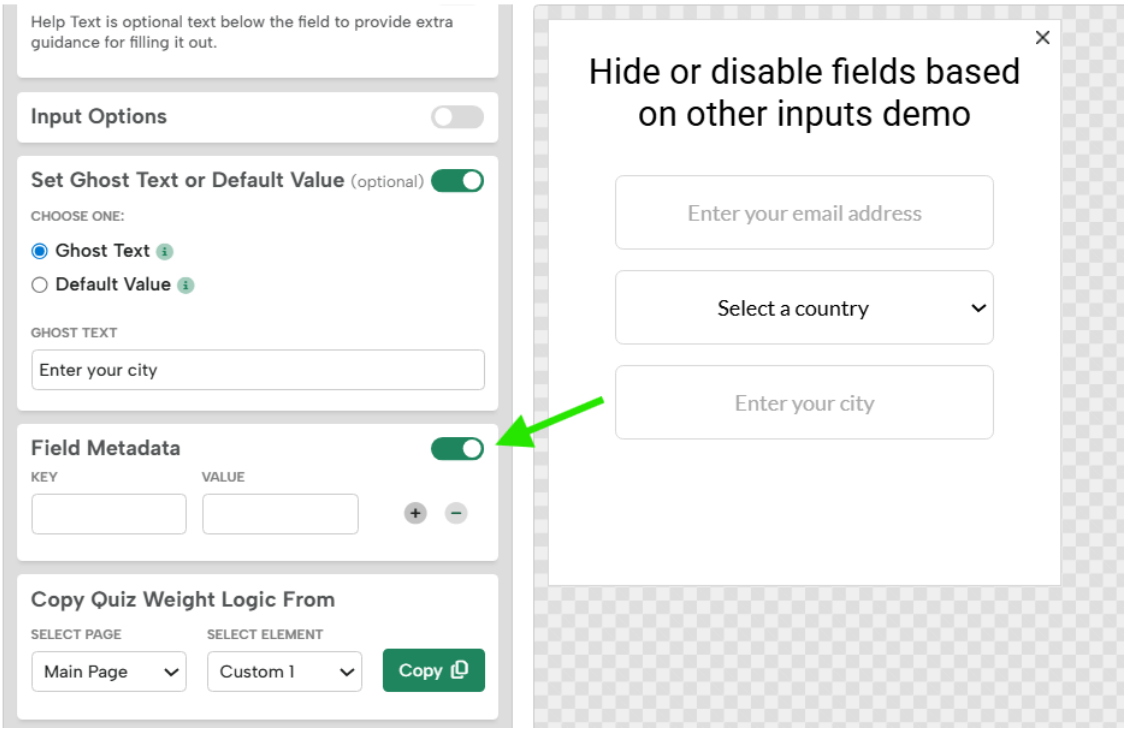


4. Click **Publish to Your Site** to activate the extension.

Configuration

The extension is configured through field-level metadata in the Design Editor. To access the metadata for a field:

1. Open the campaign in the **Design Editor**.
2. Select the field you want to configure: either click it directly on the canvas or choose it from the field dropdown in the **Form** tab.
3. Scroll down to the **Field Metadata** section and toggle it on.



4. Add your metadata key and value there.

Key	What it does
`enable_when`	Activates a field when the condition is met. If the condition is not met, the field stays grayed out and non-interactive.
`show_when`	Shows a field when the condition is met. If the condition is not met, the field stays hidden.

Basic syntax

```
enable_when : fieldName=fieldValue
show_when : fieldName=fieldValue
```

Supported operators

Operator	Meaning	Example
`=`	Field equals one of the values	`custom_1=NZ,CA,DE`
`!=`	Field does not equal any of the values	`custom_1!=NZ,CA,DE`
`*=`	Field contains any of the values	`custom_3*=red,blue`

Matching multiple values

Every operator accepts a comma-separated list of values:

```
show_when : fieldName=value1,value2,value3
```

Equals (`=`)

Triggers when the field's value exactly matches one of the listed values.

```
show_when : custom_1=NZ,CA,DE
```

Does not equal (`!=`)
Triggers when the field has a value that doesn't match any of the listed values.

```
enable_when : custom_1!=NZ,CA,DE
```

> Note: `!=` only triggers once the field has a value. If the user hasn't selected or entered anything yet, the target stays hidden or disabled.

Contains (`*=`)
Triggers when any of the listed values appears in a field that stores several values as a comma-separated list, such as a multi-select field.

```
show_when : custom_3*=red,blue
```

For example, if `custom_3` holds `red,green`, this rule shows the element because `red` is a match. With `=` instead, it wouldn't trigger, because `red,green` isn't an exact match for `red`.

> Note: `*=` matches whole values, not partial text. `custom_3*=re` would not match `red`. To match partial text, use a regular expression (see "Using regular expressions" below).

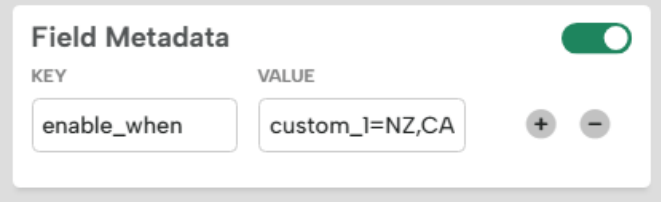
> Note: A "does not contain" operator (`!*=`) is not supported. Rules that use it are skipped, and a warning appears in the QA panel.

Example: show a text field based on a dropdown selection

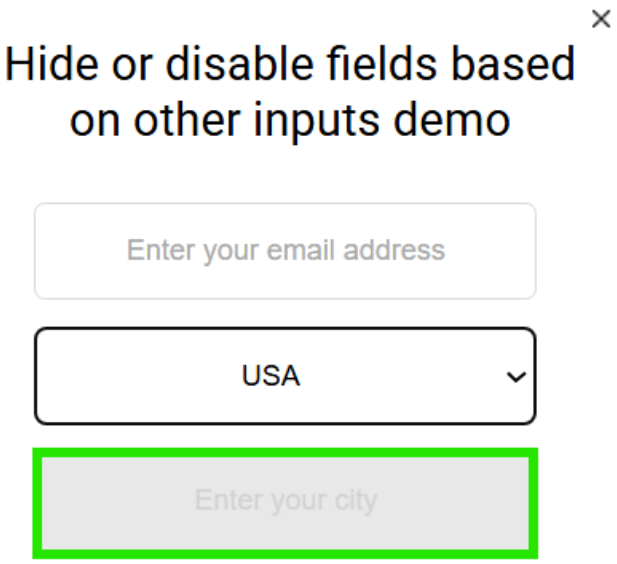
Goal: enable a free-text field (`custom_2`) only when a user picks NZ, CA, or DE from a country dropdown (`custom_1`).

Add this metadata to the `custom_2` field in the Design Editor:

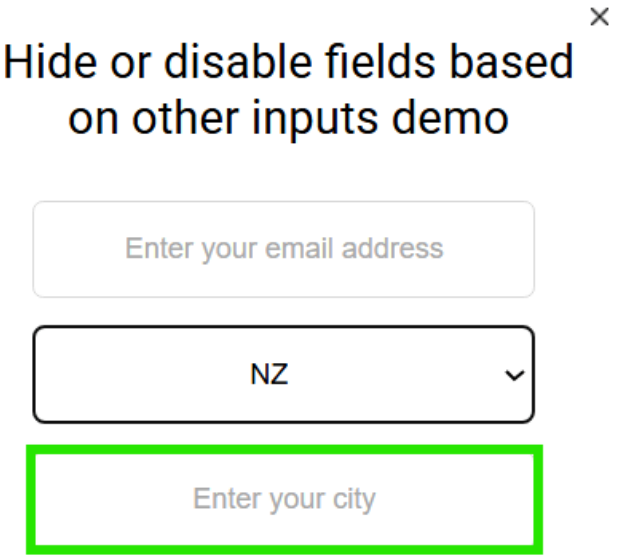
```
enable_when : custom_1=NZ,CA,DE
```



- **Default state:** the text field is grayed out when the user has not selected NZ, CA, or DE.



- **After selection:** the text field becomes active when the user picks NZ, CA, or DE.



To flip the logic and enable the field for every country *except* NZ, CA, and DE, use:

```
enable_when : custom_1!=NZ,CA,DE
```

The `show_when` key works the same way for images and text elements, not just input fields.

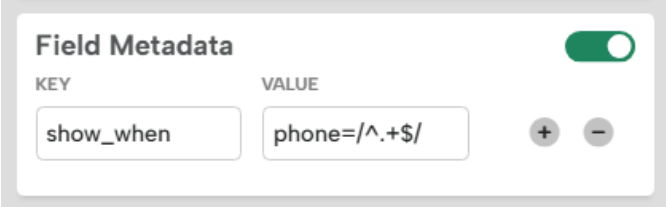
> **HTML blocks:** You can also add `show_when` metadata to an HTML block. The rule shows or hides the entire HTML block. You can't target individual items inside it, so if you need different content per answer, use a separate HTML block for each version and give each its own `show_when` rule.

Using regular expressions

You can use a regular expression instead of a static value. Wrap the pattern in forward slashes (`/ /`) so the extension treats it as a regex:

```
show_when : phone=/^.+$/
```

This example shows a field whenever the phone field contains any value.



Important: V12 regex update (released May 20, 2026): Version 12 includes a fix for how regex conditions were evaluated. In earlier versions, the extension was incorrectly stripping characters from the field value before running the regex test, which could cause some patterns to behave unexpectedly.

> **If you are upgrading from an earlier version to V12:** check any campaign on the account that uses a regex-based `show_when` or `enable_when` rule (any metadata value wrapped in `/ /`) before upgrading. If none exist, it is safe to upgrade with no action needed. If a regex rule stops working as expected after the upgrade, review the pattern, as it may have been relying on the old behavior and may need to be adjusted.

Debug logging

By default, the extension runs silently and only surfaces warnings and errors in the QA notification panel. To turn on verbose logging for a specific campaign:

1. Open the campaign in the **Design Editor**.
2. Click **Settings**, then select **Campaign Settings**.
3. Scroll down to the **Campaign Metadata** tab.
4. Add the following key and value:

Key	Value	Level
`show_hide_debug_logs`	`true`	Campaign

To disable debug logging, remove the key or set its value to anything other than `true` .

What debug logging shows

With debug logging on, you will see in the browser console:

- Each rule evaluated: target field, source field, and current value

- Condition result (whether the condition was met, the operator used)
- Action taken (showing, hiding, enabling, or disabling)
- Event handler triggers (dropdown changes, checkbox changes, button clicks, text input changes)

The following always appear in the QA panel regardless of the debug setting:

- **Warnings:** missing metadata config, wrapper element not found, field not found in the campaign
- **Errors:** rule parse failures, JavaScript exceptions

Note: Use debug logging in development or QA only. In production, the QA panel stays quiet unless something is actually misconfigured.
